UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

SISTEMA DE CONTROLE DE TÁFEGO URBANO UTILIZANDO SISTEMAS MULTI-AGENTES

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

MARCELO SCHMITZ

BLUMENAU, JUNHO/2002

2002/1-51

SISTEMA DE CONTROLE DE TRÁFEGO URBANO UTILIZANDO SISTEMAS MULTI-AGENTES

MARCELO SCHMITZ

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Jomi Fred Hübner — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Jomi Fred Hübner_____

_____

Prof. Mauro Marcelo Mattos

_____

Prof. José Roque Voltolini da Silva

Dedico este trabalho a Deus,

meus amigos

e minha família pelo amor e apoio

durante todos os anos de estudo.

AGRADECIMENTOS

Muitas pessoas contribuíram de maneira direta e indireta para a realização deste trabalho, às quais agradeço de maneira especial.

A todos os meus amigos e professores, que nas horas de dúvidas souberam ser paciente para discutir e esclarecê-las e por ajudarem na obtenção de novas idéias.

Ao meu orientador Jomi Fred Hübner, símbolo de dedicação e amizade, por através de seus ensinamentos ter mostrado e incentivado a conhecer mundo dos Sistemas Multi-Agentes.

E em especial aos meus irmãos Ruth e Ivo que deram amizade e entusiasmo em todas as horas, aos meus parentes pelo exemplo de vida, aos meus avós que me deram a base da perseverança e de querer vencer e aos meus pais, Adilson Antônio Schmitz e Inês Sueli Frutoso Schmitz, que sempre me apoiaram em todos os meus projetos de vida, mostrando segurança, força e determinação.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

MAS - Mult-Agents System (Sistema Multi-Agentes - SMA)

SACI - Simple Agent Communication Infrastructure

KQML - Knowledge Query and Manipulation Language

ITS - Intelligent Transportation Systems (Sistemas de Transportes Inteligentes - STI)

LCA - Linguagem de Comunicação entre Agentes

XML - eXtensible Markup Language

LT - Linguagem de Trânsito

SAX - Simple API for XML

RESUMO

Este trabalho tem como principal objetivo verificar a utilização de técnicas e conceitos de *Sistema Multi-Agentes* (SMA) para o desenvolvimento de um sistema de controle de tráfego urbano, composto por um conjunto de agentes que fazem o papel dos semáforos e um simulador de malha viária, no qual simulações das situações do mundo real poderão ser realizadas. É utilizada como ferramenta de comunicação entre os agentes, o *Simple Agent Communication Infrastructure* (SACI), que utiliza a *Knowledge Query and Manipulation Language* (KQML) como linguagem e protocolo para a troca de informação e conhecimento entre agentes. Para a realização da gerência e controle de tráfego os agentes que representam os semáforos nos cruzamentos, realizam entre si, através da troca de mensagens a negociação do direito de passagem dos veículos em tempo real.

ABSTRACT

This work analyses the techniques and concepts use of a *Multi-Agents System* (MAS) for the development of an urban traffic control system. This system is composed of a set of agents that play the role of the traffic lights and also a traffic system simulator, in which real world situations may be simulated. As a communication tool among the agents it is employed the *Simple Agent Communication Infrastructure* (SACI), which uses the *Knowledge Query and Manipulation Language* (KQML) as language and protocol for the exchange of information and knowledge among agents. For the accomplishment of traffic control management, the agents that play the role of the traffic lights in the crossroads, perform among themselves, through the exchange of messages, the vehicles passageway's rights negotiation in real time.

1 INTRODUÇÃO

O desenvolvimento econômico através dos anos vem evoluindo em uma alta progressão e consigo trouxe uma série de benefícios e dificuldades a todas as nações. Juntamente com o desenvolvimento econômico, vários outros desenvolvimentos surgiram ao longo destes anos.

Segundo MCT (2000), os transportes e o desenvolvimento econômico de uma nação estão fortemente ligados. O desenvolvimento econômico estimula a demanda de transportes aumentando o número de trabalhadores deslocando-se entre seus locais de trabalhos e suas casas, clientes trafegando em áreas de serviços e produtos sendo transportados entre produtores e consumidores. Esta demanda adicional pode gerar, por sua vez, a necessidade de melhoramento no sistema de transporte. Estes melhoramentos diminuem os custos finais de transporte e aumentam a qualidade de vida da sociedade, proporcionando desta forma um novo crescimento econômico.

Para o constante desenvolvimento econômico é necessário que condições de conforto, segurança e bem estar de uma sociedade sejam analisadas e estas condições dependem diretamente de um sistema de transporte que permita o rápido e eficiente transporte de pessoas e de materiais. Percebe-se que desde algum tempo atrás, conforme relata Stern (1969), o ritmo da indústria automobilística nacional acarreta uma rápida saturação das ruas e avenidas dos centros urbanos mal projetados e de capacidade já bastante limitada pelo grande número de novos veículos que entram em circulação.

Verifica-se então que, em função deste grande crescimento, graves problemas afligem o trânsito dos principais centros urbanos. Os congestionamentos constantes, a coordenação dos semáforos entre cruzamentos numa malha viária, os desvios de tráfego, a poluição do ar e sonora, a segurança do motorista, demora às respostas de emergências entre outros repercutem seriamente na economia como um todo.

O congestionamento do tráfego, em particular, causa problemas que molestam todas as grandes cidades do país. Ele acarreta diretamente, por exemplo, em poluição, acidentes de carros, insegurança no trânsito e perda de produtividade. Para os governos todos esses problemas consomem um enorme esforço em serviços e orçamentos.

Para tentar resolver e organizar estes problemas de tráfego, muitas vezes se faz o uso de sinais de trânsitos (semáforos) automatizados que sincronizam mudanças de estado, tentando assim reduzir engarrafamentos e congestionamentos. Entretanto, segundo o MCT (2000), o gerenciamento do tráfego é mais do que simplesmente sincronização de sinais. Ele envolve também o monitoramento do fluxo de tráfego, identificando e interpretando suas obstruções.

Todo este monitoramento, controle e organização do trânsito são, ainda hoje, realizados com poucas ferramentas de apoio e, em geral, precárias. Na maioria das vezes são designadas pessoas para realizar estas tarefas, tentando assim contornar estes problemas de forma e rapidez que a inteligência humana pode dar a estes sistemas.

Sincronizações de sinais de trânsito são, na maioria das vezes, realizadas sem algum estudo prévio, ou quando são realizados estudos são praticados em apenas alguns trechos urbanos, resolvendo inicialmente o problema no determinado local, mas de outra forma debilitando em outras localidades. Verifica-se ainda que na maioria dos centros urbanos o trânsito não possui um fluxo uniforme em todo o seu período de tempo. Existem períodos do dia onde o tráfego exige maior fluidez e em outros horários, menor fluidez, causando assim um trânsito caótico e impossível de ser administrado somente por pessoas.

Portanto sendo que as informações para gerência de trânsito encontram-se distribuídas em uma localidade, onde muitas dessas informações podem sofrer modificações em tempo real, é de grande dificuldade a tentativa de uma gerência global centralizada.

Em função do acima exposto vislumbrou-se a possibilidade de pesquisar tecnologias que pudessem ser utilizadas de tal forma a cooperar para uma melhor solução destes problemas. Conforme citado em MCT (2000), a aplicação de tecnologias modernas de computadores e de comunicação nos sistemas de transportes, tem resultado uma maior mobilidade, segurança e qualidade do ar nas cidades, além do próprio aumento de produtividade. Sobre a aplicação destas tecnologias denominadas de *Intelligent Transportation Systems* - Sistemas de Transportes Inteligentes (ITS), é possível dar um salto de qualidade no controle e gerência de trânsito e dos transportes (DOT, 2001).

1.1 OBJETIVO

O objetivo principal deste trabalho é utilizar os conceitos de *Sistema Multi-Agentes* (SMA), e desenvolver um simulador de malha viária, no qual simulações das situações do mundo real podem ser realizadas, visando gerenciar e controlar o tráfego urbano através da gerência dos semáforos nos cruzamentos.

1.2 ESTRUTURA DO TRABALHO

Dados os objetivos apresentados neste capítulo, no capítulo 2 faz-se uma abordagem sobre trânsito, apresentando conceitos sobre circulação e classificação das vias, definições sobre sinalização semafórica e fórmulas para a realização de cálculos dos tempos semafóricos a serem aplicados na elaboração do trabalho.

No capítulo 3 faz-se uma abordagem sobre *Sistemas Multi-Agentes* (SMA) expondo vantagens e características de sua aplicação, em seguida faz-se uma abordagem sobre *Knowledge Query and Manipulation Language* (KQML), que é uma *Linguagem de Comunicação entre Agentes* (LCA) utilizada pelo *Simple Agent Communication Infrastructure* (SACI), sendo esta uma ferramenta para o desenvolvimento da comunicação entre os agentes.

No capítulo 4 encontra-se a especificação do Sistema Multi-Agentes para o Sistema Semafórico, contendo uma visão geral, o protocolo de comunicação entre agentes, a especificação dos agentes e interface para a utilização pelo usuário.

No capítulo 5 encontra-se a implementação da interface e dos agentes, com ilustrações das classes implementadas.

No capítulo 6 encontram-se os resultados obtidos.

No capítulo 7 encontram-se as considerações finais do trabalho.

2 TRÂNSITO

Considera-se trânsito a utilização das vias por pessoas, veículos e animais, isolados ou em grupos, conduzidos ou não, para fins de circulação, parada estacionamento e operação de carga ou descarga (Lopes, 1998).

Cada vez mais o trânsito se mostra mais complexo em nossa sociedade, apresentando novas e desafiantes problemáticas àqueles que têm a incumbência de administrá-lo ou entendê-lo com o objetivo de propor novas soluções.

O crescimento das cidades gera um aumento do fluxo de tráfego das vias em geral. A intensidade dos fluxos de veículos e de pedestres nas vias, são fatores fundamentais para estudo de controle de tráfego.

2.1 CIRCULAÇÃO DE SENTIDO DUPLO

As vias com circulação natural, não requerendo sinalização estabelecendo sua circulação são denominadas Vias de Sentido Duplo, conhecidas também como vias de mão dupla. Estas vias têm como vantagens básicas a melhor acessibilidade aos seus imóveis e melhor acesso aos serviços de ônibus, os quais podem circular em ambos os sentidos da mesma via, facilitando a utilização pelos usuários. As desvantagens são de riscos de colisões frontais, conflitos nas conversões à esquerda e maior dificuldade na travessia dos pedestres.

2.2 CIRCULAÇÃO DE SENTIDO ÚNICO

As Vias de Sentido Duplo podem ser transformadas em Vias de Sentido Único, ou conhecidas também como vias de mão única, sendo escolhido um sentido para sua operação e sinalizadas para esse fim. O motivo principal para essa transformação é a saturação da via como mão dupla, causando problemas de segurança e fluidez no tráfego. Tem como vantagens a eliminação de riscos de colisão frontal, maior segurança na travessia de pedestres, redução do número de movimentos conflitantes nos cruzamentos, aumento da capacidade viária proporcionando maior fluidez, e permite ótima progressão dos semáforos. As desvantagens principais são o aumento da velocidade, podendo aumentar os acidentes em determinados horários e menor acessibilidade.

2.3 CLASSIFICAÇÃO DAS VIAS

As vias podem ser classificadas de acordo com seu tipo de fluxo, influenciado na maioria das vezes pelo tipo de cruzamento, cruzamento em desnível (pontes e viadutos) e cruzamento em nível (intersecções de ruas):

a) vias de fluxo ininterrupto: são aquelas concebidas para não haver restrições à passagem do fluxo de tráfego, havendo controle de acesso e com isso a quase inexistência de movimentos conflitantes. Os cruzamentos importantes são em desnível, e os cruzamentos em nível existentes geralmente não influenciam no escoamento do tráfego. São vias de fluxo ininterrupto as estradas e vias expressas;
b) vias de fluxo interrompido: são as demais vias em que existem cruzamentos em nível e que há a necessidade de se controlar os fluxos que se cruzam através de dispositivos de controle, placas sinalizadoras para estabelecimento das vias preferenciais ou semáforos que alternam os momentos de direito de passagem das vias.

Na área urbana, são predominantes as vias de fluxo interrompido, as quais requerem tratamentos na administração dos fluxos que se cruzam nas inúmeras intersecções em nível. Ejzemberg (1996) ao tratar da análise da circulação e fluxos de tráfego aponta que o tratamento e controle das intersecções são os principais fatores determinantes da capacidade do sistema viário, além de influir significativamente na segurança.

Conforme Espel (2000), muitos cruzamentos geram resistência à passagem dos veículos, seja por fatores geométricos e de topografia, e mais comumente quando os volumes de tráfego atingem níveis onde começa a haver a “disputa” do espaço comum na chamada área de conflito do cruzamento.

Para resolver estes problemas existem soluções tais como: sinalização de advertência, mini-rotatórias, canalização, rotatórias, semáforos, pontes, viadutos, túneis etc.

Em cidades de médio e grande porte onde já se encontram cruzamentos e um novo projeto dos mesmos se torna inviável e muitas vezes impossível, é que existe uma grande concentração de semáforos, onde a implantação de semáforos não afeta muito a geografia dos cruzamentos.

2.4 SINALIZAÇÃO SEMAFÓRICA

Conforme Lopes (1998), a sinalização semafórica é um subsistema da sinalização viária que se compõe de luzes acionadas alternada ou intermitentemente através de sistema elétrico/eletrônico, cuja função é controlar os deslocamentos, podendo ser de dois tipos:

a) sinalização de advertência: tem a função de advertir da existência de obstáculos ou situação perigosa, devendo o condutor reduzir a velocidade e adotar medidas de precaução compatíveis com a segurança para seguir adiante;
b) sinalização de regulamentação: é composta de luzes de cores pré-estabelecidas, agrupadas em um único conjunto, disposto verticalmente ao lado da via ou suspenso sobre ela. Tem a função de controlar o trânsito, alternando o direito de passagem.

2.4.1 SINALIZAÇÃO SEMAFÓRICA PARA VEÍCULOS

Segundo Lopes (1998), as cores estabelecidas pelo Código de Trânsito Brasileiro para a sinalização semafórica de veículos são:

a) vermelha: indica obrigatoriedade de parar;
b) amarela: indica “atenção”, devendo o condutor parar o veículo, salvo se isto resultar em situação de perigo para os veículos que se encontram atrás;
c) verde: indica permissão de prosseguir na marcha, efetuando o condutor a operação indicada pelo sinal luminoso.

O Código de Trânsito Brasileiro, em seu anexo II, conforme Lopes (1998), prevê a possibilidade de controle semafórico a partir das luzes vermelha e verde, somente. Nesse caso, o comando do amarelo é substituído pelas luzes acessas ao mesmo tempo.

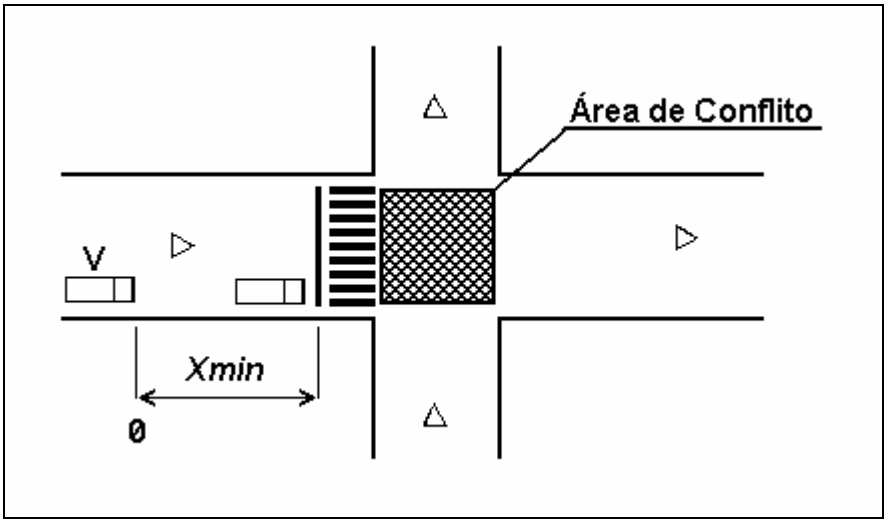
2.4.2 TEMPOS SEMAFÓRICOS

Segundo Espel (2000) a programação dos tempos semafóricos consiste na obtenção dos valores dos tempos de ciclo para elaboração dos planos de tráfego necessários para a operação dos cruzamentos.

2.4.2.1 TEMPO DE AMARELO

A utilização da luz amarela entre o verde e o vermelho no controle semafórico é necessária por não ser possível parar instantaneamente um veículo. Sua finalidade é a de avisar ao condutor da iminência do vermelho e, portanto que ele deverá decidir se há tempo para passar ou frear. Para a realizar o cálculo deste tempo, considera-se um veículo com velocidade *"V"*, conforme representado pela fig. 1.

FIGURA 1 - VARIÁVEIS DO TEMPO DE AMARELO



Fonte: adaptado de Mcshane (1990).

Sendo:

0 = origem, ou seja, posição quando o veículo recebe o amarelo;

X_{min} = distância mínima a ser percorrida até a linha de retenção;

T_{pr} = tempo de percepção e reação.

Devido ao tempo de percepção e reação, o veículo irá percorrer um determinado espaço (X_{pr}), onde:

$$X_{pr} = V \cdot T_{pr} \quad \text{(Equação I)}$$

Conforme Vilanova (1985), o tempo de percepção e reação (T_{pr}) do motorista é de 0,8*s* a 1.2*s*, sendo o ideal 1.0*s*. Até reagir ao tempo de amarelo que se iniciou, ele tem duas opções: frear o veículo ou seguir jornada.

Caso o motorista resolva frear o veículo, após o tempo de percepção e reação (T_{pr}), o tempo restante de amarelo (T_{ra}) deve ser um valor tal que o veículo pare na faixa de retenção no início do vermelho.

Usando a equação de Torricelli (Santos 2001), tem-se:

$V_f^2 - V_0^2 = 2 . a . s$

Sendo:

a = aceleração (m/s^2);

s = espaço percorrido (m);

V_0 = velocidade inicial (m/s);

V_f = velocidade final (m/s).

Para este caso tem-se:

$V_0 = 0$ e $s = X_{ra}$ = espaço percorrido após a reação à sinalização da luz amarela.

Então:

$0 - V^2 = 2 . (-a) . X_{ra}$, ($-a$, pois se trata de desaceleração). Portanto,

$$X_{ra} = \frac{V^2}{2 \cdot a}$$ (Equação II)

A distância mínima (X_{min}), mostrada na Figura 1, tem, então, dois componentes:

$$X_{min} = X_{pr} + X_a$$ (Equação III)

Substituindo-se as equações I e II na III, tem-se:

$$X_{min} = V \cdot T_{pr} + \frac{V^2}{2 \cdot a}$$ (Equação IV)

De outro modo, caso o motorista resolva seguir jornada, no início da luz amarela com a sua velocidade atual, esse tempo do amarelo deverá ser suficiente para que ele atinja a área de conflito do cruzamento.

Logo,

$X_{min} = V \cdot T_a$ (Equação V)

Onde T_a = tempo de amarelo.

Deve-se partir do princípio que o tempo de amarelo deverá atender tanto se o motorista resolver frear o veículo como se o motorista resolver seguir jornada.

Igualando-se as equações IV e V, tem-se:

$$T_a = T_{pr} + \frac{V}{2 \cdot a}$$ (Equação VI)

No caso em que o motorista resolver frear o veículo, ele conseguirá frear a tempo e na pior hipótese junto à retenção.

Por outro lado, no caso em que o motorista resolva seguir jornada, ele segue em marcha constante e na pior hipótese chega junto à área de conflito e dependendo da largura da via, poderá ser necessário um tempo de segurança para que ele possa efetuar a travessia total do cruzamento.

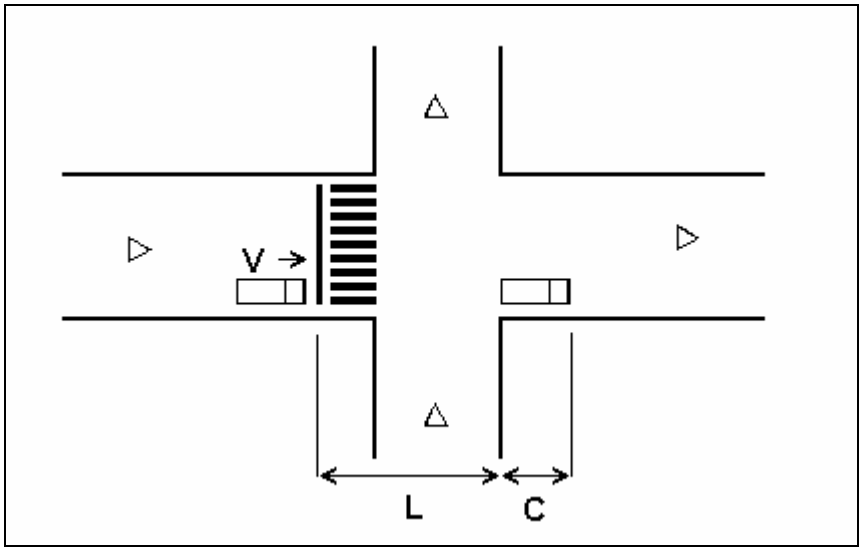
Segundo Vilanova (1985), os valores encontrados para a desaceleração máxima (a) aceita pelos motoristas, varia entre 2,0m/s^2 e 4,2m/s^2, sendo recomendado a adoção do valor 2,8m/s^2.

Adotando-se na equação VI os valores T_{pr} = 1,0s e a = 2,8m/s^2, tem-se o valor do tempo de amarelo (T_a), onde por medidas de segurança os arredondamentos deverão ser feitos para cima.

2.4.2.2 TEMPO DE VERMELHO DE SEGURANÇA

No dimensionamento do tempo de amarelo, observa-se que na pior situação, o veículo recebe o amarelo exatamente na origem, portanto deve-se calcular um tempo de segurança, o qual é denominado vermelho de segurança, sendo este tempo o necessário para que o veículo ultrapasse o cruzamento conforme representado pela fig. 2.

FIGURA 2 - VARIÁVEIS DO TEMPO DE VERMELHO DE SEGURANÇA



Fonte: adaptado de Mcshane (1990).

Sendo:

$$T_{vs} = \frac{(L + C)}{V} \quad \text{(Equação VII)}$$

Onde:

T_{vs} = Tempo de vermelho de Segurança;

L = Largura do cruzamento incluindo a faixa de pedestre anterior;

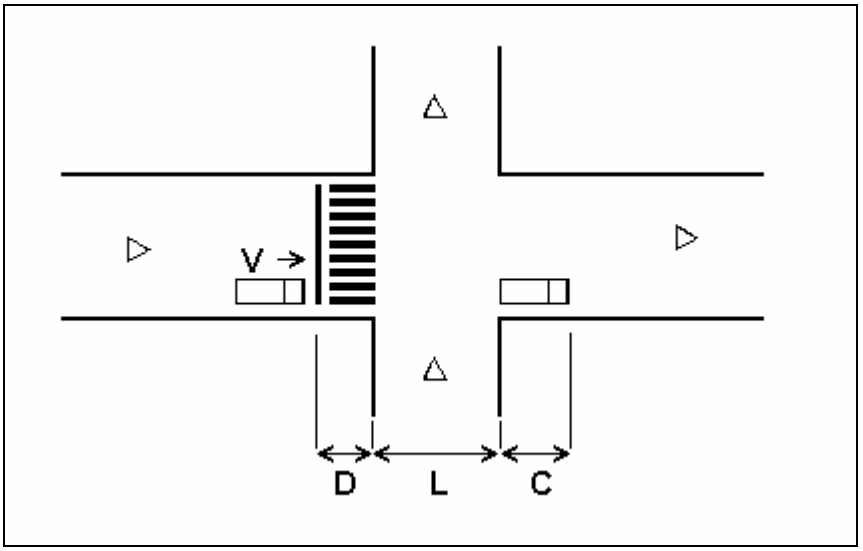
C = Comprimento do veículo;

V = Velocidade do veículo.

2.4.2.3 TEMPO DE VERDE MÍNIMO DE ESTÁGIO

É o mínimo tempo de verde que permite a saída de um veículo da retenção até a transposição do cruzamento com segurança (fig. 3).

FIGURA 3 - VARIÁVEIS DO TEMPO DE VERDE MÍNIMO DE ESTÁGIO



Fonte: adaptado de Mcshane (1990).

Sendo:

$$T_t = \sqrt{\frac{2 \cdot (D + L + C)}{a}}$$ (Equação VIII)

Onde:

T_t = tempo de travessia;

D = distância de retenção do cruzamento;

L = comprimento do cruzamento;

C = comprimento médio de um veículo;

a = aceleração do veículo.

Conforme Mcshane (1990), para este tempo deve-se adicionar um tempo de percepção e reação do motorista (T_{pr}), sendo para este caso 1,5*s*.

2.4.2.4 TEMPO DE VERDE DE ESCOAMENTO

Durante a fase vermelha é esperado a formação de uma fila. Para que esta fila possa escoar deve ser calculado um tempo de escoamento, sendo este o tempo de verde de escoamento, onde o mesmo é rapidamente equacionado em função da quantidade de carros.

Sendo:

$$T_{e1} = T_a + (N_1 - 1) \cdot \Delta t + \frac{C_v}{V_m} + \frac{(N_1 \cdot (C + 0{,}5)) + C_v}{V} \quad \text{(Equação IX)}$$

Onde:

T_{e1} = tempo de escoamento procurado;

T_a = tempo de atraso do primeiro veículo imediatamente após a abertura do sinal;

N_1 = número de Veículos em fila;

Dt = intervalo de tempo entre o arranque de dois veículos sucessivos;

C_v = distância percorrida durante o período de embalagem;

V_m = velocidade média durante o período de embalagem;

C = comprimento médio de um veículo;

V = velocidade do regime.

Segundo Stern (1969), normalmente tem-se os valores de T_a = 3,0*s*, Dt = 1,5*s*, C_v = 60,00*mt*, C = 4,00*mt* e $V_m = \frac{V}{2}$.

2.4.2.5 TEMPO DE DEFASAGEM

O tempo de defasagem (T_d) é o intervalo de tempo entre os instantes de início e final de verde de dois ou mais semáforos próximos.

Teoricamente visa que os veículos que partem do início do tempo de verde de um semáforo, após um determinado tempo de percurso até o próximo semáforo, consiga passar por este também no seu instante inicial de verde, havendo aproveitamento máximo dos tempos de verde de todo o sistema, objetivando minimizar ao máximo os inevitáveis atrasos e paradas que os semáforos ocasionam. Este resultado é comumente conhecido como “onda-verde”.

3 SISTEMAS MULTI-AGENTES

A associação de um conjunto de agentes distribuídos com um objetivo global comun interagindo entre si define um *Sistema Multi-Agentes* (SMA). Mas um SMA é mais do que uma simples reunião de agentes. Conforme Hübner (1995) SMA é uma abordagem para resolver problemas específicos dividindo o trabalho entre muitos agentes que cooperam interagindo e trocando conhecimentos sobre o problema e a sua solução, sendo que estes agentes são dotados de certa autonomia e inteligência.

Maiores informações sobre SMA podem ser encontradas em Bigus (1997), Weiss (1999), Jennings (1998) e Meneses (2001).

As duas principais características de um agente num SMA são perceber e agir. A habilidade de comunicar-se faz parte da percepção (quando recebe mensagens) e da ação (quando envia mensagens). Em SMA é necessário que a comunicação seja disciplinada para que os objetivos sejam alcançados. A comunicação permite que os agentes coordenem suas atividades e seu comportamento, resultando num sistema mais coerente (Meneses, 2001).

Assim a comunicação entre agentes está baseada na troca de mensagens, podendo ser realizadas de três formas: *ponto-a-ponto*, *multicast* ou *broadcast*. A forma como estas mensagens são trocadas entre os agentes são definidas pela *Linguagem de Comunicação entre Agentes* (LCA), onde segundo Meneses (2001) é um fator crítico na integração de agentes e sua expressividade define a capacidade de comunicação do agente.

Uma das LCA que procura implementar as características relevantes (forma, conteúdo, semântica, implementação, rede, ambiente e confiabilidade) é *Knowledge Query and Manipulation Language* (KQML) que é uma linguagem e um protocolo para troca de informação e conhecimento entre agentes.

3.1 KQML

KQML é uma LCA de alto nível, onde a troca de mensagens é independente da linguagem de conteúdo e da ontologia utilizada, ou seja, qualquer tipo de conteúdo pode ser enviado em uma mensagem KQML.

Foi desenvolvida dentro do *"Knowledge Sharing Effort"* patrocinado pelo DARPA, em 1993. KQML é independente do mecanismo de transporte (TCP/IP, SMTP, etc), independente da linguagem conteúdo (KIF, SQL, Prolog, etc), e independente da ontologia assumida pelo conteúdo (Meneses, 2001).

As mensagens KQML podem ser divididas em três camadas: camada de conteúdo, camada de comunicação, camada de mensagem

O formato das mensagens KQML é baseado na sintaxe do LISP conforme ilustra quadro 1.

QUADRO 1 - FORMATO DE MENSAGEM KQML

(*performative*		
: language	*word*	} camada de mensagem
: ontology	*word*	
: sender	*word*	
: receiver	*word*	} camada de comunicaçã o
: reply - with	*word*	
: content	*expresion*	} camada de conteúdo
...)		

Fonte: adaptado de Húbner (2000).

A camada de conteúdo é a que transporta, de fato, o conteúdo da mensagem na própria linguagem de representação do programa. Toda implementação em KQML ignora a porção de conteúdo da mensagem, exceto a que determina onde ela termina, portanto, KQML pode levar qualquer linguagem de representação, tanto em ASCII como em binário.

A camada de mensagem é a mais importante da KQML. Esta parte determina os tipos de interação que um agente pode ter com outro, bem como que operação deve ser executada com conteúdo. A função primária desta camada é identificar o protocolo de rede que vai ser usado para entregar a mensagem e a ação de fala ou **performativa** com a qual o remetente está enviando na mensagem (Meneses, 2001)

A camada de mensagem pode também definir outras opções, tais como, a **linguagem** em que está representado, o conteúdo e **ontologia** assumida que significa um dicionário para a interpretação dos termos da linguagem, em outras palavras, um vocabulário comum entre as

partes envolvidas na comunicação. Estas opções possibilitam que implementações de KMQL analisem, roteiem e entreguem corretamente as mensagens cujo conteúdo é transparente.

A linguagem KQML possui um conjunto de performativas que refletem a intenção do agente emissor da mensagem e que são organizadas em categorias básicas como:

a) consulta básica (evaluate, ask-if, ask-in, ask-one, ask-all, ...);
b) consultas com múltiplas respostas (stream-it, stream-all, …);
c) resposta (reply, sorry, …);
d) afirmação genérica (tell, achieve, cancel, untell, unachieve,...);
e) anúncio e definição de capacidades (advertise, subscribe, import, export, …);
f) rede (register, unregister, forward, broadcast,...).

Este conjunto não é fechado e nem mínimo. Um agente que utilize KQML pode escolher apenas algumas performativas para entender e tratar, podendo ser também este conjunto extensível, utilizando performativas adicionais.

3.2 SIMPLE AGENT COMMUNICATION INFRASTRUCTURE (SACI)

Simple Agent Communication Infrastructure (SACI) é uma ferramenta que torna a programação da comunicação entre agentes distribuídos mais fácil, em conformidade com um padrão, rápida e robusta (Hübner, 2001).

O Saci foi desenvolvido com base na especificação KQML, possuindo as seguintes características principais (Hübner, 2001):

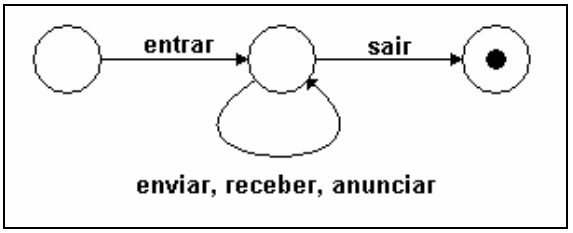
a) os agentes utilizam KQML para se comunicar: há funções para compor, enviar e receber mensagens KQML;
b) os agentes são identificados por um nome: as mensagens são transportadas utilizando-se somente o nome do receptor, sua localização na rede é transparente;
c) um agente pode conhecer os outros por meio de um serviço de páginas amarelas: agentes podem registrar seus serviços no facilitador e perguntá-lo sobre que serviços são oferecidos por quais agentes;
d) os agentes podem ser implementados como *applets* e terem sua interface em uma *home-page*;

e) os agentes podem ser iniciados remotamente;

f) os agentes podem ser monitorados: os eventos sociais (entrada na sociedade, saída, recebimento ou envio de mensagens) podem ser visualizados e armazenados para análise futura.

No SACI os agentes estão agrupados em sociedades, possuindo uma identificação única, onde cada agente interage com os demais utilizando uma linguagem comum. Estes agentes oferecem serviços aos demais agentes de sua sociedade, tendo os agentes um ciclo de vida conforme ilustrado na fig. 4:

FIGURA 4 - CICLO DE VIDA DE UM AGENTE

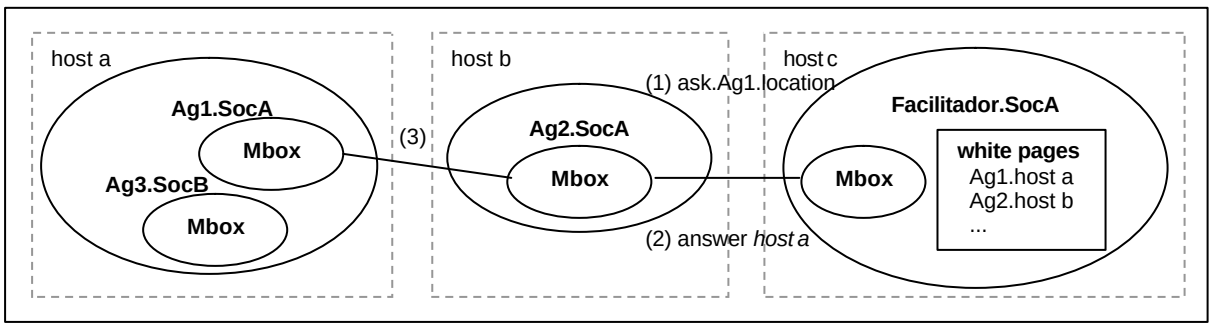


Fonte: adaptado de Hübner (2001)

3.2.1 COMUNICAÇÃO NO SACI

Para o envio e recebimento de mensagens o SACI possui um componente denominado MBox que serve como interface entre o agente e a sociedade. Conforme Hübner (2001), a finalidade do MBox é tornar transparente o envio e o recebimento de mensagens. Este componente possui funções que encapsulam a composição de mensagens KQML, o envio síncrono e assíncrono de mensagens, o recebimento de mensagens, o anúncio e a consulta de habilidades e o *broadcast* de mensagens, conforme ilustra fig. 5.

FIGURA 5 - ENVIO E RECEBIMENTO DE MENSAGENS

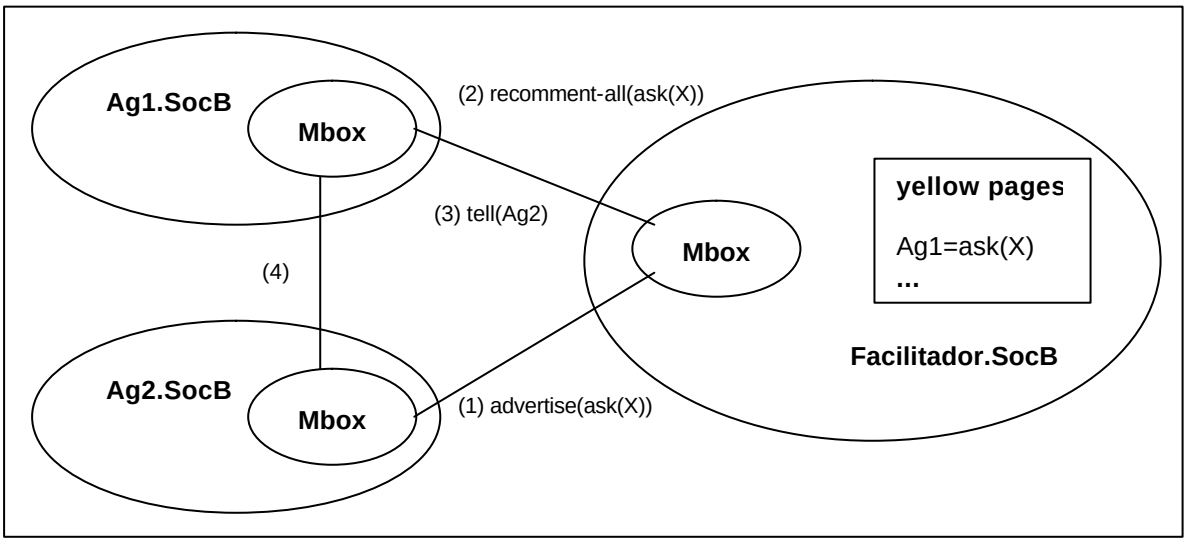


Fonte: adaptado de Hübner (2001)

No caso da fig. 5, o agente *Ag2* da sociedade *SocA* deseja comunicar-se com o agente *Ag1* que pertence à mesma sociedade e que o *Ag2* conhece o nome. Inicialmente, o *MBox* do *Ag2* precisa saber a localização do *Ag1*, portanto, pergunta ao facilitador da sociedade *SocA* tal *localização* (marcação(1) na fig. 5), que lhe responde "*host a*" (marcação (2)). Tendo a localização, o *Ag2* inicia a comunicação com o agente *Ag1* (marcação (3)). O agente que está no *host a* pertence a duas sociedades (*SocA* e *SocB*), tendo um *Mbox* para cada sociedade.

Para serem conhecidos melhor na sociedade os agentes podem anunciar suas habilidades ao facilitador de forma análoga ao serviço de páginas amarelas. Assim, quando um agente precisa de um serviço e não conhece o nome de um agente capaz de realizá-lo, pode requisitar ao facilitador uma lista de agentes com tal habilidade, conforme ilustrado na fig. 6. Na verdade, este é apenas um modelo disponível no SACI para a apresentação, sendo que outras formas podem ser utilizadas, como o *broadcasting* das habilidades (Hübner, 2001).

FIGURA 6 - SERVIÇO DE PÁGINAS AMARELAS



Fonte: adaptado de Hübner (2001)

No caso da fig. 6 tendo o agente *Ag*2 anunciado a habilidade *x* (*seta (1)*), se o agente *Ag*1 pedir ao facilitador uma lista de agentes com a habilidade *x* (*seta (2)*), receberá *Ag*2 como resposta (*seta (3)*) e poderá iniciar a comunicação com este agente (*seta (4)*).

Um exemplo de mensagem do agente *Ag*2 anunciando uma habilidade ao facilitador é ilustrado no quadro 2.

QUADRO 2 - ANÚNCIO DE HABILIDADE

```
(advertise
      :receiver    Facilitado r
      :sender      Ag2
      :language    KQML
      :ontology    yp
      :content     (ask-one
                           :receiver   Ag2
                           :language   alg
                           :ontology   math
                           :content    "X+Y"))
```

Fonte: adaptado de Hübner (2001)

Um exemplo de mensagem do agente Ag1 solicitando ao facilitador uma lista de agentes com habilidades é ilustrado na quadro 3, e a resposta do facilitador é ilustrado no quadro 4.

QUADRO 3 - SOLICITAÇÃO DE LISTA DE AGENTES COM HABILIDADES

```
(recomend -all
      :receiver      Facilitado r
      :sender        Ag1
      :reply-with    id1
      :language      KQML
      :ontology      yp
      :content       (ask-one
                           :language   alg
                           :ontology   math
                           :content    "X+Y"))
```

Fonte: adaptado de Hübner (2001)

QUADRO 4 - RESPOSTA DO FACILITADOR COM A LISTA DE AGENTES

```
(tell
      :receiver      Ag1
      :sender        Facilitador
      :in-reply-to   id1
      :language      KQML
      :ontology      yp
      :content       (Ag2))
```

Fonte: adaptado de Hübner (2001)

Além da comunicação entre os agentes o SACI possui mais um serviço que é a criação de agentes (lançamento), denominado *launcher*. Para serem criados os agentes através do *launcher* do SACI devem ser informados os seguintes parâmetros:

a) class: classe que foi implementada o agente a ser criado;
b) society: sociedade a qual o agente a ser criado irá pertence;
c) name: nome que será identificado o agente na sociedade;
d) creationMethod: método de como será criado o agente, podendo ser o agente um *processo* ou uma *thread*;
e) host: máquina onde será criado o agente;
f) args: lista de argumentos que podem ser passados aos agentes na hora da criação para futura utilização.

3.2.2 ÁRVORE DE DIRETÓRIOS DO SACI

O SACI possui uma árvore de diretório exposto na seguinte maneira:

a) saci/bin/ :arquivos responsáveis pela inicialização do SACI;
b) saci/doc/ : documentação do SACI;
c) saci/lib/ : bibliotecas utilizadas pelo SACI;
d) saci/samples/ : exemplos de programação de agentes;
e) saci/ulib/ : diretório onde deverão ficar as classes que foram implementados os agentes.

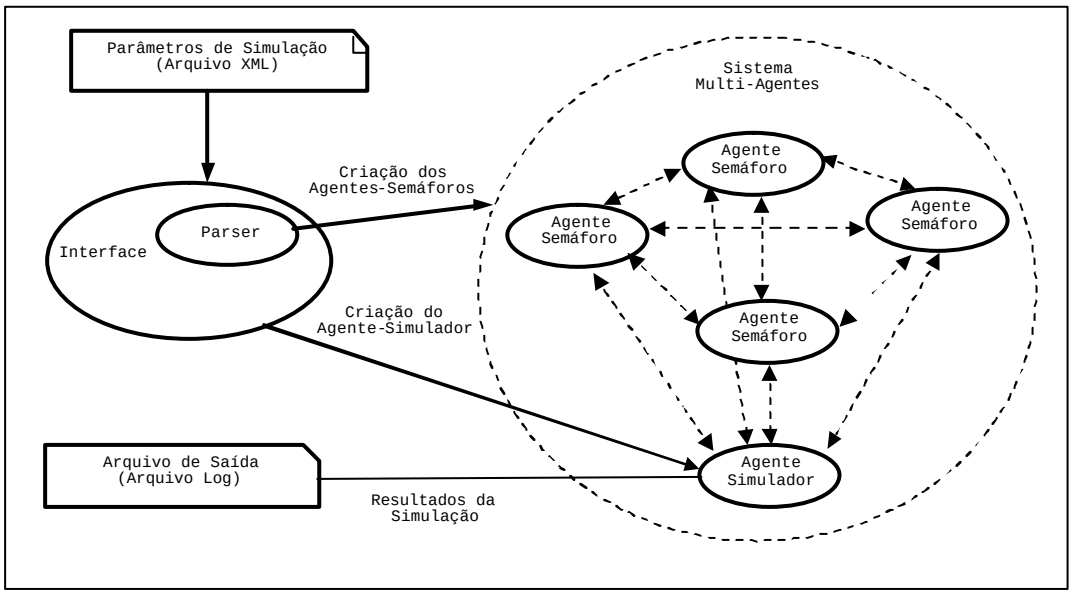
4 ESPECIFICAÇÃO DO SISTEMA MULTI-AGENTES PARA O SISTEMA SEMAFÓRICO

O sistema tem como principal foco o gerenciamento e controle de tráfego nas vias urbanas, tomando como objeto de estudo o direito de passagem dos veículos em cruzamentos, comumente conhecido através do acionamento da luz verde do semáforo. Sendo estudados os cruzamentos em nível com vias de circulação de mão dupla, onde a gerência e controle dos semáforos destes cruzamentos podem afetar em várias vezes o sincronismo de tempo nos semáforos em série.

Para a realização deste gerenciamento e controle, o sistema é composto por um conjunto de agentes distribuídos, conforme ilustra fig. 7, onde cada semáforo do mundo real será representado por um agente denominado no sistema **agente-semáforo**, que se comunicará com os semáforos do cruzamento ao qual pertence, para que tomem decisões e num todo tentem atingir o objetivo de gerenciamento e controle de tráfego.

Dados as dificuldades operacionais de aplicação real do sistema no trânsito de uma cidade, é necessário o desenvolvimento de um simulador para representar o mundo real e ajudar na realização da tarefa de gerenciamento e controle de tráfego. Este simulador é implementado em um agente denominado **agente-simulador**, responsável por simulações das situações do mundo real das vias urbanas e, servindo assim de fonte de dados para que os agentes-semáforos apliquem o mecanismo de decisão.

FIGURA 7 - ESTRUTURA DO SISTEMA



Utilizando o SACI, são criados os agentes-semáforos e o agente-simulador através de uma interface, conforme ilustra fig. 7, que utiliza como entrada um arquivo contendo os parâmetros de simulação no formato *eXtensible Markup Language* (XML). Neste arquivo encontram-se mapeados os semáforos existentes na malha viária simulada. Nesta interface é implementado um *parser* que realiza a leitura dos elementos do arquivo XML, em seguida criando os agentes-semáforos mapeados com seus atributos para que através dos protocolos de comunicação, estes comuniquem entre si trocando informações e decisões.

Os agentes-semáforos negociarão entre si o direito de passagem no cruzamento, comumente conhecido como a fase verde de um semáforo, sendo este definido como o **objeto de negociação**. No entanto assim que um agente-semáforo analisa as condições de sua via e verifica que necessita do direito de passagem, o mesmo realiza um pedido do objeto de negociação ao agente-semáforo que o possui e aguarda uma resposta. Observando que em um cruzamento, somente um único agente-semáforo possuirá o direito de passagem, este mesmo agente é o responsável pela decisão da liberação do objeto de negociação ao agente-semáforo solicitante.

Assim que um agente-semáforo recebe um pedido do objeto de negociação, o mesmo analisa as condições de sua via e decide por passar ou não o objeto de negociação. No decorrer da simulação e da decorrente passagem do objeto de negociação entre os agentes-semáforos, ocorre a alternância das fases das luzes nos semáforos.

A alternância entre o direito de passagem de um agente-semáforo a outro é sempre comunicado ao agente-simulador, servindo assim como resultado da simulação do mundo. Estes resultados são gravados pelo agente-simulador em um arquivo para que seja possível posteriormente a análise e estudos sobre os resultados obtidos.

No entanto algumas limitações sobre trânsito são impostas na realização deste estudo, como: não são considerados os cruzamentos em nível com faixas específicas para conversão, vias com faixas de estacionamento, vias com artérias e os tempos semafóricos para pedestres.

O desenvolvimento do sistema por sua vez limita-se na geração de um arquivo de saída contendo os resultados obtidos na simulação, não sendo implementadas interfaces gráficas para simular o mundo real como as ruas, automóveis e semáforos.

4.1 ESPECIFICAÇÃO DOS PROTOCOLOS DE COMUNICAÇÃO

O protocolo de comunicação estabelece como será a troca de mensagens entre os agentes-semáforos e, entre agentes-semáforos e agente-simulador.

Nestes protocolos de comunicação será utiliza uma linguagem definida como *Linguagem de Trânsito* (LT), e uma ontologia definida como *transito.* Esta linguagem e ontologia são padrão para todos os protocolos utilizados no sistema.

4.1.1 PROTOCOLOS DO AGENTE-SIMULADOR

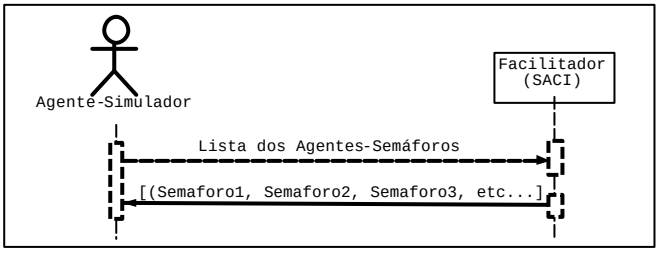
Este protocolo é composto por mensagens que são de origem do agente-simulador, e possui duas categorias: protocolo de solicitação da listagem dos agentes-semáforos e protocolo de aviso da entrada de carros na via.

4.1.1.1 PROTOCOLO DE SOLICITAÇÃO DA LISTAGEM DOS AGENTES-SEMÁFOROS

Este protocolo tem como objetivo a obtenção da listagem dos agentes-semáforos existentes que possuem as habilidades sobre trânsito, sendo necessária para posteriormente o agente-simulador aplicar o protocolo avisando aos agentes-semáforos sobre a entrada de carros em suas vias.

Para isto o agente-simulador faz uma requisição ao facilitador do SACI solicitando a listagem dos agentes-semáforos que anunciaram nas páginas amarelas a habilidade de receber mensagens com o conteúdo *EntrouCarro* (fig. 8).

FIGURA 8 - PROTOCOLO DA SOLICITAÇÃO DA LISTAGEM DOS AGENTES-SEMÁFOROS

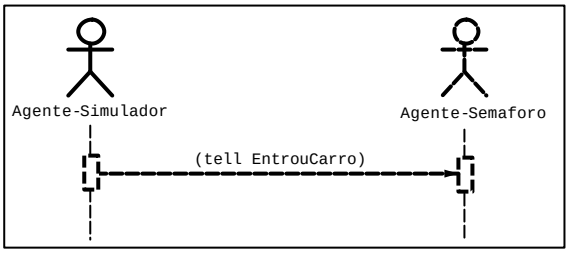


4.1.1.2 PROTOCOLO DE AVISO DA ENTRADA DE CARRO NA VIA

O objetivo deste protocolo é realizar a simulação da entrada de carro na via de cada agente-semáforo e se realiza entre o agente-simulador e os agentes-semáforos.

Após a obtenção da listagem dos agentes-semáforos que possuem a habilidade de serem avisados sobre a entrada de carros na via, o agente-simulador é configurado para em um determinado intervalo de tempo enviar uma mensagem avisando a cada agente-semáforo desta listagem sobre a entrada de carros em sua via (fig. 9).

FIGURA 9 - PROTOCOLO DA ENTRADA DE CARROS NA VIA



A mensagem sobre o aviso de carro utiliza a performativa *tell*, indentificando que é apenas um aviso, não necessitando o envio de resposta. Também consta na mensagem a quantidade de carros que entram na via de cada agente-semáforo, conforme exemplificado no quadro 5, sendo informada através da palavra chave *qtde*.

QUADRO 5 - EXEMPLO DE MENSAGEM DA ENTRADA DE CARROS

```
(tell
      :language      LT
      :ontology      transito
      :receiver      Semaforo4
      :sender        Semaforo1
      :content       EntrouCarro
      :qtde          1)
```

4.1.2 PROTOCOLO DO AGENTE-SEMÁFORO

Este protocolo é composto por mensagens que são de origem do agente-semáforo, podendo ser enviadas a outros agentes-semáforos no caso dos protocolos de solicitação do objeto de negociação e aviso do recebimento do objeto de negociação. Também podendo ser

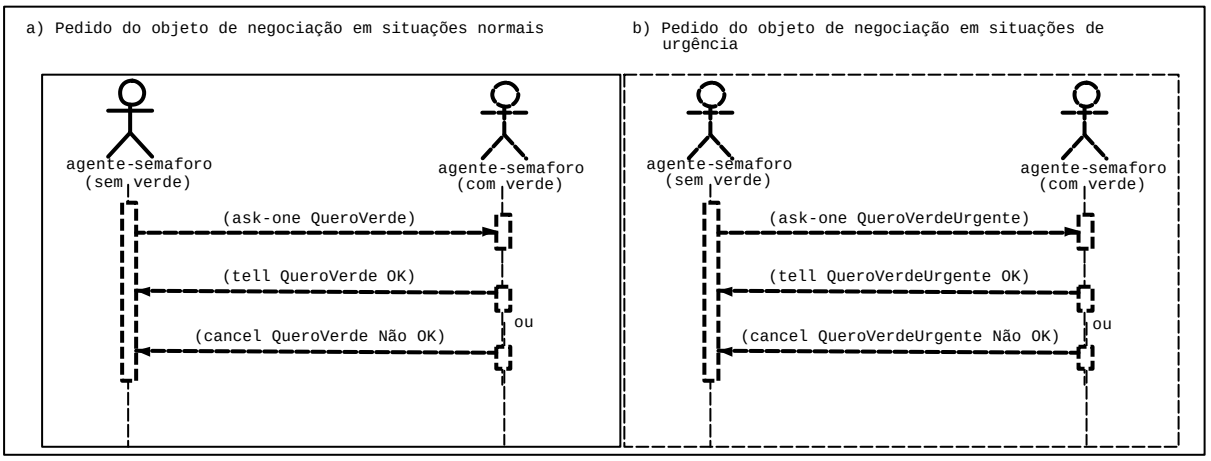
enviadas ao agente-simulador, através do protocolo de aviso ao simulador do estado do semáforo.

4.1.2.1 PROTOCOLO DE SOLICITAÇÃO DO OBJETO DE NEGOCIAÇÃO

O objetivo deste protocolo é a passagem do objeto de negociação entre os agentes-semáforos.

Assim que um agente-semáforo analisa as condições de sua via e verifica a necessidade do objeto de negociação, o mesmo utiliza-se deste protocolo pra realizar o pedido, enviando uma mensagem com a performativa *ask-one* solicitando o objeto de negociação ao agente-semáforo que o possui, sendo que o mesmo deve receber uma resposta que pode conter em sua performativa *tell* avisando que o pedido foi atendido, ou *cancel* avisando que o pedido não foi atendido (fig. 10).

FIGURA 10 - PROTOCOLO DE SOLICITAÇÃO DO OBJETO DE NEGOCIAÇÃO



Este protocolo de solicitação do objeto de negociação pode ser utilizado sob duas situações:

a) situações normais: a solicitação do objeto de negociação em situações normais ocorre quando o agente-semáforo solicitante analisa as condições de sua via e verifica que está ao ponto de realizar o pedido normal;
b) situações de urgência: a solicitação do objeto de negociação em situações de urgência ocorre quando o agente-semáforo solicitante analisa as condições de sua via e verifica que ela está comprometida, ultrapassando os limites permitidos.

Um exemplo da mensagem utilizando o protocolo de solicitando o objeto de negociação pode ser observada no quadro 6.

QUADRO 6 - EXEMPLO DE MENSAGENS SOLICITANDO O OBJETO DE NEGOCIAÇÃO

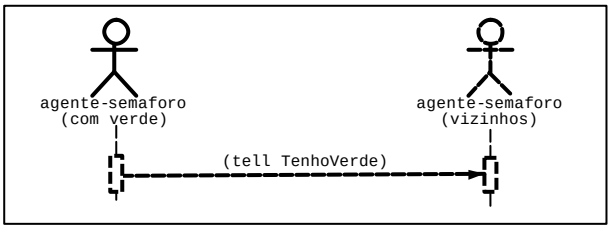
MENSAGEM EM CONDIÇÕES NORMAIS		MENSAGEM EM CONDIÇÕES DE URGÊNCIA	
(ask-one		(ask-one	
:language	LT	:language	LT
:ontology	transito	:ontology	transito
:receiver	Semaforo4	:receiver	Semaforo4
:sender	Semaforo1	:sender	Semaforo1
:content	QueroVerde)	:content	QueroVerdeUrgente)

4.1.2.2 PROTOCOLO DE AVISO DE RECEBIMENTO DO OBJETO DE NEGOCIAÇÃO

Este protocolo tem como objetivo avisar aos agentes-semáforos do cruzamento sobre o recebimento do objeto de negociação por algum agente-semáforo.

No momento em que um agente-semáforo recebe o objeto de negociação, ele avisa aos agentes-semáforos do seu cruzamento sobre o recebimento do objeto de negociação, para que na próxima vez em que eles precisarem realizar o pedido do objeto de negociação terem qual é o atual proprietário, conforme ilustra fig. 11.

FIGURA 11 - PROTOCOLO DE RECEBIMENTO DO OBJETO DE NEGOCIAÇÃO



Um exemplo da mensagem utilizando este protocolo pode ser visualizada no quadro 7.

QUADRO 7 - EXEMPLO DE MENSAGEM AVISANDO SOBRE O RECEBIMENTO DO OBJETO DE NEGOCIAÇÃO

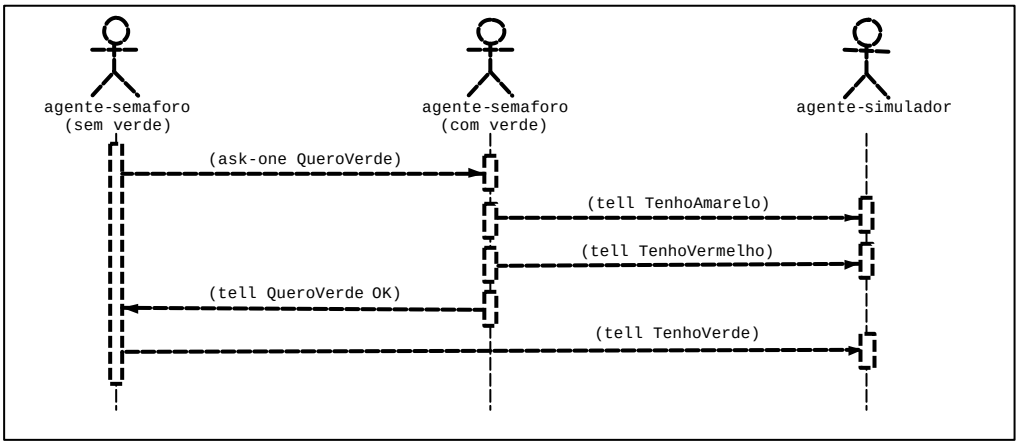
```
(tell
      :language      LT
      :ontology      transito
      :receiver      Semaforo4
      :sender        Semaforo1
      :content       TenhoVerde)
```

4.1.2.3 PROTOCOLO DE AVISO AO SIMULADOR DO ESTADO DO SEMÁFORO

A utilização deste protocolo objetiva a utilização dos tempos semafóricos de segurança (amarelo e vermelho de segurança), avisando ao agente-simulador sobre o estado atual do agente-semáforo.

Quando um agente-semáforo recebe o pedido do objeto de negociação e resolve por liberar ele ao solicitante, antes da liberação ele deve esperar pelos tempos de amarelo e o vermelho de segurança avisando ao agente-simulador através de uma mensagem com a performativa *tell* sobre o início de cada fase semafórica, conforme ilustra fig. 12.

FIGURA 12 - PROTOCOLO DO AVISO AO SIMULADOR DO ESTADO DO SEMÁFORO



Após passados os tempos semafóricos de segurança, o agente-simulador proprietário do objeto de negociação o passa ao agente solicitante, que por sua vez avisa ao agente-

simulador através de uma mensagem com a performativa *tell* sobre o recebimento do objeto de negociação.

Um exemplo da mensagem utilizando este protocolo avisando ao simulador sobre o início da fase semafórica é exemplificado no quadro 8, quadro 9 e quadro 10.

QUADRO 8 - EXEMPLO DE MENSAGEM AVISANDO AO SIMULADOR DO INÍCIO DA FASE AMARELA

```
(tell
      :language      LT
      :ontology      transito
      :receiver      Simulador
      :sender        Semaforo1
      :content       TenhoAmarelo)
```

QUADRO 9 - EXEMPLO DE MENSAGEM AVISANDO AO SIMULADOR DO INÍCIO DA FASE VERMELHA

```
(tell
      :language      LT
      :ontology      transito
      :receiver      Simulador
      :sender        Semaforo1
      :content       TenhoVermelho)
```

QUADRO 10 - EXEMPLO DE MENSAGEM AVISANDO AO SIMULADOR DO INÍCIO DA FASE VERDE

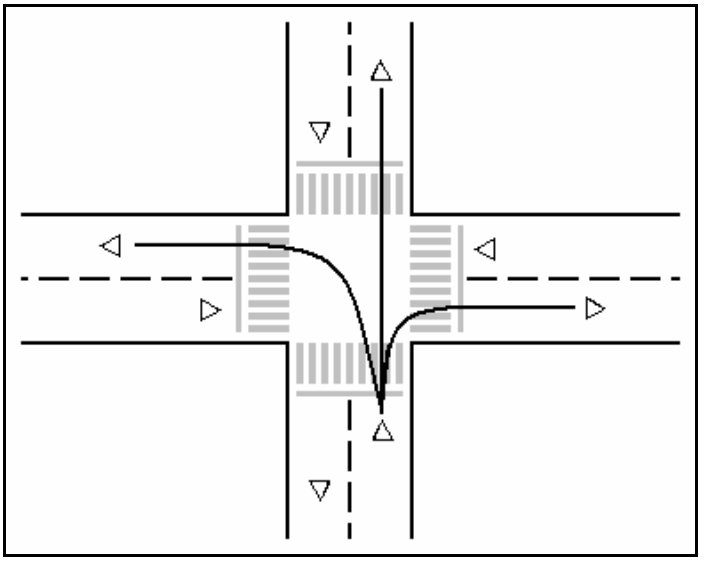
```
(tell
      :language      LT
      :ontology      transito
      :receiver      Simulador
      :sender        Semaforo1
      :content       TenhoVerde)
```

4.2 AGENTE-SEMÁFORO

Nos cruzamentos em nível e com fluxo interrompido o controle e gerenciamento do tráfego urbano são realizados na maioria das vezes pela alternância do direito de passagem dos veículos em suas vias, sendo esta a função dos semáforos de regulamentação, assim também sendo a função dos agentes-semáforos no sistema.

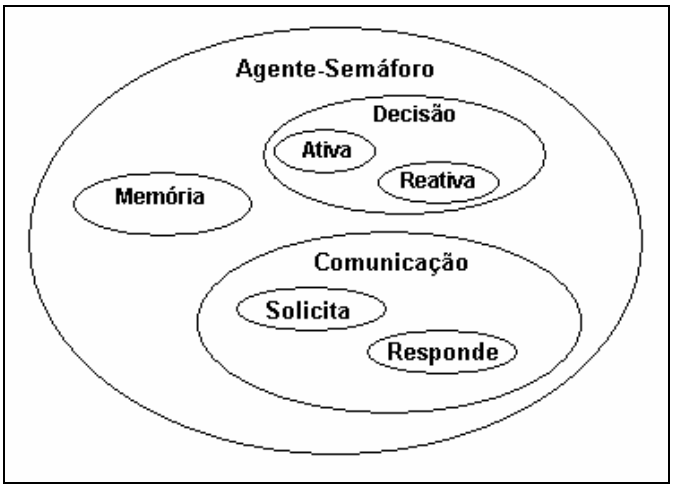
Para os agentes-semáforos, serão considerados no sistema, cruzamentos em nível e vias de sentido duplo. Tendo uma via o direito de passagem o motorista pode escolher em convergir para alguma via perpendicular ou seguir em linha reta, conforme ilustra fig. 13.

FIGURA 13-CONVERSÃO EM CRUZAMENTO



O agente-semáforo é composto, conforme ilustra fig. 14, por três módulos principais: módulo de memória, módulo de decisão e módulo de comunicação. O módulo de memória é responsável pelo armazenamento do objeto de negociação e dos dados necessários para o agente-semáforo tomar decisões no sistema. O módulo de comunicação é composto pelos protocolos de comunicação e responsável por realizar a comunicação entre os agentes-semáforos e agente-simulador.

FIGURA 14 - ESTRUTURA DO AGENTE-SEMÁFORO



4.2.1 MEMÓRIA

O módulo de memória de um agente-semáforo abrange informações sobre todo o cruzamento ao qual ele pertence, informações pré-estabelecidas conforme revisão bibliográfica, informações alimentadas na criação do agente e algumas informações que podem sofrer alterações no decorrer da execução do agente.

As informações do módulo de memória pré-estabelecidas que não sofrem alterações na execução do agente são valores padrões conforme Stern (1969):

a) tempo de atraso: tempo de atraso do primeiro veículo imediatamente após a abertura do sinal (3 segundos);
b) intervalo de arranque: intervalo de tempo entre o arranque de dois veículos sucessivos (1,5 segundos);
c) distância de embalagem: distância percorrida durante o período de embalagem (60 metros);
d) comprimento do veículo: comprimento médio de um veículo (4 metros).

As informações sobre o cruzamento que o semáforo pertence, ilustrado na fig. 15, alimentadas na criação do agente-semáforo e que não sofrem alterações na execução do agente são:

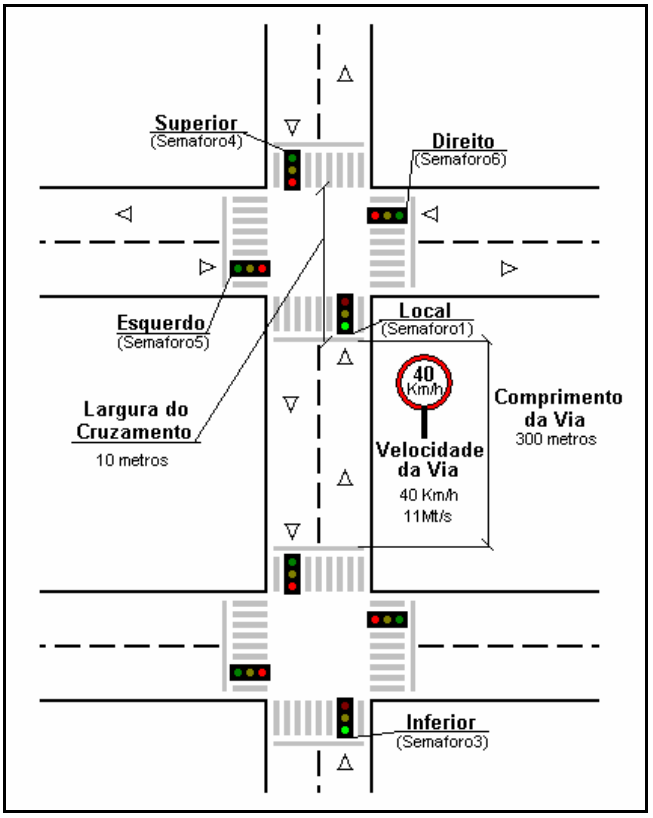
a) semáforo local: identificação do agente-semáforo local (os semáforos são identificados no sistema pela junção do nome “Semaforo” e por uma identificação única, como por exemplo Semaforo1);
b) semáforo superior: identificação do agente-semáforo que se localiza acima do cruzamento;
c) semáforo esquerdo: identificação do agente-semáforo que se localiza a esquerda;
d) semáforo direito: identificação do agente-semáforo que se localiza a direita;
e) semáforo inferior: identificação do agente-semáforo que se localiza abaixo, na maioria das vezes agente-semáforo de um outro cruzamento;
f) largura do cruzamento: largura do cruzamento incluindo a faixa de pedestres (expressa em metros);
g) velocidade da via: velocidade permitida para a via (expressa em metros/segundos);
h) comprimento da via: comprimento da via até o início da faixa de pedestre (expressa em metros).

As informações alimentadas na criação do agente-semáforo e que podem sofrer alterações na execução do agente são:

a) semáforo verde: semáforo que inicia o funcionamento com o direito de passagem, em outras palavras, semáforo que inicia o funcionamento com o objeto de negociação;
b) quantidade de carros: quantidade de carros que entrou em sua via;
c) tempo de escoamento: tempo de verde necessário para escoamento da fila existente em sua via (expressa em segundos);
d) tempo de paciência: momento inicial em que o agente-semáforo passou o verde para outro semáforo (expressa em segundos);
e) início do verde: momento inicial em que o agente-semáforo ganhou o verde.

O módulo de memória é alimentado na criação do agente-semáforo e por mensagens enviadas pelo simulador, que informam ao agente-semáforo a entrada de carro em sua via. Além disso, algumas decisões tomadas pelo próprio semáforo alteram algumas destas informações. Um exemplo de como obter estas informações é ilustrado na fig. 15.

FIGURA 15 - VARIÁVEIS DO MÓDULO DE MEMÓRIA



4.2.2 DECISÃO

São implementados no agente-semáforo, algoritmos de decisão, que agem sobre eles fazendo com que tomem a melhor decisão em relação a enviar uma mensagem (decisão ativa) (fig. 16), ou em relação à ação a ser realizada em função de uma mensagem recebida (decisão reativa) (fig. 17).

FIGURA 16 - DECISÃO ATIVA

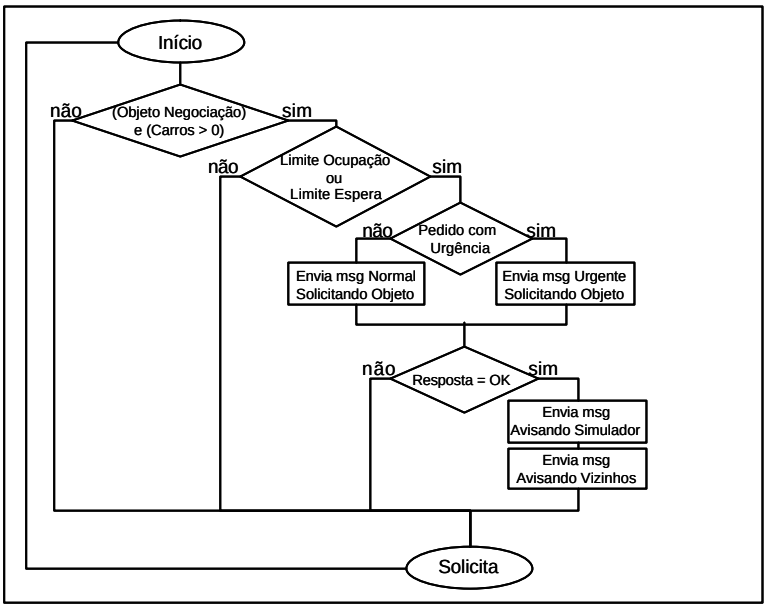
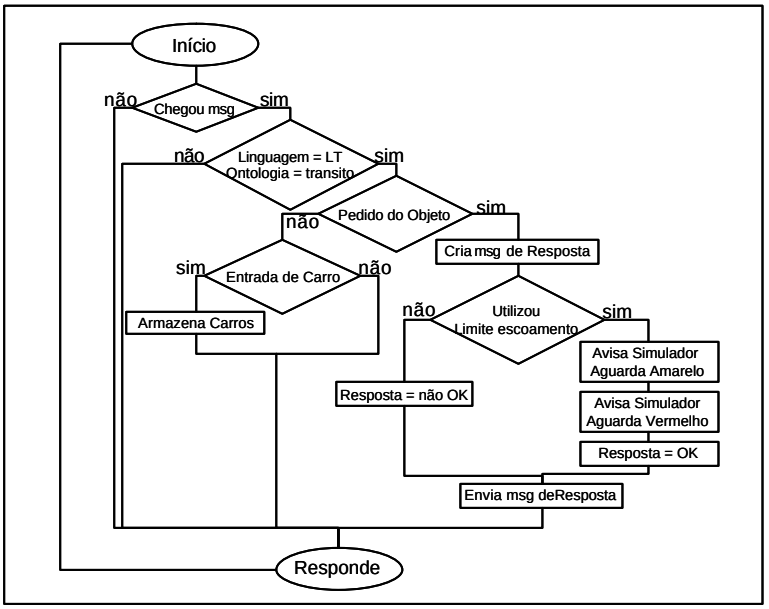


FIGURA 17 - DECISÃO REATIVA



Para que os agentes-semáforos possam tomar estas decisões, foram criados fatores de paciência e fatores de ocupação, que auxiliam no cálculo da obtenção do valor de ocupação da via e da paciência do motorista, podendo ser configurados para cada sistema.

Para o agente-semáforo tomar a decisão devem ser analisadas as condições da via em relação aos seguintes fatores:

a) fator de ocupação: responsável por não permitir que a utilização da via seja esgotada, ou seja, este fator é o que determina em que percentual de ocupação da via o agente-semáforo deve solicitar o verde;
b) fator de paciência: responsável por tentar criar um trânsito tranqüilo, fazendo com que o agente-semáforo não permaneça com o vermelho por muito tempo, não permitindo que chegue ao ponto deste tempo torna-se um quantitativo para o nível de irritação do motorista.
c) fator de ocupação urgência: possui a mesma função do fator de ocupação, mas com a diferença que deverá solicitar o verde com urgência;
d) fator de paciência urgência: possui a mesma função do fator de paciência, com a diferença que deverá solicitar o verde com urgência.

Assim através da comparação da situação da via com os fatores de ocupação e paciência, conforme especificado no quadro 11 (linha 3), o agente-semáforo tomará a decisão sobre enviar uma mensagem solicitando o objeto de negociação e deverá aguardar a resposta (decisão ativa), caso o agente-semáforo ganhe o objeto de negociação é realizado o cálculo do tempo de escoamento necessário, utilizando a equação IX (quadro 11, linha 4). De outro modo os fatores de urgência determinam que o agente deverá fazer o pedido do objeto de negociação com urgência, fazendo com que o agente-semáforo que possui o objeto de negociação perceba que a solicitação foi realizada e as condições da via do solicitante estão comprometidas.

QUADRO 11 - DECISÃO ATIVA

```
1 INICIO
2   Se (Não tiver o objeto de negociação) e (Quantidade de Carros na Via > 0) então
3     Se (% Ocupação da Via > Fator de Ocupação) ou
        (Tempo de Espera pelo verde > Fator de Paciência) então
4        Tempo de escoamento = Tempo de atraso +
                               (Quantidade de Carros - 1) * Intervalo de arranque +
                               (Distância de Embalagem /(Velocidade da Via / 2)) +
                               (((Quantidade Carros * (Comprimento do Veículo + 0,5)) +
                                Distância de Embalagem)/Velocidade da via);
5       Envia Mensagem Pedindo Verde e aguarda Resposta (ask-one :language LT
                                      :ontology transito :sender SemaforoLocal
                                      :receiver SemaforoVerde :content QueroVerde);
6       Se (Resposta = tell) então
7         //Ganhou o objeto de negociação
8         Verde = Semaforo Local;
9         Inicio do Verde = Hora atual;
10        Envia Mensagem Avisando Simulador (ask-one :language LT   :ontology transito
                                            :sender SemaforoLocal :receiver Simulador
                                            :content TenhoVerde)");
11        Envia Mensagem Avisando Vizinhos(ask-one :language LT   :ontology transito
                                          :sender SemaforoLocal :content TenhoVerde)");
12      Fim Se;
13      Se (Resposta = cancel) então
14        //Não Ganhou o Objeto de negociação
15      Fim Se;
16    Fim Se;
17  Fim Se;
18FIM;
```

O agente-semáforo é capacitado também para, ao receber uma mensagem de solicitação do objeto de negociação, decidir sobre que resposta deverá enviar (decisão reativa), conforme especificado no quadro 12. Para que o agente-semáforo responda a este pedido, foi criado um fator de escoamento, o qual é configurado para permitir o tempo mínimo de escoamento que o agente-semáforo poderá utilizar.

Caso o semáforo resolva passar o direito do objeto de negociação, deverá aguardar os tempos de segurança (amarelo e vermelho de segurança) obtidos através da utilização das equações VI, e VII, e especificado no quadro 12 através das linhas 14 e 15.

QUADRO 12 - DECISÃO REATIVA

```
1 INICIO
2   Aguarda o recebimento de mensagem;
3   Se (Chegou Mensagem) então
4     Se (Linguagem da mensagem = LT) e (Ontologia = transito) então
5       É uma mensagem do sistema;
6       Se (Performativa da mensagem recebida= Ask-One) então
7         Cria mensagem de resposta(tell :language LT :ontology transito
                                    :receiver sender da mensagem recebida);
8         Se (Conteúdo da mensagem recebida = Pedido do Verde) então
9           Se (Tempo de utilização do verde < Fator de Escoamento) então
10            Não Pode enviar o verde;
11            Conteúdo da mensagem de resposta = cancel;
12            Envia Resposta;
13          Senão
14            Aguarda o tempo do amarelo;
15            Aguarda o tempo do vermelho de segurança;
16            Pode enviar o verde;
17            Inicio do Tempo de espera pelo verde = Hora atual;
18            Verde = Semáforo sender da mensagem recebida;
19            Envia Resposta;
20          Fim Se;
21        Fim Se;
22      Fim Se;
23      Se (performativa da mensagem recebida = tell) então
24        Se (Conteúdo da mensagem recebida = Entrou Carro) então
25          Soma a quantidade de carros existente no campo “qtde” da mensagem recebida;
26        Fim Se;
27      Fim Se;
28    Fim Se;
29  Fim Se;
30FIM;
```

4.3 AGENTE-SIMULADOR

Com o objetivo de tentar simular situações do mundo real, foi desenvolvido um simulador, o qual é iniciado logo após a criação dos agentes-semáforos. Após isto o agente-simulador utiliza o protocolo de solicitação da listagem dos agentes-semáforos para receber a listagem dos agentes-semáforos. Conhecendo todos os agentes-semáforos existentes, é possível através do agente-simulador informar a intensidade do tráfego na via de cada agente-semáforo, assim começa o processo de envio de mensagem para os agentes-semáforos avisando sobre a entrada de carro em sua via, utilizando o protocolo de aviso da entrada de carro na via.

Após o agente-simulador ter sido iniciado, e os agentes-semáforos começarem a tomar decisões em solicitar ou passar o objeto de negociação, o agente-simulador é informado sobre o estado de cada agente, em outras palavras, é informado sobre o momento em que o agente-semáforo está com o amarelo, com o vermelho e com o verde, sendo estas informações

enviadas pelo protocolo de aviso ao simulador do estado do semáforo e armazenadas em um arquivo de *log* para posterior análise e estudo.

5 IMPLEMENTAÇÃO

O sistema foi implementado em Java, utilizando como ferramenta para a criação e comunicação entre os agentes o SACI.O desenvolvimento das classes dos agentes é realizado por meio da herança da classe *Agent*, contida nas bibliotecas do SACI.

5.1 AGENTE-SEMÁFORO

Para a implementação do agente-semáforo foi desenvolvida a classe *Semaforo* com o módulo de memória sendo os atributos de cada agente, conforme ilustra quadro 13.

O agente-semáforo é iniciado através da chamada do método *initAg()* (quadro 13, linha 39). Por meio deste método é realizado o anúncio de suas habilidades nas páginas amarelas (quadro 14, linhas 4 a 9) e, em seguida são alimentados os dados sobre o cruzamento, gravando também o início do tempo de paciência e se caso o semáforo possuir o objeto de negociação grava o início do tempo em que possui este objeto conforme ilustrado no quadro 14, linha 25.

Após iniciado o agente-semáforo é colocado em execução através da chamada do método *run()* (quadro 13, linha 40). Este método, por sua vez faz chamada a dois outros métodos: o *solicita()* que é responsável pela decisão do envio de mensagem pedindo o objeto de negociação e o método *responde()* que é responsável pelo tratamento do recebimento de mensagens de solicitação e de outras mensagens.

O método *solicita()* é o responsável por consultar o módulo de decisão e verificar se o agente deve tomar a iniciativa de solicitar o objeto de negociação e por gravar o tempo de escoamento necessário através do resultado do cálculo realizado pela chamada de método *escoamento()* que faz uso da equação IX, conforme ilustra quadro 15, linha 21.

No método *solicita()*, o módulo de decisão pode ser acionado através da chamada de método *possoPedirVerde()*, que pode levar como parâmetro *true* indicando que a verificação deve considerar como um pedido de urgência, ou *false*, onde deve considerar como um pedido normal.

QUADRO 13 - IMPLEMENTAÇÃO DA CLASSE SEMAFORO

```
1 import saci.*;
2 import java.util.*;
3 import java.text.*;
4
5 public class Semaforo extends Agent {
6   private String        local,               //Identificacao do semaforo local
7                         superior,            //Identificacao do semaforo superior
8                         esquerdo,            //Identificacao do semaforo esquerdo
9                         direito,             //Identificacao do semaforo direito
10                        inferior,            //Identificacao do semaforo inferior
11                        verde;               //Identificacao do semaforo que tem o verde
12
13  private int           cruzamento,          //Comprimento do cruzamento
14                        velocidade,          //Velocidade do Regime (Velocidade da via)
15                        comprimento,         /*Comprimento da Via em Relacao
                                                 o Semaforo Local e o inferior*/
16                        qtdecarros,          //Quantidade de Carros na via
17                        tescoamento;         /*Tempo de verde necessario
                                                 para escoamento da fila*/
18
19  private Date          tpaciencia,           /*Inicio do tempo de paciencia
                                                  esperando pelo verde
                                                  40=Ideal; 60=Aceitavel;
                                                  80=Irritante; 100=Insuportável*/
21                        inicioverde;          //marca a hora que ganhou o verde
22
23  private static double atraso        = 3,    /*Tempo de atraso do primeiro veiculo
                                                  imediatamente apos abertura do sinal */
24                        arranque      = 1.5,  /*Intervalo de tempo entre o arranque de
                                                  dois veiculos sucessivos*/
25                        distembalagem = 60,   /*Distancia percorrida durante o periodo
                                                  de embalagem*/
26                        compveiculo   = 4;    //Comprimento medio de um veiculo
27
28  private static int    ftpaciencia  = 90,    //% permitido fator paciencia
29                        ftocupacao   = 60,    //% permitido fator ocupacao
30                        ftupaciencia = 110,   //% permitido fator paciencia urgencia
31                        ftuocupacao  = 90,    //% permitido para fator ocupacao urgência
32                        fescoamento  = 50,    //% permitido para fator escoamento
33                        fescoamento  = 50,    //% permitido fator escoamento
34                        fuescoamento = 30;    //% permitido fator escoamento urgencia
35
36  public void main (String[] args) {
37    Agent a = new Semaforo();
38    if(a.enterSoc("SocSem")){
39      a.initAg(null);
40      a.run();
41    }
42  }
43.
44.
45.
```

O método *possoPedirVerde()* é a implementação do módulo de decisão que analisa os fatores de paciência e ocupação. Conforme ilustra quadro 16, linhas 3 e 14, inicialmente um agente-semáforo só ira pedir o objeto de negociação se possuir pelo menos um carro em sua via e se não possuir o objeto de negociação, assim impedindo que um agente-semáforo que não possui carros em sua via ganhe o objeto de negociação.

QUADRO 14 - MÉTODO INITAG()

```
1  public void initAg(String[] args) {
2    try {
3      //Anúncio das habilidades
4      mbox.advertise("ask-one","LT","transito","TenhoVerde");
5      mbox.advertise("ask-one","LT","transito","TenhoAmarelo");
6      mbox.advertise("ask-one","LT","transito","TenhoVermelho");
7      mbox.advertise("ask-one","LT","transito","QueroVerde");
8      mbox.advertise("ask-one","LT","transito","QueroVerdeUrgente");
9      mbox.advertise("tell","LT","transito","EntrouCarro");
10
11     System.out.println("Inciando Semaforo");
12
13     this.local        = "Semaforo" + args[0];
14     this.superior     = "Semaforo" + args[1];
15     this.esquerdo     = "Semaforo" + args[2];
16     this.direito      = "Semaforo" + args[3];
17     this.inferior     = "Semaforo" + args[4];
18     this.verde        = "Semaforo" + args[8];
19     this.cruzamento   = Integer.parseInt(args[5]);
20     this.velocidade   = Integer.parseInt(args[6]);
21     this.comprimento  = Integer.parseInt(args[7]);
22     this.qtdecarros   = 0;
23     this.tpaciencia   = new Date();  //inicializar a variavel com o tempo atual
24     if (this.verde.equals(this.local)) {
25       this.inicioverde = new Date();
26     }
27
28     System.out.println("Semaforo" + mbox.getName() + " Iniciado com Sucesso! \n");
29
30   } catch (Exception e){
31     System.err.println("Erro Iniciando Semaforo "  + e );
32   }
33 }
34
35 public void run() {
36   while (true) {
37     solicita();
38     responde();
39   }
40 }
```

Em seguida é realizada uma verificação da ocupação da via e comparada com o fator permitido (quadro 16, linhas 5 e 16). Esta verificação tem os mesmos princípios tanto para o pedido normal, como para o pedido com urgência, alterando somente a comparação com os fatores permitidos. Juntamente deve ser analisado o tempo em que o motorista está aguardando pela abertura do sinal. Esta análise é realizada através do método *tempoEspera()*, e posteriormente comparada com o fator permitido.

A liberação do pedido do objeto de negociação só deve acontecer, caso a comparação dos fatores de ocupação e dos fatores de paciência, seja ultrapassada pelas situações da via. Assim que o agente-semáforo recebe a resposta, o mesmo só irá realizar alguma ação caso a resposta for positiva, conforme ilustrado no quadro 15, pela chamada de método *passaVerde()* levando como parâmetro *false* para um pedido normal, ou *true* para um pedido com urgência.

QUADRO 15 - MÉTODO SOLICITA()

```
1  public void solicita(){
2    try {
3      if (possoPedirVerde(true)) {
4        this.tescoamento = escoamento();
5        passaVerde(true);
6      } else {
7        if (possoPedirVerde(false)) {
8          this.tescoamento = escoamento();
9          passaVerde(false);
10       }
11     }
12   } catch (Exception e) {
13     System.err.println("Erro na Solicitação ");
14   }
15 }
16
17 int escoamento(){
18   double tempo;
19   //calcula o tempo de verde necessario para o escoamento da fila
20   //Te = Ta + (N1 - 1) * _t + (Cv/Vm) + (((N1 * (C + 0,5))+Cv)/V)
21   tempo = (this.atraso + (this.qtdecarros -1) * this.arranque +
              (this.distembalagem/(this.velocidade/2)) +
              (((this.qtdecarros*(this.compveiculo+0.5))+
              this.distembalagem)/this.velocidade));
22   if ((tempo - (int)tempo) > 0 ) {
23     return (int)tempo + 1;
24   } else {
25     return (int)tempo;
26   }
27 }
```

O método p*assaVerde()*, ilustrado no quadro 17, faz parte do módulo de comunicação, onde é enviado uma mensagem pedindo o objeto de negociação ao agente-semáforo que o possui, aguardando por uma resposta. Esta resposta é tratada e caso receba o direito do objeto de negociação, é feito o envio de uma mensagem para cada vizinho do cruzamento avisando sobre o novo dono do objeto de negociação através da chamada ao método *avisaVizinho()*, ilustrado no quadro 18, passando como parâmetro a identificação do vizinho a ser enviada.

QUADRO 16 - MÉTODO POSSOPEDIRVERDE()

```
1 public boolean possoPedirVerde(boolean urgente){
2   if (!urgente) {
3     if (!(this.verde.equals(this.local)) && (this.qtdecarros > 0)) {
4       //verifica se deve pedir verde sem urgencia
5       if ((((100 * (this.qtdecarros * (this.compveiculo + 0.5))) / this.comprimento)
            > this.ftocupacao) || ((tempoEspera(this.tpaciencia) > this.ftpaciencia))) {
6         return true;       //Retorna que deve pedir
7       } else {
8         return false;      //Retorna que nao deve pedir
9       }
10    } else {
11      return false;
12    }
13  } else {
14    if (!(this.verde.equals(this.local)) && (this.qtdecarros > 0)) {
15      //verifica se deve pedir verde com urgencia
16      if ((((100 * (this.qtdecarros * (this.compveiculo + 0.5))) / this.comprimento)
            >= this.ftuocupacao)
            || ((tempoEspera(this.tpaciencia) >= this.ftupaciencia))) {
17        return true;       //Retorna que deve pedir
18      } else {
19        return false;      //Retorna que nao deve pedir
20      }
21    } else {
22      return false;
23    }
24  }
25}
26
27 int tempoEspera(Date inicio){
28   Date agora = new Date();
29   if (((agora.getTime() - inicio.getTime()) / 1000 ) <= 40)
30     return 90;
31   else if (((agora.getTime() - inicio.getTime()) / 1000 ) <= 60)
32     return 100;
33   else if (((agora.getTime() - inicio.getTime()) / 1000 ) <= 80)
34     return 110;
35   else if (((agora.getTime() - inicio.getTime()) / 1000 ) <= 100)
36     return 120;
37   else
38     return 0;
39 }
```

O tratamento das mensagens recebidas é feito pelo método *responde()*, ilustrado no quadro 19, programado para tratar o recebimento de quatro tipos de mensagem:

a) pedido do objeto de negociação: ao receber este tipo de mensagem é acionado o módulo de decisão através da chamada ao método *possoEnviarVerde()*, o qual analisa e só libera o objeto de negociação se o agente-semáforo proprietário do objeto já o utilizou por mais tempo que o fator de ocupação configurado conforme ilustrado no quadro 20. Caso o módulo de decisão resolver por passar o objeto de negociação, o método *responde()* envia uma resposta positiva à mensagem recebida;

b) pedido do objeto de negociação com urgência: possui as mesmas características do item anterior, com a diferença de ser comparado com os fatores de urgência;

c) aviso da entrada de carro na via: o recebimento deste tipo de mensagem faz com que o agente-semáforo altere em seu módulo de memória a quantidade de carros existentes em sua via, para isto esta mensagem não necessita de nenhum tipo de resposta;

d) aviso do recebimento do objeto de negociação por um outro agente-semáforo: o agente-semáforo ao receber este tipo de mensagem altera em seu módulo de memória qual é o agente que possui o objeto de negociação.

QUADRO 17 - MÉTODO PASSAVERDE()

```
1  public boolean passaVerde(boolean urgente) {
2    Message p = new Message("(ask-one :language LT :ontology transito)");
3    if (urgente) {
4      p.put("content", "QueroVerdeUrgente");
5    } else {
6      p.put("content", "QueroVerde");
7    }
8    try {
9      p.put("receiver", (String)this.verde);
10     Message r = mbox.ask(p);      //Envia a msg de pedido e aguarda a resposta
11     if (r != null){
12       if ((r.get("language").equals("LT"))&&(r.get("ontology").equals("transito")))  {
13         if (r.get("performative").equals("tell")) {
14           // ganhou o verde
15           this.verde = this.local ;
16           this.qtdecarros = 0;
17           this.inicioverde = new Date();
18           //enviar msg p/ todos do cruzamento q ganhou o verde
19           Message v = new Message("(ask-one :content TenhoVerde
                                              :language LT :ontology transito)");
20           v.put("receiver","Simulador");
21           mbox.sendMsg(v);
22           avisaVizinho(this.superior);
23           avisaVizinho(this.esquerdo);
24           avisaVizinho(this.direito);
25           return true;
26         } else {
27           if (r.get("performative").equals("cancel")) {
28             System.out.println("Nao ganhou o verde");
29           }
30           if (r.get("performative").equals("error")) {
31             System.out.println("Mensagem de erro recebida");
32           }
33           return false;
34         }
35       } else {
36         return false;
37       }
38     } else {
39       return false;
40     }
41   } catch (Exception e) {
42     p = null;
43     return false;
44   }
45 }
```

QUADRO 18 - MÉTODO AVISAVIZINO()

```
1  boolean avisaVizinho(String idvizinho){
2    try {
3      if (!((idvizinho).equalsIgnoreCase("Semaforo0"))) {
4        Message m = new Message("
                        (tell :content TenhoVerde :language LT :ontology transito)");
5        m.put("receiver",(String)idvizinho);
6        mbox.sendMsg(m);
7        return true;
8      } else {
9        return false ;
10     }
11   } catch (Exception e){
12     return false;
13   }
14 }
```

Nos casos em que o agente-semáforo recebe o pedido do objeto de negociação, antes de enviar a mensagem passando o objeto de negociação, o mesmo deve aguardar os tempos de segurança (amarelo e vermelho de segurança, calculados pelas equações VII e VIII), através da chamada de método *aguardaSeguranca()*.

O método *aguardaSeguranca()* é ilustrado no quadro 21 e faz com que o agente-semáforo aguarde o tempo necessário para as fases do sinal amarelo (quadro 21, linha 16) e vermelho (quadro 21, linha25), assim enviando a cada fase uma mensagem ao simulador sobre o seu estado atual. Isto é feito para que o simulador gere um arquivo com as decisões tomadas pelos agentes.

QUADRO 19 - MÉTODO RESPONDE()

```
1   public void responde(){
2     Message m;
3     try {
4       m = mbox.polling(100);
5     } catch (Exception e) {
6       m = null;
7     }
8     if (m != null){
9       if (!(m.get("performative").equals("error"))) {
10        if ((m.get("language").equals("LT"))&&(m.get("ontology").equals("transito"))) {
11          if(m.get("performative").equals("ask-one")) {
12            Message r = new Message ("(tell)");
13            r.put("receiver", m.get("sender"));
14            r.put("in-reply-to", m.get("reply-with"));
15            r.put("ontology","transito");
16            r.put("language","LT");
17            if (m.get("content").equals("QueroVerde")) {
18              r.put("content", "QueroVerde");
19              if (!(possoenviarverde(false))) {
20                r.put("performative", "cancel");
21              } else {
22                aguardaSeguranca();
23                this.verde = (String)m.get("sender");
24                this.tpaciencia = new Date();
25              }
26            }
27            if (m.get("content").equals("QueroVerdeUrgente")) {
28              r.put("content", "QueroVerdeUrgente");
29              if (!(possoEnviarVerde(true))) {
30                r.put("performative", "cancel");
31              } else {
32                aguardaSeguranca();
33                this.tpaciencia = new Date();
34              }
35            }
36            if ((m.get("content").equals("TenhoVerde"))) {
37              r.put("content","TenhoVerde");
38              this.verde = (String)m.get("sender");
39            }
40           try {
41              mbox.sendMsg(r);
42            } catch (Exception e) {
43              System.err.println("Erro no envio da resposta ");
44              r = null;
45            }
46          }
47          if(m.get("performative").equals("tell")) {
48            if ((m.get("content").equals("EntrouCarro"))) {
49              //r.put("content", "EntrouCarro");
50              if (this.qtdecarros == 0) {
51                this.tpaciencia = new Date();
52              }
53              this.qtdecarros = this.qtdecarros +
54                                Integer.parseInt((String)m.get("qtde"));
55            }
56          }
57        }
58      }
59    }
60  }
```

QUADRO 20 - MÉTODO POSSOENVIARVERDE()

```
1   public boolean possoEnviarVerde(boolean urgente) {
2     try {
3       if (this.verde.equals(this.local)) {
4         Date agora = new Date();
5         if (!urgente) {
6           if ((100 * ((agora.getTime() - this.inicioverde.getTime())/1000))
7               > this.fescoamento )
8             return true;
9           } else {
10            return false;
11          }
12        } else {
13          if ((100 * ((agora.getTime() - this.inicioverde.getTime())/1000))
14                > this.fuescoamento )
15            return true;
16          } else {
17            return false;
18          }
19        }
20      } else {
21        return false;
22      }
23    } catch (Exception e) {
24      return false;
25    }
26  }
```

QUADRO 21 - MÉTODO AGUARDASEGURANCA()

```
1  public void aguardaSeguranca(){
2     try {
3       Message a = new Message("
4                       (tell :content TenhoAmarelo :language LT :ontology transito)");
        a.put("receiver","Simulador");
5       mbox.sendMsg(a);
6       Thread.sleep(amarelo()*1000);//Aguarda o tempo de amarelo
7       this.tpaciencia = new Date();
8       Message v = new Message("
                        (tell :content TenhoVermelho :language LT :ontology transito)");
9       v.put("receiver","Simulador");
10      mbox.sendMsg(v);
11      Thread.sleep(vermelhoseguranca()*1000);//Aguarda o tempo de vermelho
12    } catch (Exception e){
13      System.err.println("Erro aguardando tempos de segurança");
14    }
15  }
16  int amarelo(){ /Ta = Tpr + (V/(2*a))
17    double tempo;
18    tempo = 1 + (this.velocidade /(2*2.8));
19    if ((tempo - (int)tempo) > 0 ) {
20      return (int)tempo + 1;
21    } else {
22      return (int)tempo;
23    }
24  }
25  int vermelhoSeguranca(){ /Tvs = (L + C)/V
26    double  tempo;
27    tempo = (this.compveiculo + this.cruzamento)/this.velocidade;
28    if ((tempo - (int)tempo) > 0 ) {
29      return (int)tempo + 1;
30    } else {
31      return (int)tempo;
32    }
33  }
```

5.2 AGENTE-SIMULADOR

Para a implementação do agente-simulador foi desenvolvida a classe *Simulador* e para interação do usuário com o sistema foi desenvolvida uma interface definida pela classe *Gerencia*.

Ao ser criado, o agente-simulador é iniciado através da chamada ao método *initAg()*. Através deste método é realizado o anúncio de suas habilidades nas páginas amarelas (quadro 22, linhas 4 a 6) e, em seguida é realizado uma busca dos agentes que possuem a habilidade de receber mensagens avisando sobre a entrada de carros em sua via (quadro 22, linha 7). Posteriormente é enviado para a classe *Gerencia* através da chamada do método *setSemaforos()* os nomes destes agentes, para que possam ser exibidos futuramente ao usuário do sistema.

QUADRO 22 - MÉTODO INITAG()

```
1  public void initAg(String[] args) {
2    try {
3      System.out.println("Iniciando Simulador");
4      mbox.advertise("tell","LT","transito","TenhoVerde");
5      mbox.advertise("tell","LT","transito","TenhoAmarelo");
6      mbox.advertise("tell","LT","transito","TenhoVermelho");
7      receptores = mbox.recommendAll("tell","LT","transito","EntrouCarro");
8      frm.setSize(new Dimension(582, 412));
9      frm.show();
10     frm.setSemaforos(receptores);
11   } catch (Exception e){
12     System.err.println("Erro Iniciando Semaforo "  + e );
13   }
14 }
15
16 public void run() {
17   while (true) {
18     semaforos = frm.getCarros();
19     if (semaforos.size()>0) {
20       responde();
21       enviaCarros();
22     }
23   }
24 }
```

Após iniciado, o agente-simulador é colocado em execução através da chamada ao método *run()*, sendo verificado neste método através da chamada de método *getCarros()* conforme ilustra quadro 22, se o usuário já informou ao simulador os dados sobre quantidade de carros por segundos que entram na via de cada semáforo necessários para a realização da simulação. Caso já tenham sido informados os dados para simulação, o agente-simulador está apto a tratar mensagens recebidas através do método *responde()* e enviar mensagens aos

agentes-semáforos avisando sobre a entrada de carros em sua via através do método *enviaCarros()*.

O método *responde()*, ilustrado no quadro 23, é responsável por tratar as mensagens recebidas enviadas pelos agentes-semáforos, onde fica aguardando por um tempo de um segundo (quadro 23, linha 4), podendo tratar três tipos de mensagens: *TenhoAmarelo* (quadro 23, linha 13), *TenhoVermelho* (quadro 23, linha 16) e *TenhoVerde* (quadro 23, linha 19).

QUADRO 23 - MÉTODO RESPONDE()

```
1 public void responde(){
2   Message m;
3   try {
4     m = mbox.polling(1000);
5   } catch (Exception e) {
6     m = null;
7   }
8   try {
9     if (m != null){
10      if (!(m.get("performative").equals("error"))) {
11        if ((m.get("language").equals("LT"))&&(m.get("ontology").equals("transito"))) {
12          if(m.get("performative").equals("ask-one")) {
13            if (m.get("content").equals("TenhoAmarelo")) {
14              gravaLog("Em " + new Date() + " " + m.get("sender") +
                        " iniciou fase amarela.", "c:/sema.txt");
15            }
16            if (m.get("content").equals("TenhoVermelho")) {
17              gravaLog("Em " + new Date() + " " + m.get("sender") +
                        " iniciou fase vermelha.", "c:/sema.txt");
18            }
19            if (m.get("content").equals("TenhoVerde")) {
20              gravaLog("Em " + new Date() + " " + m.get("sender") +
                        " iniciou fase verde.", "c:/sema.txt");
21            }
22          }
23        }
24      }
25    }
26  } catch (Exception e) {
27    System.out.println("Erro na recebimento de mensagem" + e);
28  }
29}
30
31 public void gravaLog(String gravar, String arquivo){
32   try {
33     PrintWriter out = new PrintWriter(new FileWriter(arquivo, true));
34     out.println(gravar);
35     out.close();
36   } catch (Exception e) {
37     System.out.println("Erro gravando Log");
38   }
39 }
```

Ao receber alguma destas mensagens, o agente-simulador grava no arquivo de *log* o agente-semáforo que enviou a mensagem juntamente com o seu estado e a hora, sendo esta gravação realizada pelo método *gravaLog()*, ilustrado no quadro 23, linhas 31 a 39.

Após o método *responde()* aguardar pelo tempo de um segundo e responder as mensagens recebidas, a chamada ao método *enviaCarros()*, ilustrado no quadro 24, faz com que o agente-simulador envie mensagens aos agentes-semáforos, contendo a quantidade de carros que entram por segundo em sua via, tentando realizar assim a simulação de um mundo real.

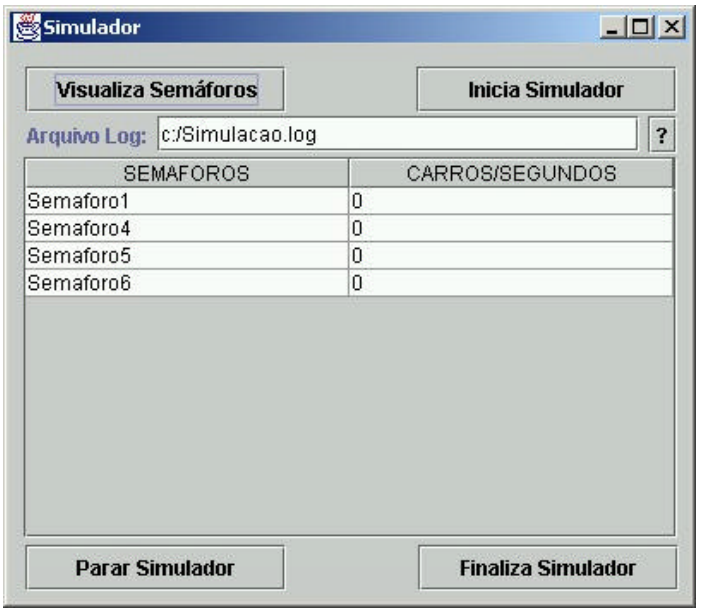
QUADRO 24 - MÉTODO ENVIACARROS()

```
1 public void enviaCarros(){
2   try {
3     for (int i=0; i < semaforos.size(); i++) {
4       Vector linha = (Vector)semaforos.elementAt(i);
5       Message e = new Message("(tell :language LT :ontology transito)");
6       e.put("content", "EntrouCarro");
7       e.put("qtde", (String)linha.elementAt((int)1));
8       e.put("receiver", (String)linha.elementAt((int)0));
9       mbox.sendMsg(e);
10    }
11  } catch (Exception e) {
12    System.out.println("Erro enviando carros");
13  }
14}
```

5.2.1 INTERFACE

Ao ser criado o agente-simulador é instanciada a classe *Gerencia*, a qual implementa um formulário, conforme ilustrado na fig. 18, para que seja possível ao usuário informar a quantidade de carros por segundos que entram na via de cada agente-semáforo.

FIGURA 18 - INTERFACE SIMULADOR

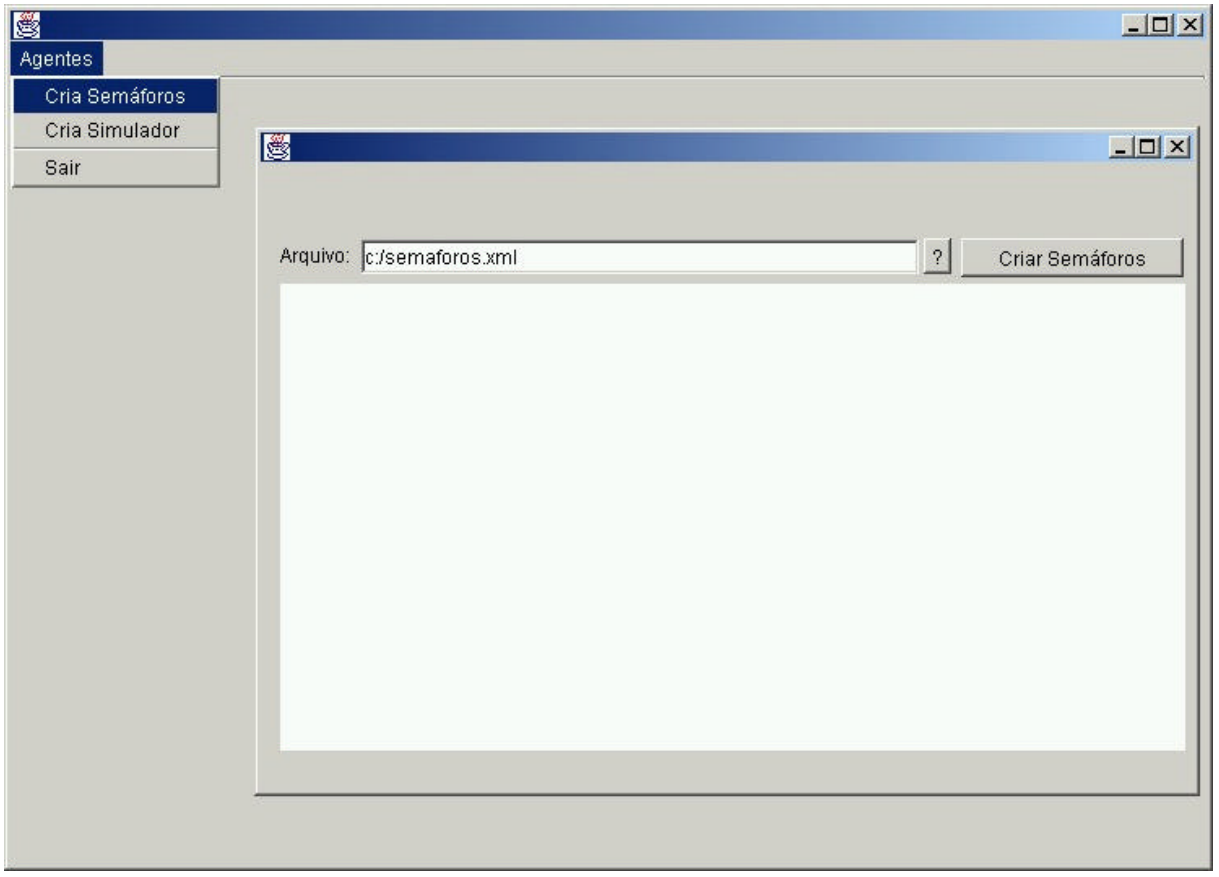


Após ser informada a quantidade de carros por segundo referente à via de cada agente-semáforo, o botão *Inicia Simulador* armazena os dados informados deixando-os disponíveis para o agente-simulador iniciar o envio das mensagens de entrada de carro aos agentes-semáforos.

5.3 CRIAÇÃO DOS AGENTES

Para facilitar a criação dos agentes foi desenvolvida uma interface conforme ilustra fig. 19, sendo possível através da mesma criar os agentes-semáforos e o agente-simulador. A criação dos agentes-semáforos é feita através da leitura do arquivo de entrada que contém os dados sobre os semáforos e seu cruzamento conforme ilustrado no quadro 25.

FIGURA 19 - CRIAÇÃO DOS AGENTES



O arquivo de entrada XML possui os seguintes atributos:

a) id_local: nome do semáforo a ser criado;
b) id_superior: nome do semáforo superior;
c) id_esquerdo: nome do semáforo esquerdo;

d) id_direito: nome do semáforo direito;

e) id_inferior: nome do semáforo inferior;

f) cruzamento: largura do cruzamento;

g) velocidade: velocidade da via;

h) comprimento: comprimento da via;

i) verde: semáforo do cruzamento que inicia com o objeto de negociação;

j) maquina: nome da máquina onde será criado o agente-semáforo.

QUADRO 25 - ARQUIVO DE ENTRADA

```
<Cidade>
    <Cruzamento id_Cruzamento="1" verde="1">
       <semaforo id_local="1" id_superior="4" id_esquerdo="5" id_direito="6"
       id_inferior="3" cruzamento="10" velocidade="11" comprimento="300"
       maquina="marcelo"/>

       <semaforo id_local="4" id_superior="1" id_esquerdo="6" id_direito="5"
       id_inferior="0" cruzamento="10" velocidade="11" comprimento="500"
       maquina="marcelo"/>

       <semaforo id_local="5" id_superior="6" id_esquerdo="4" id_direito="1"
       id_inferior="0" cruzamento="10" velocidade="11" comprimento="200"
       maquina="marcelo"/>

       <semaforo id_local="6" id_superior="5" id_esquerdo="1" id_direito="4"
       id_inferior="0" cruzamento="10" velocidade="11" comprimento="800"
       maquina="marcelo"/>
    </Cruzamento>
</Cidade>
```

Por ser um arquivo de entrada no formato XML, foi utilizado um *parser* implementado através da *Simple API for XML* (SAX) que faz a leitura dos atributos e para cada elemento faz uma chamada ao método *criaAgente()*, conforme ilustra quadro 26 linha 6, onde é realizado a criação do agente-semáforo.

QUADRO 26 - LEITURA DOS ATRIBUTOS DE UM ELEMENTO EM XML

```
1   public void startElement(String namespaceURI, String localName, String rawName,
                             Attributes atts) throws SAXException {
2       if (localName.equalsIgnoreCase("cruzamento")) {
          jedpLog.setText(jedpLog.getText() + "\n Novo Cruzamento \n");
3         this.verde = atts.getValue("verde");
4       } else {
5         jedpLog.setText(jedpLog.getText() + "\n Novo Semaforo \n");
6         criaAgente(atts.getValue("id_local"),atts.getValue("id_superior"),
          atts.getValue("id_esquerdo"),atts.getValue("id_direito"),
          atts.getValue("id_inferior"),atts.getValue("cruzamento"),
          atts.getValue("velocidade"),atts.getValue("comprimento"),
          this.verde atts.getValue("maquina"));
7       }
```

Para a criação do agente-semáforo é localizado o *launcher* existente criado pelo SACI (quadro 27, linha 4), sendo consolidada a criação do agente-semáforo através da utilização do comando *START_AGENT*, onde através do mesmo são informados os argumentos necessários para a criação do agente através do SACI, conforme ilustra quadro 27.

QUADRO 27 - CRIAAGENTE()

```
1 void criaAgente(String Local, String Superior, String Esquerdo, String Direito,
                  String Inferior, String Cruzamento, String Velocidade,
                  String Comprimento, String Verde, String Maquina) {
3   try {
4     Launcher l = Agent.getLauncher();
5     String args = Local + " " + Superior + " " + Esquerdo + " " + Direito + " " +
                    Inferior + " " + Cruzamento + " " + Velocidade + " " +
                    Comprimento +  " " + Verde +  " " + Maquina;
6
7     Command c1 = new Command(Command.START_AGENT);
8     c1.addArg("class", "Semaforo");
9     c1.addArg("society", "SocSem");
10    c1.addArg("name",  "Semaforo" + Local);
11    c1.addArg("creationMethod", "thread");
12    c1.addArg("host", Maquina);
13    c1.addArg("args",args);
14    Vector allIds = (Vector)l.execCommand("Test",c1);
15    if (allIds != null) {
16      AgentId a2 = (AgentId)allIds.get(0);
17      jedpLog.setText(jedpLog.getText() + " ** Agente criado, nome: "+a2.getName());
18    } else {
19      jedpLog.setText(jedpLog.getText() + "\n Problema na Criação");
20    }
31  } catch (Exception e) {
42    jedpLog.setText(jedpLog.getText() + "\n Error:"+e);
23  }
24}
```

5.4 ESTUDO DE CASO

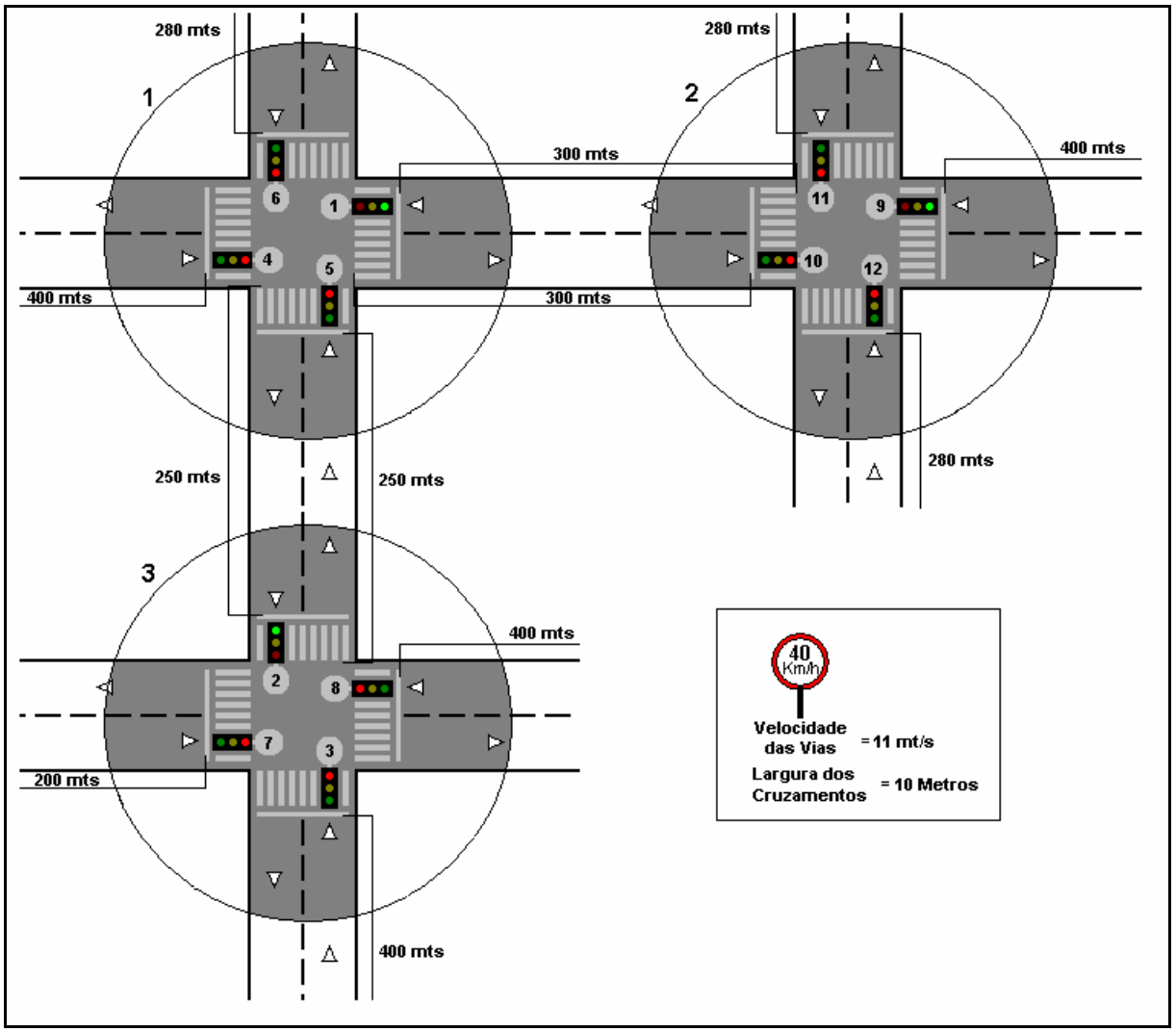
Para demonstração da implementação é realizado um estudo de caso em uma malha viária com três cruzamentos, computando ao todo doze semáforos, tendo todas as vias uma velocidade permitida de 40Km/h, e largura de cruzamentos iguais a 10 metros, conforme ilustra fig. 20.

Tendo o mapeamento da malha viária que se deseja realizar o controle e gerenciamento, deve-se partir para a elaboração do arquivo de entrada no formato XML, conforme ilustra quadro 28. Além dos dados sobre o cruzamento, é necessário informar ao arquivo de entrada, o semáforo que inicia com o objeto de negociação para cada cruzamento,

sendo definido para este estudo de caso, como dono do objeto no cruzamento 1 o semáforo 1, para o cruzamento 2 o semáforo 9 e para o cruzamento 3 o semáforo 2.

Também é necessário informar no arquivo de entrada a máquina onde será criado cada agente-semáforo, assim utiliza-se neste estudo de caso a criação dos semáforos dos cruzamentos 1 e 2 na máquina local denominada *Marcelo* e os semáforos do cruzamento 3 em outra máquina denominada *Leo*.

FIGURA 20 - CRUZAMENTO PARA ESTUDO DE CASO



QUADRO 28 - ARQUIVO XML ESTUDO DE CASO

```
<Cidade>
	<Cruzamento id_Cruzamento="1" verde="1">
		<semaforo id_local="1"  id_superior="4" id_esquerdo="5"  id_direito="6"
		id_inferior="9"  cruzamento="10" velocidade="11" comprimento="300"
		maquina="marcelo"/>
		<semaforo id_local="4"  id_superior="1" id_esquerdo="6"  id_direito="5"
		id_inferior="0"  cruzamento="10" velocidade="11" comprimento="400"
		maquina="marcelo"/>
		<semaforo id_local="5"  id_superior="6" id_esquerdo="4"  id_direito="1"
		id_inferior="3"  cruzamento="10" velocidade="11" comprimento="250"
		maquina="marcelo"/>
		<semaforo id_local="6"  id_superior="5" id_esquerdo="1"  id_direito="4"
		id_inferior="0"  cruzamento="10" velocidade="11" comprimento="280"
		maquina="marcelo"/>
	</Cruzamento>
	<Cruzamento id_Cruzamento="2" verde="2">
		<semaforo id_local="3"  id_superior="2" id_esquerdo="7"  id_direito="8"
		id_inferior="0"  cruzamento="10" velocidade="11" comprimento="400"
		maquina="leo"/>
		<semaforo id_local="2"  id_superior="3" id_esquerdo="8"  id_direito="7"
		id_inferior="6"  cruzamento="10" velocidade="11" comprimento="250"
		maquina="leo"/>
		<semaforo id_local="7"  id_superior="8" id_esquerdo="2"  id_direito="3"
		id_inferior="0"  cruzamento="10" velocidade="11" comprimento="200"
		maquina="leo"/>
		<semaforo id_local="8"  id_superior="7" id_esquerdo="3"  id_direito="2"
		id_inferior="0"  cruzamento="10" velocidade="11" comprimento="400"
		maquina="leo"/>
	</Cruzamento>
	<Cruzamento id_Cruzamento="3" verde="9">
		<semaforo id_local="9"  id_superior="10" id_esquerdo="12"
		id_direito="11" id_inferior="0"  cruzamento="10" velocidade="11"
		comprimento="400" maquina="marcelo"/>
		<semaforo id_local="10" id_superior="9"  id_esquerdo="11"
		id_direito="12" id_inferior="4"  cruzamento="10" velocidade="11"
		comprimento="300" maquina="marcelo"/>
		<semaforo id_local="11" id_superior="12" id_esquerdo="9"
		id_direito="10" id_inferior="0"  cruzamento="10" velocidade="11"
		comprimento="280" maquina="marcelo"/>
		<semaforo id_local="12" id_superior="11" id_esquerdo="10"
		id_direito="9"  id_inferior="0"  cruzamento="10" velocidade="11"
		comprimento="280" maquina="marcelo"/>
	</Cruzamento>
```

Para a utilização do sistema, primeiramente as classes *Semaforo* e *Simulador* devem estar no diretório de bibliotecas de usuário do SACI (saci/ulib/). Após deve ser iniciado o SACI para que seja criado o *launcher*. Para isto existem duas opções:

a) através da execução do arquivo *saci.bat*: que inicia toda a interface gráfica do SACI com suas ferramentas, contendo um administrador para o *launcher*, conforme ilustra fig. 21;

b) através da execução do arquivo *launcherd.bat* que exibe apenas uma janela em modo texto, conforme ilustra fig. 22.

Devido ao sistema neste estudo de caso estar distribuído em duas máquinas, deve-se criar um *launcher* em cada máquina onde se deseja criar os agentes-semáforos.

Tendo-se criado os *launchers* necessários, inicia-se a criação dos agentes-semáforos, informando na interface gráfica de criação dos agentes-semáforos o arquivo de entrada e sua localização, conforme ilustrado na fig. 19. Ao serem criados, são expostas na tela as informações sobre os agentes-semáforos com seus atributos, conforme ilustra fig. 23, podendo também ser acompanhada a criação através do *laucher* no SACI, conforme ilustra fig. 22.

FIGURA 21 - SACI

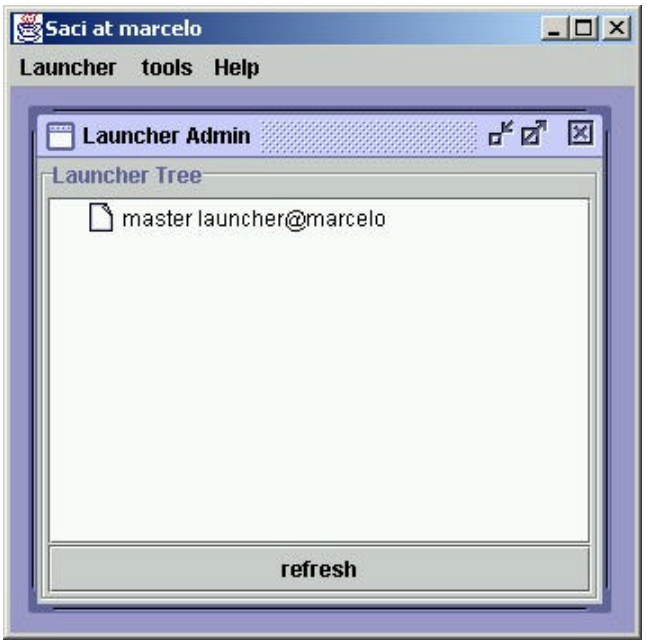


FIGURA 22 - LAUCHERD

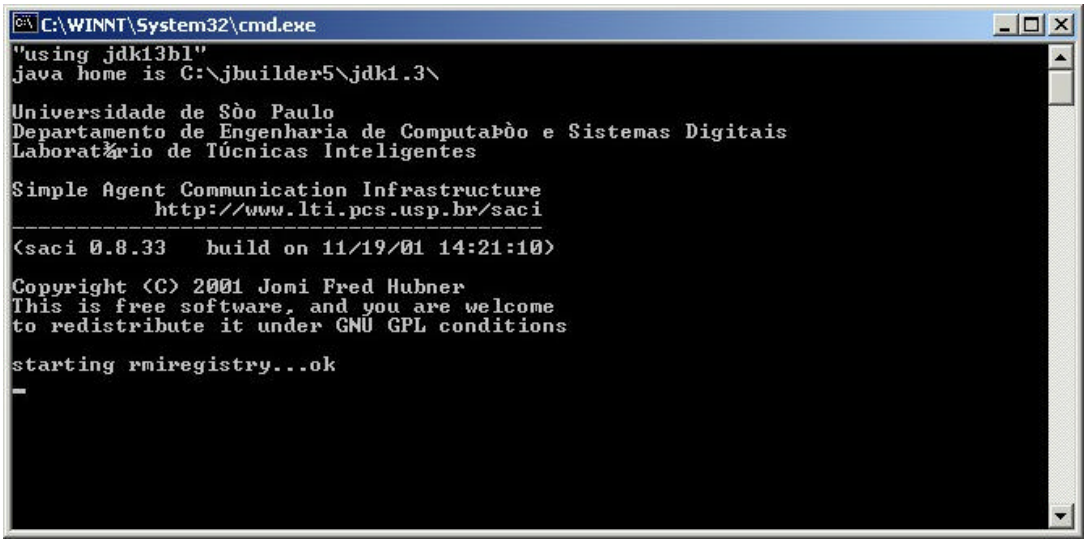


FIGURA 23 - CRIAÇÃO DOS AGENTES-SEMÁFOROS

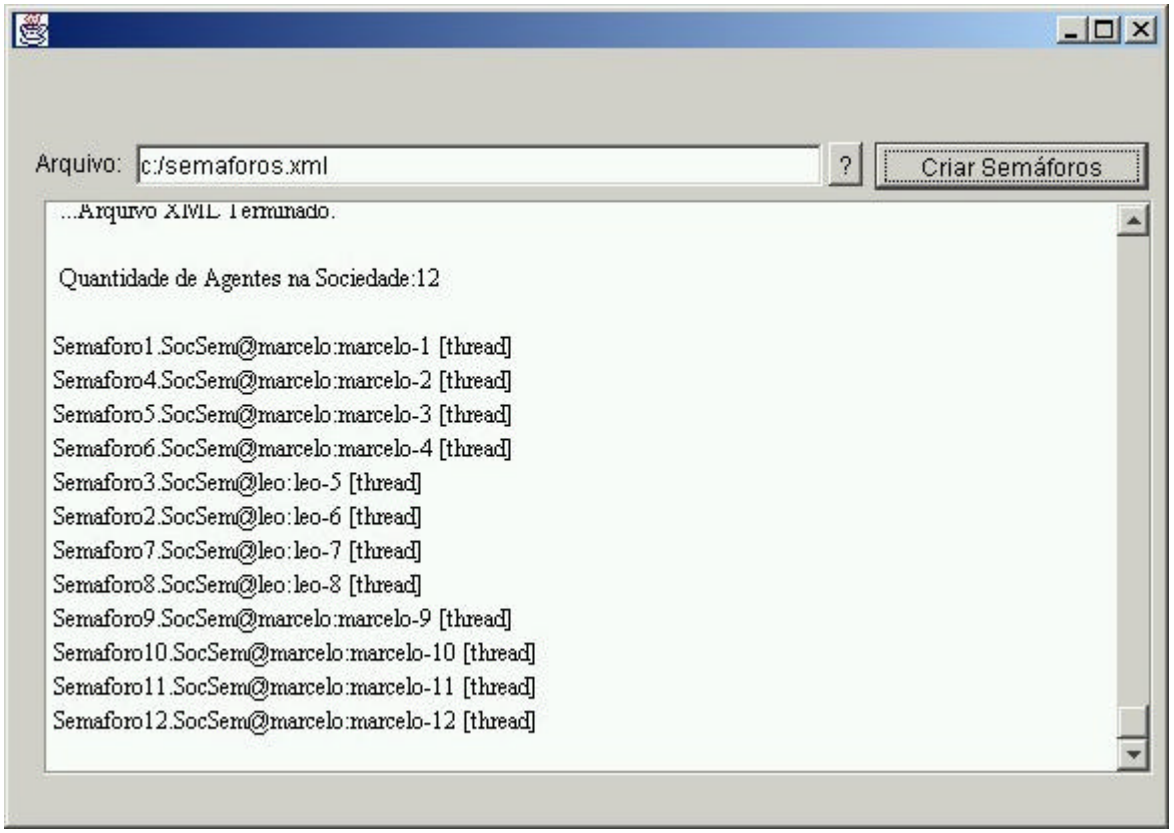
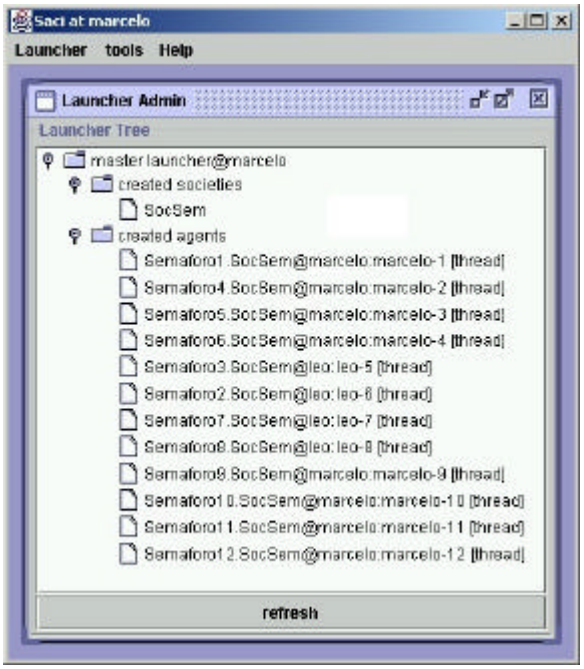


FIGURA 24 - LAUNCHER ADMIN



Após a criação dos agentes-semáforos deve ser realizada a criação do agente-simulador, ilustrado na fig. 25. Sendo criada a interface gráfica, deve-se informar o local e o arquivo onde o simulador gravará as informações da simulação e posteriormente, fazer uso do botão *Visualizar Semáforos* para que sejam exibidos os agentes-semáforos contidos no *launcher* e que tenham as habilidades relacionadas ao sistema (:ontologia transito :linguagem LT). Em seguida informar para cada agente-semáforo existente a quantidade de carros por segundo que entram em sua via.

FIGURA 25 - SIMULADOR

A utilização do botão *Inicia Simulador* faz com que o agente-simulador inicie o envio de mensagens aos agentes-semáforos avisando-os sobre a entrada de carros em sua via, ativando a comunicação e a negociação entre os agentes-semáforos e realizando a gravação dos resultados no arquivo de *log*, ilustrado no quadro 29. O envio aos agentes-semáforos das mensagens sobre a entrada de carros na via podem ser acompanhadas através do *communication monitor* existente no SACI.

O envio de mensagens aos agentes-semáforos sobre a entrada de carros pode ser interrompido a qualquer momento através a utilização do botão *Parar Simulador*, sendo que a troca de mensagens entre agentes-semáforos e o aviso dos agentes-semáforos ao simulador sobre o seu estado atual não é interrompida até que as decisões permitam negociar o objeto de negociação.

QUADRO 29 - ARQUIVO LOG

```
Em  Jun 19 00:08:29 2002 Semaforo2 iniciou fase amarela.
Em  Jun 19 00:08:32 2002 Semaforo2 iniciou fase vermelha.
Em  Jun 19 00:08:34 2002 Semaforo1 iniciou fase amarela.
Em  Jun 19 00:08:34 2002 Semaforo9 iniciou fase amarela.
Em  Jun 19 00:08:36 2002 Semaforo7 iniciou fase verde.
Em  Jun 19 00:08:36 2002 Semaforo1 iniciou fase vermelha.
Em  Jun 19 00:08:36 2002 Semaforo9 iniciou fase vermelha.
Em  Jun 19 00:08:38 2002 Semaforo6 iniciou fase verde.
Em  Jun 19 00:08:38 2002 Semaforo12 iniciou fase verde.
Em  Jun 19 00:08:46 2002 Semaforo6 iniciou fase amarela.
Em  Jun 19 00:08:48 2002 Semaforo6 iniciou fase vermelha.
Em  Jun 19 00:08:50 2002 Semaforo12 iniciou fase amarela.
Em  Jun 19 00:08:51 2002 Semaforo5 iniciou fase verde.
Em  Jun 19 00:08:53 2002 Semaforo12 iniciou fase vermelha.
Em  Jun 19 00:08:55 2002 Semaforo11 iniciou fase verde.
Em  Jun 19 00:09:06 2002 Semaforo11 iniciou fase amarela.
Em  Jun 19 00:09:06 2002 Semaforo5 iniciou fase amarela.
Em  Jun 19 00:09:07 2002 Semaforo7 iniciou fase amarela.
Em  Jun 19 00:09:09 2002 Semaforo5 iniciou fase vermelha.
Em  Jun 19 00:09:10 2002 Semaforo11 iniciou fase vermelha.
Em  Jun 19 00:09:10 2002 Semaforo7 iniciou fase vermelha.
Em  Jun 19 00:09:11 2002 Semaforo4 iniciou fase verde.
Em  Jun 19 00:09:11 2002 Semaforo9 iniciou fase verde.
Em  Jun 19 00:09:12 2002 Semaforo3 iniciou fase verde.
Em  Jun 19 00:09:13 2002 Semaforo4 iniciou fase amarela.
Em  Jun 19 00:09:16 2002 Semaforo4 iniciou fase vermelha.
Em  Jun 19 00:09:18 2002 Semaforo5 iniciou fase verde.
Em  Jun 19 00:09:20 2002 Semaforo9 iniciou fase amarela.
Em  Jun 19 00:09:23 2002 Semaforo9 iniciou fase vermelha.
Em  Jun 19 00:09:25 2002 Semaforo11 iniciou fase verde.
Em  Jun 19 00:09:54 2002 Semaforo5 iniciou fase amarela.
Em  Jun 19 00:09:56 2002 Semaforo11 iniciou fase amarela.
Em  Jun 19 00:09:58 2002 Semaforo11 iniciou fase vermelha.
Em  Jun 19 00:09:59 2002 Semaforo5 iniciou fase vermelha.
Em  Jun 19 00:09:59 2002 Semaforo4 iniciou fase verde.
Em  Jun 19 00:09:59 2002 Semaforo9 iniciou fase verde.
.
.
```

Para a finalização do agente-simulador utiliza-se o botão *Finaliza Simulador*, sendo realizado também através deste botão, a finalização do SACI e dos agentes-semáforos.

5.5 RESULTADOS E DISCUSSÃO

Para análise dos resultados foram realizadas simulações com dezesseis malhas viárias diferentes, criadas aleatoriamente, não sendo nenhum modelo real existente, contendo até seis cruzamentos por malha viária, e informado valores variando entre um a quatro (carros por segundos) para os valores da entrada de carros em sua via.

Sendo utilizado o módulo de memória implementado, verificou-se os resultados apresentados na tabela 1.

TABELA 1 - RESULTADOS OBTIDOS

Pedidos	%	Negados	Fator Ocupação	Fator Paciência
Normais	82%	7%	69%	6%
Urgentes	18%	1%	8%	11%

Analisando os dados da tabela 1, verifica-se uma grande quantidade de pedidos do objeto de negociação em condições normais, sendo eles a maioria por ultrapassar o fator de ocupação permitido e poucos ultrapassando o fator de paciência.

Portanto é conclusivo nos pedidos normais, que ultrapassando poucas vezes o fator de paciência, os motoristas das vias não esperam pelo direito de passagem por tempos prolongados, criando assim um trânsito calmo com esperas do direito de passagem dentro dos padrões permitidos.

Observando os pedidos urgentes, verifica-se que o fator de paciência é ultrapassado por mais freqüência que o fator de ocupação, concluindo que o fator de urgência resolve na maioria das vezes a espera do direito de passagem por um período maior do que os padrões permitidos.

Observa-se também um percentual negado para o pedido normal, sendo que esta negação não faz com que o agente-semáforo faça o pedido com urgência da próxima vez, por outro lado, a negação do pedido de urgência pode ocorrer em casos remotos.

6 CONCLUSÕES

Este trabalho apresentou a utilização de técnicas e conceitos de *Sistema Multi-Agentes* (SMA) para a implementação de um sistema de gerência e controle de tráfego urbano, tendo como enfoque a negociação entre os agentes-semáforos do direito de passagem de veículos nas vias realizada através da troca de mensagens KMQL e de fórmulas de decisões criadas. Contudo a gerência e o controle de tráfego urbano, não se limita apenas em sincronismo de semáforos, mas sim em uma análise e decisão em tempo real sobre as ações do mundo.

Por se utilizar o controle do direito de passagem dos veículos nos cruzamentos, o sincronismo dos semáforos dispostos em série foi prejudicado, por outro lado, não privilegiando apenas alguns semáforos, mas sim todo um conjunto trabalhando com um objetivo em comum.

Visando realizar o controle do tráfego através do direito de passagem dos veículos, a estratégia de decisão por pedir ou ceder o objeto de negociação correspondeu com as expectativas de não deixar os motoristas esperando por tempos maiores que os padrões permitidos e por não comprometer as vias esgotando a capacidade de sua ocupação. Por outro lado saiu do padrão de utilizar os tempos de ciclo semafóricos onde comumente é dado o direito de passagem a semáforos que não necessitam, fazendo assim a liberação do direito de passagem somente sobre a necessidade do semáforo.

Devido aos objetos de estudo do sistema encontrarem-se geograficamente distribuídos em um mundo real e não possuindo variáveis com valores fixos, a utilização de técnicas de SMA tornou o desenvolvimento do trabalho mais coerente com este mundo, onde foi possível distribuir os agentes-semáforos em várias máquinas, fazendo com que os agentes-semáforos agissem e reagissem de acordo com as situações de sua via em tempo real.

A utilização de SMA trouxe como problema, agentes que ficavam aguardando por respostas de agentes ocupados com alguma outra tarefa, assim não realizando a devida passagem do objeto de negociação entre eles. Sendo resolvidos problemas como este através da diminuição o tempo de espera por respostas.

O uso do SACI como ferramenta para a comunicação dos agentes, tornou o desenvolvimento do trabalho mais fácil e em conformidade com um padrão, com métodos

simplificados de criação e comunicação, diminuiu consideravelmente o tempo de desenvolvimento dos agentes e os erros comumente realizados na comunicação dos mesmos.

Foi de grande valia utilizar JAVA para a criação de todo o sistema, devido ao SACI ter sido desenvolvido em JAVA, foi possível fazer com que o sistema fosse portável, sendo encontrado informações sobre portablidade em Deitel (2001). Por se tratar de ferramentas de livre distribuição, tornam os custos de uma futura aplicação real mais reduzidos, sendo posteriormente também de grande facilidade a implantação destes tipos de sistemas em equipamentos que possuem *Java Virtual Machine* (JVM) incorporada.

6.1 EXTENSÕES

Propõem-se como extensões, vários estudos que complementam o trabalho desenvolvido:

a) criação de novas fórmulas de decisões: modificando ou acrescentando novas maneiras de fazer os agentes-semáforos agirem ou reagirem à fórmula de decisão já existente, colocando como base novos pontos de vista sobre tráfego;
b) estudo em outros tipos de vias: incluindo cruzamentos com faixa de conversão e fatores que simulem a entrada e saída de carros dos estacionamentos laterais existentes nas vias;
c) inclusão dos tempos de pedestres: incluir no controle dos tempos dos semáforos os tempos referidos à passagem dos pedestres, realizando também pelo simulador a simulação dos pedestres esperando para realizar a travessia;
d) interface para mapeamento dos semáforos na malha viária: desenvolver interface gráfica para a criação da malha viária, assim realizando o mapeamento dos semáforos, simplificando a criação do arquivo de entrada;
e) interface gráfica para exibição das ações dos semáforos: desenvolver uma interface gráfica que exiba os semáforos com seus estágios, alternando o direito de passagem entre eles;
f) desenvolver novos agentes: desenvolver no sistema como agentes os carros que circulam pelas vias, tentando realizar assim uma melhor simulação, criando mecanismos “inteligentes” nos carros, onde os mesmos possam escolher as vias que comumente possuem menos tráfego e menos paradas nos semáforos;

g) aplicação da implementação em equipamentos: incluir o sistema implementado em equipamentos (microprocessadores) que possuem *Java Virtual Machine* (JVM);
h) análise de *log*: desenvolver uma ferramenta que analise o arquivo de *log* gerado e mostre a funcionalidade do sistema.

Sendo assim verifica-se que o trabalho desenvolvido tem uma grande contribuição para o início do desenvolvimento de tecnologias de gerência de tráfego distribuídas, onde certamente este trabalho possui extensões que vão além das comentadas acima.

REFERÊNCIAS BIBLIOGRÁFICAS

BIGUS, Joseph P.; BIGUS, Jennifer. **Constructing intelligent agents with Java**: a programmer's guide to smarter applications. New York: Wiley Computer Publishing, 1997.

DEITEL, H. M.; DEITEL P. J. **Java**: como programar. Tradução Edson Furnankiewicz, 3. ed. Porto Alegre: Bookman, 2001.

DOT - U.S. Departament of Ttransportation. **Intelligent transportation system,** Washington, out. 2001. Disponível em: < http://www.its.dot.gov>. Acesso em: 10 nov. 2001.

EJZEMBERG, Sergio. **Análise da circulação e fluxos de tráfego.** São Paulo: Companhia de Engenharia de Tráfego, 1996.

ESPEL, Marcelo. **O controle eficaz dos semáforos para melhoria do tráfego urbano.** 2000, 53 f. Monografia (Especialista em Gestão Integrada de Trânsito), Universidade Católica de Santos, Santos.

JENNINGS, Nicholas R.; WOOLDRIDGE, Michael J. **Agent technology**: foundations, applications, and markets. London: Springer-Verlag, 1998.

HÜBNER, Jomi Fred. **Migração de agentes em sistemas multi-agentes abertos.** Dissertação (Mestrado em Ciências da Computação) UFRGS, Porto Alegre, 1995.

HÜBNER, Jomi Fred. **SACI**: simple agent communication infrastructure, São Paulo, mar. 2001. Disponível em: <http://www.lti.pcs.usp.br/saci>. Acesso em: 10 nov. 2001.

LOPES, Mauricio Antonio Ribeiro. **Código de trânsito brasileiro anotado.** São Paulo: Ed. Revista dos Tribunais, 1998.

MCSHANE, William R.; ROESS, Roger P. **Traffic engineering.** Englewood Cliffs: Prentice Hall, 1990.

MCT - Ministério da Ciência e Tecnologia. **Transporte e trânsito,** Brasília, mar. 2000. Disponível em: < http://www.cct.gov.br/gtsocinfo/atividades>. Acesso em: 10 nov. 2001.

MENESES, Eudênia Xavier. **Agentes,** São Paulo, out. 2001. Disponível em: <http://www.ime.usp.br/~eudenia/jaia>. Acesso em: 10 nov. 2001.

SANTOS, José Nazareno; SILVA Romero Tavares. **Pesquisa e ensino em física,** João Pessoa, abr. 2001. Disponível em: <http://www.fisica.ufpb.br/prolicen/produtos.html>. Acesso em: 22 mar. 2002.

STERN, Yvone et al. **Um estudo sobre tráfego**: sincronização de sinais. Rio de Janeiro: Instituto de Pesquisas Rodoviárias, 1969.

VILANOVA, Luis Molist. **Dimensionamento do tempo de amarelo**. Nota Técnica 108. Companhia de Engenharia de Tráfego. São Paulo, 1985. 16 p.

WEISS G. **Multiagent systems**: a modern approach to distributed artificial intelligence. Massachussetts: MIT Press, 1999.