Carros Autônomos Tolerantes a Falhas

Cleybson Cardoso Leite, Lucas Ventura Paiva

[1]Engenharia de Computação – Universidade Estadual de Feira de Santana (UEFS)
Feira de Satana – BA – Brazil

cleybsonc@gmail.com, lukspaiva95@gmail.com

Abstract. *Hardware errors were associated with problems found in the previous system. Therefore, this version of the autonomous cars system is proposed to be tolerant to faults and problems that can put at risk drivers and pedestrians using the roads.*

Resumo. *Problemas relacionados à erros de hardware foram encontrados no sistema anterior. Portanto, esta versão do sistema de carros autônomos propõe ser tolerante à falhas e problemas que possam por em risco motoristas e pedestres que utilizam as vias públicas.*

1. Introdução

Com a construção do sistema de carros autônomos, pôde-se analisar a relevância e importância desse projeto. Com os testes feitos percebeu-se que a quantidade de colisões diminuiu a zero porcento. E isso é um caso de muita alegria a todos, pois um cruzamento perigoso como o que foi utilizado para testes, através de pesquisas, com o tempo foi percebido que a maioria dos acidentes aconteciam por conta de um carro avançar no cruzamento, sendo que a colisão era facilmente percebida que iria acontecer, ou seja, a colisão acontecia por um motorista desatento e que ocasionava uma colisão, envolvendo ate o motorista que está dirigindo atentamente e corretamente.

Visando os benefícios que este avanço pode trazer a humanidade, está ideia dos carros autônomos foi testada de diversas forma, para detectar o máximo de falhas possíveis. E da forma que estava sendo feita, seria inviável quando a quantidade de carros for grande, pois da maneira que estava implementada, ocasionava o dobro de conexões necessárias.

Sendo assim, este problema deve ser resolvido, minimizando ao máximo as conexões e ainda assim, ser confiável e segura, para que não ocorram acidentes.

2. Fundamentação Teórica

2.1. Protocolo de transporte da internet UDP

O UDP (User Datagram Protocol). oferece um meio para as aplicações enviarem datagramas IP encapsulados sem que seja necessário estabelecer uma conexão, transmitindo segmentos que consistem em um cabeçalho de 8 bytes, seguido pela carga útil [Tanenbaum 2009].

Ele não realiza controle de fluxo, controle de erros ou retransmissão após a recepção de um segmento incorreto. Cabe ao usuário implementar essas ações (se necessário), gerando segurança na troca de informações. O que ele faz é fornecer uma interface para o protocolo IP com o recurso adicional de demultiplexação de vários processos que utilizam as portas [Tanenbaum 2009].

3. Metodologia

O sistema desenvolvido visa coordenar as ações de carros autônomos em um cruzamento entre duas vias (Figura 1). O objetivo é que todos os carros atravessem o cruzamento sem que ocorram colisões. Os carros se comunicam através da troca de mensagens, utilizando protocolo de transporte UDP e a partir das informações recebidas cada veículo executa determinada ação.

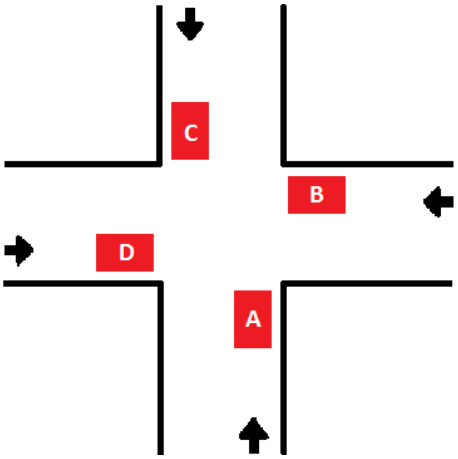

Figura 1. Modelo do cruzamento

Na primeira implementação dos carros autônomos, utilizamos o protocolo TCP, porém cada veículo tinha uma conexão ponto a ponto com os demais, havendo uma quantidade enorme de conexões. O *Socket* do 'Carro1' se conecta com o *SeverSocket* do 'Carro2', e o *Socket* do 'Carro2' se conecta com o *ServerSocket* do 'Carro1', como mostra a Figura 2. Portanto, a mudança no modelo de comunicação foi umas das melhorias aplicadas ao sistema.

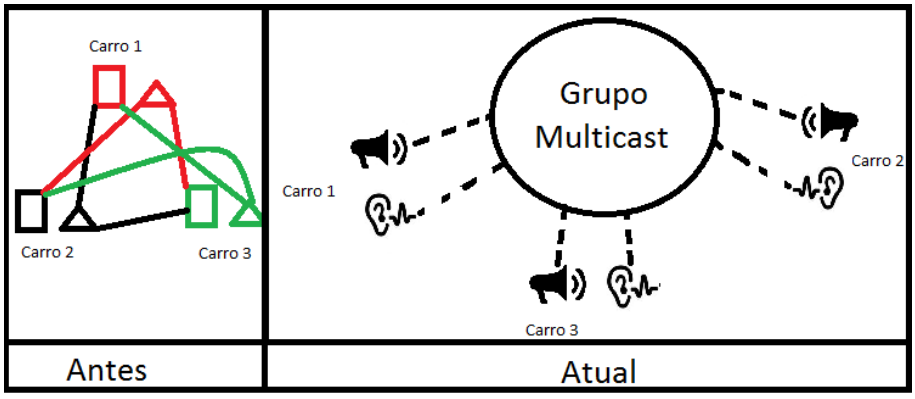

Figura 2. Antes e atual

Para que ocorra uma travessia segura os veículos sabem a localização exata uns dos outros, além de conhecerem o trajeto percorrido por cada um deles. Portanto, os carros estão sempre se comunicando ao longo do trajeto, enviando e recebendo a posição atual de cada um e o seu trajeto ao longo do percusso.

Como esse *software* se trata de uma modificação de um sistema anterior, foi possível resolver problemas e melhorar a forma como os carros interagem entre si. Inicialmente trocamos o protocolo TCP por UDP, tornando a comunicação mais rápida, porém menos confiável, sendo necessário a criação de algoritmos que tratassem de falhas (desconexão de um *host* por exemplo) e permitissem o correto funcionamento do sistema.

O *software* deve funcionar corretamente em um sistema real, onde não podemos confiar totalmente no *hardware*. Portanto, erros de perda de conexão com a rede devem ser tratados. Quando um veículo perde acesso à rede ele para e os outros veículos admitem que ele saiu da via, ignorando a presença daquele carro.

4. Resultados e Discussões

4.1. Tratamento de Colisão

O algoritmo para exclusão mútua é simples e de espera ociosa. Os veículos estão sempre enviando sua posição atual para os demais e comparando o trajeto que cada um vai percorrer com o próprio trajeto. Dessa forma, quando um carro entra em uma área de conflito, outro veículo que quiser entrar vai ficar esperando até que o primeiro saia daquele ponto.

Esse algoritmo é o mesmo utilizado na versão anterior do *software*, já foi testado e funciona perfeitamente no ambiente do cruzamento. Portanto, quando o programa é iniciado, basta o usuário informar qual a origem e o destino desejado e começar o trajeto. A partir daí serão enviadas mensagens contendo as coordenadas *(x,y)*, a direção do veículo, o IP do remetente e a trajetória atual do carro.

4.2. Comunicação

Como já foi mencionado, a comunicação entre os *hosts* ocorre utilizando o protocolo UDP. Cada veículo envia periodicamente informações sobre sua posição para um grupo de *multicast* onde todos os outros carros estão presentes. Dessa forma não há necessidade de conexão entre os *hosts*, promovendo maior simplicidade na troca de mensagens.

A utilização de UDP solucionou alguns problemas de comunicação que ocorriam quando ainda era utilizado o TCP. No sistema anterior, em alguns casos, acontecia de um *host* não se conectar com outro, ou se conectava duas vezes com o mesmo, gerando problemas na interface, gasto desnecessário de processamento e até possíveis colisões num sistema real.

O protocolo UDP é mais simples, porém não garante confiabilidade. Portanto, foi necessário elaborar algumas estratégias para que o sistema funcione corretamente tanto em um sistema virtual quanto numa situação real.

4.3. Tratamento de Erros

Quando um veículo está na via e outro carro entra, o primeiro recebe as informações do segundo e armazena uma *key* que identifica esse *host*. Dessa forma, toda mensagem recebida pelo mesmo *host* é armazenada substituindo as informações antigas. Essa *key* é gerada a partir do endereço IP daquela máquina.

Na recepção de mensagens o *host* verifica o endereço do remetente, pois, como ele também pertence ao grupo *multicast*, o *host* vai receber a mensagem que ele mesmo

enviou. Dessa forma, o seu veículo é tratado como único e a lógica de conflito é aplicada apenas aos demais *hosts*.

Para saber se um veículo presente na via perdeu a conexão com a rede, foi criado um sistema que armazena em um *ArrayList* o horário da ultima mensagem recebida pelo *host*. Uma *Thread* está sempre percorrendo esse *ArrayList* e verificando se o tempo da ultima mensagem passou de 5 segundos. Se sim, consideramos que esse veículo perdeu o acesso à rede e continuamos o trajeto sem nos preocuparmos com ele.

Quando o próprio *host* perde conexão com a rede, em um caso real, é necessário que o veículo pare, afim de evitar acidentes. Um carro que não está na rede não consegue visualizar a presenças de outros, se tornando um risco aos usuários. Portanto, quando um *host* sai da rede o sistema logo é fechado, garantindo a integridade de todos. Quando o veículo muda de rede o seu IP muda, sendo assim, a verificação ocorre comparando sempre o IP inicial com o atual, se diferente o sistema fecha.

5. Conclusão

O sistema funciona de forma totalmente distribuída, a comunicação é realizada conforme as requisições que nos foi passada, com isso os carros conseguem atravessar o cruzamento sem que ocorram colisões. A mudança para utilização do protocolo UDP nos permitiu perceber quais as formas de empregar confiabilidade ao sistema, tolerando falhas.

Carros autônomos são projetados para ambientes reais, portanto não pode haver total confiança no sistema de *hardware*, havendo necessidade de tratar possíveis falhas que ocorram nesses aparelhos. A correção dessas falhas permite que o sistema seja utilizado nas ruas, diminuindo a quantidade de acidentes nas vias.

Referências

Tanenbaum, A. S. (2009). *Sistemas Operacionais Modernos*. Pearson, 3th edition.