Segurança rodoviária suportada em computação de proximidade

Ana Araújo[1], Eva Castro[2], Filipe Peixoto[3] e Tiago Sousa[4]

[1] Universidade do Minho, Guimarães a89404@alunos.uminho.pt
[2] Universidade do Minho, Guimarães a93097@alunos.uminho.pt
[3] Universidade do Minho, Guimarães a93096@alunos.uminho.pt
[4] Universidade do Minho, Guimarães a89392@alunos.uminho.pt

Resumo Este artigo visa documentar o processo de desenvolvimento de um protótipo de rede veicular que promove a segurança rodoviária. O protótipo oferece um serviço que permite coligir informações em tempo real e prevenir acidentes de trânsito.

Palavras-chave: Redes Veiculares · Segurança Rodoviária

1 Introdução

Na unidade curricular de Redes Veiculares foi proposto um trabalho cujo objetivo é a construção de um protótipo funcional de um serviço de segurança rodoviária baseado na computação de proximidade.

O sistema pretendido permitirá que os veículos enviem mensagens com dados sobre a sua mobilidade, nas quais são recebidas tanto por outros veículos vizinhos, como por unidades de comunicação fixas instaladas na proximidade das vias de circulação.

Estas unidades computacionais têm a capacidade de criar uma visão abrangente da rede veicular local, permitindo a deteção de perigos diversos, que após a recolha e processamento dessas informações, podem enviar mensagens informativas e de alerta de segurança de volta aos veículos, com o objetivo principal de melhorar a segurança rodoviária.

2 Especificação do serviço

Em termos simples, e como já mencionado, o serviço construído tem como objetivo promover a segurança rodoviária. Assim, nesta secção será abordada a especificação do serviço oferecido, bem como os protocolos desenvolvidos que o suportam.

De forma genérica, o serviço implementado permite que uma unidade de computação, diretamente ligada à RSU (*Road Side Unit*), processe os dados recebidos, relativos a veículos na sua proximidade, e gere alertas de segurança. Também é possível que alertas de segurança sejam gerados nos próprios veículos.

Os veículos geram mensagens periódicas (1 em 1 segundo) com informações relativas a si mesmos e ao seu estado (mensagens CAM), i.e., o tipo de veículo, peso, dimensões, velocidade, posição e estado dos sensores (chuva, piso molhado, nevoeiro).

Por sua vez, estas mensagens são envidas para o veículo mais próximo da RSU no raio de alcance do veículo que gerou/possui a mensagem, isto acontece sucessivamente até a mensagem chegar à RSU (mais detalhes sobre a implementação serão abordados na secção 2.2). Ao chegar À RSU, as mensagens são enviadas para um servidor que irá processar os dados recebidos e gerar alertas de segurança (mensagens DENM). Mas como já mencionado, alguns alertas são gerados nos veículos.

Neste protótipo foram definidos 6 tipos de nós: RSU, carro, camião, mota e autocarro, sendo representados pelos valores 1, 2, 3, 4 e 5, respetivamente.

2.1 Alertas de segurança

Neste protótipo foram definidos dois tipos de alertas, um informa sobre o risco de colisão iminente e outro sobre congestionamento de trânsito num determinado local.

O alerta de colisão iminente é gerado pelo próprio veículo, que ao receber as informações sobre os veículos diretamente conectados a si, calcula em quanto tempo pode ocorrer uma colisão. Se esse tempo for inferior a 2 segundos, o veículo envia um alerta para o outro veículo envolvido na possível colisão.

No caso do alerta de congestionamento de trânsito, este é gerado pelo servidor conectado à RSU. Ao se aperceber que existe uma grande concentração de veículos numa determinada área, gera um alerta que deverá ser encaminhado até chegar a uma área próxima da área onde existe o congestionamento de trânsito.

A implementação dos algoritmos que geram estas mensagens será abordado na secção 3.

2.2 Protocolo de encaminhamento

Para os veículos poderem encaminhar mensagens entre si, têm de manter uma base de dados atualizada sobre que outros veículos estão conectados a si, uma vez que os veículos estão sempre em movimento, as ligações entre si estão em constante alteração. Deste modo, os veículos enviam uma mensagem periódica (de 0.5 em 0.5 segundos), com a sua posição e uma *timestamp*, para os seus vizinhos a um salto de distância. Quando um veículo percebe que não recebe estas mensagens de atualização de um vizinho há mais de 1.7 segundos, assume que a ligação foi quebrada e retira-o da base de dados.

O encaminhamento das mensagens CAM é baseado em Geo-Unicast. Sabe-se à priori localização do nó de destino, que para este caso em particular será quase sempre o RSU. Assim, um nó com uma mensagem para encaminhar vai escolher entre os seus vizinhos aquele com menor distância para o destino. A distância é calculada através da fórmula:

$$d = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} \tag{1}$$

Após a seleção do melhor nó, o seu endereço IPv6 é colocado no campo "*next_hop*"da mensagem CAM. A mensagem é enviada em *multicast*, no entanto, os nós ao receberem uma mensagem verificam se o seu próprio endereço corresponde ao endereço presente no campo "*next_hop*", se for armazenam a mensagem para encaminhar.

No caso das mensagens DENM do tipo risco de colisão iminente, estas não exigem um protocolo de encaminhamento porque são trocadas diretamente entre nós vizinhos.

Já nas mensagens DENM do tipo congestionamento de tráfego, o encaminhamento é baseado em Geo-Anycast. A mensagem é reenviada nó a nó até atingir a área de destino. O nó escolhido para o reenvio, é aquele mais próximo da área de destino. Um nó que recebe este tipo de mensagens, verifica se é o próximo nó a quem a mensagem é dirigida, caso seja, verifica se está dentro da área de destino, se estiver envia a mensagem em *multicast*, colocando no campo "*next_hop*"um endereço IPv6 *multicast*, se não estiver dentro da área, reenvia a mensagem para o seu vizinho mais próximo da área de destino.

2.3 Formato das mensagens protocolares

De forma a simplificar, as mensagens trocadas entre os vários nós da rede têm todas formato JSON (*JavaScript Object Notation*), que ao serem enviadas pelo canal, são codificadas numa sequência de bytes.

Foram definidos três tipos de mensagens: mensagem de atualização de conexão, mensagem de dados e mensagem de alerta, estas mensagens são identificadas pelo campo "*type_msg*", com o valor 0, 1 e 2, respetivamente. Estas mensagens também contêm o campo "*origin*", que indica o endereço IPv6 do nó origem e o campo "*node_name*", que indica o nome do nó origem.

Além dos campos mencionados, as mensagens de atualização de conexão contém mais 4 campos:

type_node - indica o tipo de nó, conforme a identificação já mencionada.
pos_x - posição no eixo dos x.
pos_y - posição no eixo dos y.
timestamp.

As mensagens de dados contém os seguintes campos adicionais:

next_hop - endereço IPv6 do próximo nó.
type_node - indica o tipo de nó, conforme a identificação já mencionada.
weight - peso do veículo.
height - altura do veículo.
lenght - comprimento do veículo.
width - largura do veículo.
velocity - velocidade instantânea do veículo.
rain - assume o valor 1 ou 0, dependendo se o sensor deteta ou não chuva.
fog - assume o valor 1 ou 0, dependendo se o sensor deteta ou não nevoeiro.
wet_floor - assume o valor 1 ou 0, dependendo se o sensor deteta ou não o piso molhado.
timestamp.
dest - endereço IPv6 do nó destino.
dest_x - posição no eixo dos x do destino.
dest_y - posição no eixo dos y do destino.

Foram definidos dois tipos de mensagens de alerta, um para cada situação de alerta mencionada. Assim, neste tipo de mensagens existe um campo *"denm_type"* que pode assumir os valores 0 ou 1, dependendo se é um alerta de colisão iminente, ou um alerta de congestionamento de tráfego.

As mensagens de alerta do tipo colisão iminente possuem os seguintes campos adicionais:

dest - endereço IPv6 do destino.

As mensagens de alerta do tipo congestionamento de tráfego possuem os seguintes campos adicionais:

epicenter_x - coordenada x do epicentro do congestionamento.
epicenter_y - coordenada y do epicentro do congestionamento.
radius_s - limite inferior da área alvo do envio da mensagem de congestionamento.
radius_b - limite superior da área alvo do envio da mensagem de congestionamento.
next_hop - endereço IPv6 do próximo nó.
timestamp.

3 Implementação

Para a elaboração deste protótipo foi utilizada a plataforma de emulação CORE (*Common Open Research Emulator*) [1], para a emulação da topologia de rede. As aplicações que correm nos veículos, na RSU e no servidor, foram desenvolvidas em *python* [2].

Nesta secção será abordado todo o processo de implementação deste protótipo.

3.1 Topologia

A figura 1 mostra a topologia de rede utilizada. A rede de *backbone* e a rede infraestruturada utilizam como protocolo de encaminhamento OSPFv3 (*Open Shortest Path First v3*), em que a rede de *backbone* pertence à área 0 e a rede infraestruturada à área 1.

A rede AdHoc comunica com a rede infraestruturada através da RSU.

Os nós móveis (veículos), bem como a RSU são nós MDR [3]. Os nós móveis possuem um *script* de mobilidade (*.scen*) que permite que se movam ao longo do tempo.

3.2 Comunicação entre nós

Cada nó é identificado pelo seu endereço IPv6. Os nós móveis, i.e., os veículos, eles próprios formam uma rede. Como estes nós estão em constante movimento, a comunicação entre os nós tem de ser feita em *multicast*. Cada nó junta-se a um grupo *multicast*, através de uma *socket* UDP, no endereço FF05::4 e porta 3000. O RSU também se junta a este grupo.

A comunicação entre o RSU e o servidor, também é feita através de uma *socket* UDP em *unicast*.

Recorreu-se à utilização da biblioteca *socket*, que permite o acesso à interface de *socket* BSD (*Berkeley Software Distribution*) [4].

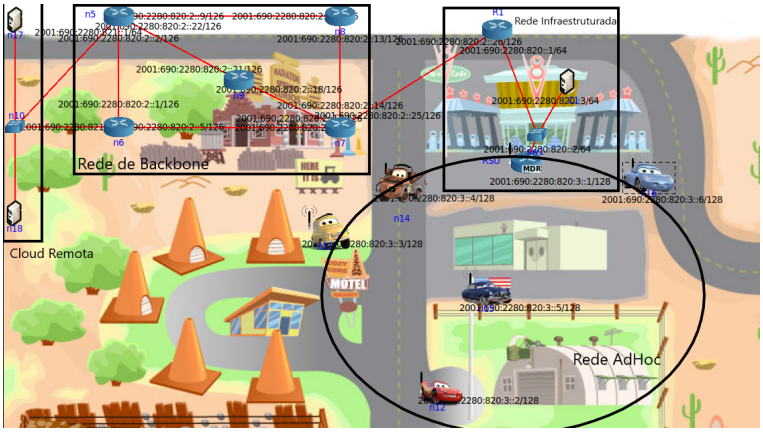

Figura 1. Topologia híbrida de rede AdHoc e rede fixe de infraestrutura.

3.3 Deteção de colisões

Ao receber uma mensagem CAM, o nó verifica se o endereço IPv6 no campo *origin* pertence aos seus vizinhos. Com as informações presentes na mensagem CAM, o veículo que recebe a mensagem pode detetar se vai haver uma colisão entre os dois.

O tempo até os dois veículos colidirem é estimado do seguinte modo:

$$v_{total} = (v_A + v_B)/3.6(m/s) \tag{2}$$

$$t = d/v_{total}(s) \tag{3}$$

É calculada a velocidade total dos dois veículos em (2). De seguida, é calculado o tempo, conforme (3), em que d é a distância entre os veículos, calculada através da equação (1).

Se este tempo for inferior a 2 segundos é gerada uma DENM e enviada para o outro veículo envolvido.

3.4 Deteção de congestionamento de tráfego

O cálculo para a deteção de congestionamento é efetuado no servidor. Esta entidade possui uma visão geral da rede, isto porque recebe mensagens CAM dos veículos da topologia, de forma regular, guarda a posição e outras informações relativas aos veículos.

A figura 2 ajuda a uma melhor compreensão do algoritmo.

circunferência amarela $\frac{R}{3}$.
circunferência vermelha R.
circunferência verde $R \times 3$.

O servidor percorre todos os veículos cuja a informação está presente em memória e verifica quantos carros estão a uma distância R de si, representada na figura pela circunferência vermelha. Se nessa área estiverem presentes um número de veículos superior ao número máximo definido, assume-se que há congestionamento.

Neste cenário, é enviada, em direção ao epicentro, uma mensagem DENM do tipo *traffic jam* com o objetivo de atingir todos os carros presentes entre a circunferência vermelha e a verde, com o propósito de os alertar dum possível congestionamento. Esta mensagem é armazenada no servidor. Quando atinge o primeiro carro dentro desta área designada, a mensagem é retransmitida em *multicast*.

Este processo é repetido por todos os veículos, como mencionado anteriormente, exceto para os veículos cuja posição está a uma distância inferior a $\frac{R}{3}$, representado pela circunferência laranja, dum outro veículo que já gerou uma mensagem *traffic jam* há menos de 10 segundos, i.e, é o epicentro da área de congestionamento de tráfego.

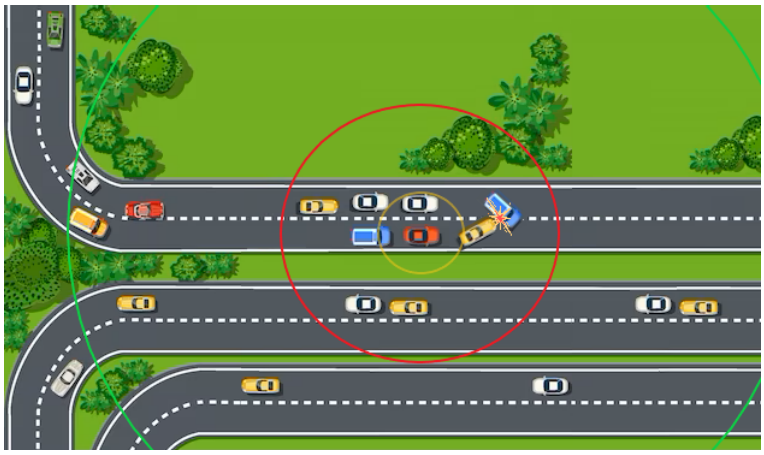

Figura 2. Calculo do congestionamento de tráfego.

Se o servidor não receber uma mensagem CAM de um determinado veículo após 1.7 s, que foi previamente adicionado na sua base de dados, elimina os dados desse veículo da sua base de dados, pois provavelmente já não está na proximidade da RSU a si conectado. Isto evita que o servidor gere falsos alertas de congestionamento de tráfego e também que processe dados de veículos que já não estão na área de cobertura.

4 Testes e resultados

Nesta secção serão demonstrados os testes efetuados para comprovar o funcionamento do protótipo desenvolvido.

Os testes foram efetuados em baixa escala, ou seja, com poucos nós veículos, isto porque a sobrecarga é grande no simulador.

O primeiro teste efetuado, visa mostrar que o servidor recebe as mensagens CAM dos vários veículos, mesmo dos que não estão diretamente conectados à RSU. A figura 3 mostra o servidor a receber dados dos nós n12, n13, n14 e n15. Isto também comprova que o protocolo de encaminhamento está a funcionar.

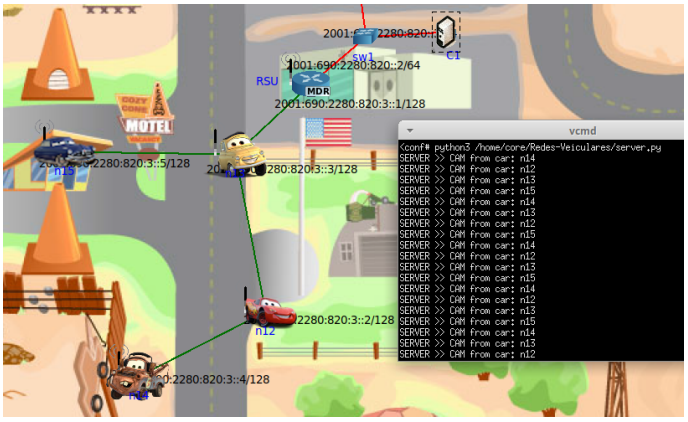

Figura 3. Receção de mensagens CAM no servidor.

De seguida, testou-se se os veículos recebiam mensagens de alerta do tipo risco de colisão. Para tal colocou-se os nós n12 e n14, a uma velocidade de 69 km/h e 73 km/h , respetivamente, a uma distância de 43 m. Neste caso estão em risco de colisão e portanto, os nós envolvidos recebem o alerta, conforme podemos observar na figura 4.

Figura 4. Receção de mensagens de alerta do tipo risco de colisão nos nós n12 e n14.

Por último, testou-se se os veículos recebiam mensagens de alerta do tipo congestionamento de tráfego. Para este teste, assumiu-se que bastava dois carros estarem bastante próximos (menos de 50 m) para haver congestionamento de tráfego.

Num primeiro teste, não se colocou nenhum veículo perto do local de congestionamento. Apesar do servidor gerar o alerta, é suposto que nenhum veículo receba o alerta, pois nenhum veículo está na área de interesse (conforme já explicado em secções anteriores). Na figura 5 pode-se observar isso. O nó n12 não recebe o alerta pois está longe do epicentro do congestionamento de tráfego.

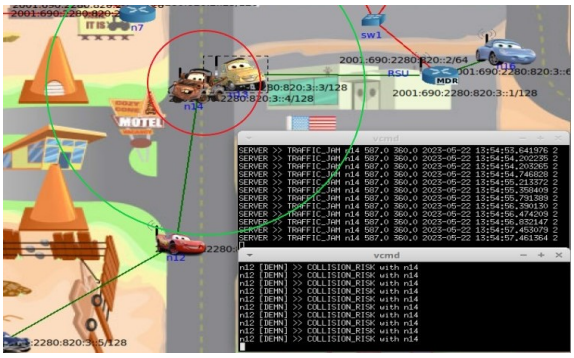

Figura 5. Envio de alerta do tipo congestionamento de tráfego, sem veículos na área de interesse.

Noutro teste, colocou-se um veículo afastado do epicentro do congestionamento de tráfego, mas dentro da área de interesse, neste caso, o veículo vai receber o alerta. Conforme pode-se observar na figura 6.

Deste modo, pode-se comprovar que o protótipo desenvolvido está operacional, ainda que possam existir algumas melhorias.

5 Conclusões

Em suma, este trabalho ajudou a consolidar as matérias abrangidas nas aulas, permitindo que o conhecimento adquirido se prolongasse para um meio mais prático, o que ajudou na realização deste projeto.

Apesar de não terem sido alcançados todos os objetivos, não por falta de motivação e interesse mas sim por falta de tempo, o grupo considera que este projeto foi realizado com sucesso e com aproveitamento positivo.

6 Trabalho futuro

Na sequência deste protótipo desenvolvido, existem várias direções futuras que poderiam ser exploradas:

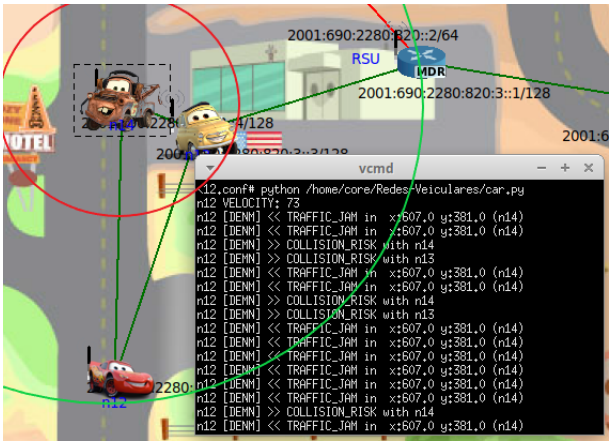

Figura 6. Envio de alerta do tipo congestionamento de tráfego, com veículos na área de interesse.

- Neste protótipo apenas foi utilizada uma RSU. No entanto, seria interessante expandir a rede para incluir mais RSU's, o que permitiria uma maior cobertura da estrada e, consequentemente, comunicações mais eficientes.
- Apesar do protocolo de encaminhamento desenvolvido ser eficaz, este pode ser melhorado. Por exemplo, não escolher um nó de cada vez para encaminhar dados, mas escolher vários, de modo a aumentar a probabilidade de chegada da mensagem ao destino.
- Embora foram desenvolvidos dois alertas de segurança para promover a segurança rodoviária, seria benéfico desenvolver mais alertas de segurança.
- O alerta do tipo risco de colisão iminente, podia ser implementado de forma mais refinada, no sentido em que podia levar em consideração a direção dos veículos. Por exemplo, dois veículos a viajarem a 50 km/h a uma distância de 10 m, vão embater em 0.36 s, segundo este protótipo. No entanto, na realidade isso só seria verdade se os dois carros viajassem em direções opostas. Portanto, a geração deste tipo de alerta devia levar isso em consideração para ser mais preciso.
- No algoritmo de deteção de congestionamento não é levado em consideração a presença de veículos em faixas adjacentes em sentidos opostos, ou seja, os veículos que viajam nessas faixas serão contabilizados e não deveriam. Também não é levado em conta o número de faixas disponíveis, isto é, num cenário em que uma estrada tem várias faixas, o número de carros dentro do raio definido é superior, logo, o numero de carros necessários para alertar para congestionamento devia variar consoante o numero de faixas. Por fim, seria importante também ter em consideração a velocidade a que os carros se deslocam, mas como neste protótipo a velocidade é atribuída aleatoriamente, seria pouco pratico aplicar neste contexto.
- Os veículos respondem aos alertas de segurança de modo passivo, ou seja, o condutor apenas é informado, no entanto, seria útil que os veículos respondessem de forma ativa. Por exemplo, ao receber um alerta de colisão iminente, o veículo decidia, sem a intervenção do condutor, diminuir a velocidade.

Assim, estas são algumas direções futuras a ser tomadas para criar uma rede veicular mais segura e eficiente.

Referências

1. Common open research emulator. Acedido em: 08-05-2023. [Online]. Available: https://www.nrl.navy.mil/Our-Work/Areas-of-Research/Information-Technology/NCS/CORE/.
2. P. S. Foundation. Python. Acedido em: 08-05-2023. [Online]. Available: https://www.python.org/
3. coreemu. Core nodes. Acedido em: 24-05-2023. [Online]. Available: http://coreemu.github.io/core/gui.html#core-nodes
4. P. S. Foundation. socket — low-level networking interface. Acedido em: 08-05-2023. [Online]. Available: https://docs.python.org/3/library/socket.html