Universidade Federal do Paraná

Renan Domingos Merlin Greca

TruMan: Trust Management for Vehicular Networks

Curitiba-PR
2018

Renan Domingos Merlin Greca

TruMan: Trust Management for Vehicular Networks

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Luiz Carlos Pessoa Albini.

Curitiba-PR
2018

Ficha catalográfica

Substituir o arquivo 0-iniciais/catalografica.pdf (PDF em formato A4) pela ficha catalográfica fornecida pela Biblioteca da UFPR a pedido da secretaria do PPGInf/UFPR.

O conteúdo exato da ficha catalográfica é preparado pela Biblioteca Central da UFPR, a pedido da secretaria do PPGINF. Portanto, não "invente" um conteúdo para ela.

ATENÇÃO: por exigência da Biblioteca da UFPR, esta ficha deve ficar no verso da folha de rosto (que contém o nome do orientador e área de concentração). Cuide desse detalhe quando imprimir as cópias finais.

MINISTÉRIO DA EDUCAÇÃO
SETOR CIÊNCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **RENAN DOMINGOS MERLIN GRECA** intitulada: **TruMan: Trust Management for Vehicular Networks**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 21 de Maio de 2018.

LUIZ CARLOS PESSOA ALBINI
Presidente da Banca Examinadora (UFPR)

ANDRÉ RICARDO ABED GRÉGIO
Avaliador Interno (UFPR)

KEIKO VERONICA ONO FONSECA
Avaliador Externo (UTFPR)

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação em Informática

To my parents, who have stood by me all the way through.

Acknowledgments

I would like to express my deep gratitude toward my advisor, Professor Luiz Albini, for all the guidance he offered throughout the research that led to this dissertation. His frequent ideas and suggestions were what inspired me to pursue knowledge and be encouraged to find solutions to problems that are far from simple.

Furthermore, I would like to thank Professor Eduardo Todt, who has helped me in ways beyond this dissertation, and my fellow students, Eric, Nelson and Ivan, who always brought interesting conversations.

I would also like to thank my friends, in particular Taiane, Cainã, Douglas, Lucas and Giancarlo for bringing joy to my life for many years.

Perhaps most importantly, I would like to thank my parents, Edison and Lizmari, who supported me throughout the whole process and always reminded me that I was on a path worth pursuing, as well as my grandmother Glacial and everyone in my family.

Resumo

À medida em que computadores tornam-se menores e mais poderosos, a possibilidade de integrá-los a objetos do cotidiano é cada vez mais interessante. Ao integrar processadores e unidades de comunicação sem fio a veículos, é possível criar uma rede veicular ad-hoc (VANET), na qual carros compartilham dados entre si para cooperar e criar ruas mais seguras e eficientes. Uma solução descentralizada ad-hoc, que não depende de infraestrutura pré-existente, conexão com a internet ou disponibilidade de servidores, é preferida para que a latência de entrega de mensagens seja a mais curta possível em situações críticas. No entanto, assim como é o caso de muitas novas tecnologias, VANETs serão um alvo de ataques realizados por usuários maliciosos, que podem obter benefícios ao afetar condições de trânsito. Para evitar tais ataques, uma importante característica para redes veiculares é o gerenciamento de confiança, permitindo que nós filtrem mensagens recebidas de acordo com valores de confiança previamente estabelecidos e designados a outros nós. Para gerar esses valores de confiança, nós usam informações adquiridas de interações passadas; nós que frequentemente compartilham dados falsos ou irrelevantes terão valores de confiança mais baixos do que os que aparentam ser confiáveis. Este trabalho introduz TruMan, um modelo de gerenciamento de confiança para redes veiculares no contexto de trajetos diários, utilizando o *Working Day Movement Model* como base para a mobilidade de nós. Este modelo de movimentação permite a comparação entre VANETs e redes sociais tradicionais, pois é possível observar que pares de veículos podem se encontrar mais de uma vez em diversos cenários: por exemplo, eles podem pertencer a vizinhos ou colegas de trabalho, ou apenas tomar rotas similares diariamente. Através de repetidos encontros, uma relação de confiança pode ser desenvolvida entre um par de nós. O valor de confiança resultante pode também ser usado para auxiliar outros nós que podem não ter uma relação desenvolvida entre si. O TruMan é baseado em um algoritmo já existente, que é desenvolvido para redes centralizadas e focado em modelos ad-hoc estáticos; seus conceitos são adaptados para servir uma rede descentralizada e dinâmica, que é o caso de VANETs. Usando valores de confiança formados por interações entre nós, um grafo de confiança é modelado; suas arestas representam as relações de confiança entre pares de nós. Então, componentes fortemente conexos do grafo são formados, de forma que cada nó em um componente confie nos outros nós do mesmo componente direta ou indiretamente. Um algoritmo de coloração de grafo é usado no grafo de componentes resultantes e, usando os resultados de coloração, é possível inferir quais nós são considerados maliciosos pelo consenso da rede. TruMan é rápido, colocando pouca carga nos computadores dos veículos, e satisfaz a maioria das propriedades desejáveis para modelos de gerenciamento de confiança veiculares.

Palavras-chave: redes veiculares, gerenciamento de confiança, identificação de nós maliciosos.

Abstract

As computers become small and powerful, the possibility of integrating them into everyday objects is ever more appealing. By integrating processors and wireless communication units into vehicles, it is possible to create a vehicular ad-hoc network (VANET), in which cars share data amongst themselves in order to cooperate and make roads safer and more efficient. A decentralized ad-hoc solution, which doesn't rely on previously existing infrastructure, Internet connection or server availability, is preferred so the message delivery latency is as short as possible in the case of life-critical situations. However, as is the case with most new technologies, VANETs might be a prime target for attacks performed by malicious users, who may benefit from affecting traffic conditions. In order to avoid such attacks, one important feature for vehicular networks is trust management, which allows nodes to filter incoming messages according to previously established trust values assigned to other nodes. To generate these trust values, nodes use information acquired from past interactions; nodes which frequently share false or irrelevant data will have lower trust values than the ones which appear to be reliable. This work introduces TruMan, a trust management model for vehicular networks in the context of daily commutes, utilizing the Working Day Movement Model as a basis for node mobility. This movement model allows the comparison of VANETs to traditional social networks, as it can be observed that pairs of vehicles are likely to meet more than once in several scenarios: for example, they can belong to neighbors or work colleagues, or simply take similar routes every day. Through these repeated encounters, a trust relationship can be developed between a pair of nodes. The resulting trust value can also be used to aid other nodes which might not have a developed relationship with each other. TruMan is based on a previously existing algorithm, which was developed for centralized networks and focused on static ad-hoc models; its concepts were adapted to serve a decentralized and dynamic network, which is the case of VANETs. Using trust values formed by node interactions, a trust graph is modeled; its edges represent trust relationships between pairs of nodes. Then, strongly connected components are formed so that each node in each component trusts other nodes in the same component directly or indirectly. A graph coloring algorithm is used on the resulting components graph and, using the coloring results, it is possible to infer which nodes are considered malicious by the consensus of the network. TruMan is fast, so it incurs low pressure on on-board computers, and is able to satisfy most desired properties for vehicular trust management models.

Keywords: vehicular networks, trust management, malicious node identification.

Contents

List of Figures

List of Acronyms

DTN	Delay-Tolerant Network
GPS	Global Positioning System
LTE	Long-term Evolution
MANET	Mobile Ad-hoc Network
WDM	Working Day Movement Model
OBU	On-Board Unit
RSU	Road-Side Unit
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VANET	Vehicular Ad-hoc Network
WAVE	Wireless Access in Vehicular Environments

List of Symbols

δ	Network density (see subsection 4.3.1).
ρ	Transmission range.
η	Number of nodes in a network.
α	Area of a simulation.
π	The constant pi.

Resumo Extendido em Português

Introdução

Dentro dos próximos anos, uma grande parte de novos veículos virão equipados com funcionalidades de comunicação. Essas funcionalidades permitirão o compartilhamento de dados com outros dispositivos e podem ser ferramentas importantes para reduzir o trânsito e o risco de acidentes. Acidentes de trânsito são uma das maiores causas de morte no mundo [World Health Organization, 2015], tornando necessárias soluções para melhorar a segurança nas ruas.

O compartilhamento rápido de dados entre veículos permite, por exemplo, que veículos inteligentes alertem seus motoristas sobre condições de trânsito [Lee et al., 2004] e que veículos autônomos formem pelotões [Amoozadeh et al., 2015].

Assim, surge a necessidade de redes veiculares *ad-hoc* (*vehicular ad-hoc networks*, ou VANETs), nas quais veículos são os nós ou membros da rede e compartilham informações entre si, sem depender da internet ou de infraestrutura. Porém, como é o caso em muitas tecnologias novas, VANETs podem ser um alvo de ataques de usuários maliciosos. Um usuário malicioso local pode alterar dados para manipular o trânsito, enquanto atacantes remotos podem invadir veículos para controlar a rede [Garip et al., 2015]. Ataques podem ser apenas inconveniências até ameaças à vida, portanto é importante que redes veiculares estejam preparadas para mitigá-los.

Em redes *ad-hoc*, uma forma de mitigar certos ataques é usando dados coletados previamente para filtrar mensagens que aparentam ser maliciosas, incorretas ou irrelevantes. Para tal, emprega-se o conceito de *confiança*. Ao receber mensagens de outros nós, um membro da rede pode construir uma relação de confiança com os outros. Caso o valor de confiança da origem de uma mensagem seja muito baixo, essa mensagem pode ser considerada não confiável. Além disso, essas relações de confiança podem ser propagadas pela rede, permitindo que nós que ainda não formaram suas próprias opiniões possam se beneficiar da informação que já circula pela rede.

Hoje, muitos veículos já vêm equipados com o hardware necessário para processamento e comunicação veicular. É esperado que, até 2022, a maioria dos veículos comuns também venham com tais funcionalidades [Viereckl et al., 2016]. Redes veiculares *ad-hoc* (VANETs) são uma aplicação muito estudada quando se trata de veículos inteligentes ou autônomos. Nelas, todos os nós são relacionados ao trânsito, como veículos ou unidades posicionadas em infraestrutura ao lado das ruas.

O padrão de comunicação mais usado para redes veiculares é o IEEE 802.11p, que descreve dois tipos de membros (ou nós) para redes veiculares: unidades a bordo (*on-board units* ou OBUs) e unidades de beira de estrada (*roadside units* ou RSUs). Comunicação entre pares de OBUs é denominada *vehicle-to-vehicle* (V2V), enquanto comunicação entre OBUs e RSUs é chamada de *vehicle-to-infrastructure* (V2I). Este estudo aborda apenas casos V2V e, portanto, refere-se apenas a veículos como membros de uma rede veicular.

Como é esperado para novas tecnologias, redes veiculares podem se tornar um alvo relevante para usuários maliciosos e atacantes. Alguns exemplos de potenciais problemas são: módulos e sensores, como GPS e velocímetro, defeituosos, inibindo aplicações de segurança ou eficiência [Isaac et al., 2010]; veículos intencionalmente transmitindo dados falsos [Golle et al., 2004]; atacantes remotos controlando múltiplos veículos para congestionar a rede [Garip et al., 2015]; invasão de privacidade ao tentar decifrar e ler mensagens alheias [Isaac et al., 2010]; disrupção de sinal para impedir a comunicação de outros veículos [Isaac et al., 2010].

Como em outros tipos de rede, VANETs dependem de membros que se comportam de maneira correta e previsível e informações incorretas comprometem a utilidade da rede. Há uma distinção importante a ser feita entre nós maliciosos e defeituosos, porém, em termos de confiabilidade, é possível tratá-los da mesma forma, pois, afinal, a principal característica de ambos é a transmissão de informação incorreta.

Em geral, soluções de gerenciamento de confiança são divididos em dois tipos: os que usam *confiança orientada a entidade*, nos quais confiança é relacionada a membros da rede e leva-se em consideração quem transmitiu certa mensagem, ou *confiança orientada a dados*, nos quais o conteúdo da mensagem é mais importante do que quem a transmitiu. Existem também algumas soluções que combinam ambos métodos.

Algumas das propriedades únicas de redes veiculares, que afetam soluções de confiança para elas, são: topologia que muda constantemente e rapidamente; mobilidade de nós restringidas às ruas disponíveis; fragmentação, quando duas ou mais partes da rede estão distantes demais para se comunicar; comunicação pouco confiável com nós distantes; nenhuma restrição notável de energia, quando comparadas a redes de dispositivos móveis; densidade potencialmente muito alta; topologia suscetível a comportamentos erráticos de motoristas.

Em [Zhang, 2011], oito propriedades desejáveis para modelos de confiança para redes veiculares são apresentadas: construção de confiança descentralizada; lidar bem com baixas densidades; dinâmicas relacionadas a local, tempo, eventos e tarefas; escalabilidade; medida de certeza integrada; segurança a nível de sistema; sensibilidade a privacidade; e robustez.

Este trabalho propõe um novo modelo de confiança, TruMan, para gerenciar relações de confiança em uma rede veicular. Usando o modelo proposto, nós de uma rede veicular podem rapidamente identificar quais outros nós são dignos de confiança ou não. Como redes veiculares são altamente dinâmicas, nós adquirem mais informações à medida do tempo e podem se beneficiar de propriedades sociais de VANETs para construir relações fortes com outros nós encontrados frequentemente.

O modelo TruMan é baseado em outro já existente, chamado MaNI [Vernize et al., 2015], que era restrito para redes estáticas. Utilizando algoritmos de grafos, TruMan demonstra-se uma solução eficiente para o problema de gerenciamento de confiança em redes veiculares.

As próximas seções são as seguintes. A **Revisão Bibliográfica** mostra estudos relevantes na área de confiança para redes veiculares. Em **Projeto e Implementação do TruMan**, os objetivos e hipóteses do TruMan são apresentados, além das explicações dos algoritmos que compõe o modelo. A seguir, **Avaliação do TruMan** mostra as ferramentas usadas para validar o TruMan e os resultados dos experimentos realizados. Por fim, a **Conclusão** contém os pensamentos finais sobre o projeto.

Revisão Bibliográfica

Muitas soluções para confiança em redes veiculares foram propostas ao longo dos anos, como [Patwardhan et al., 2006], [Gerlach, 2007], [Raya et al., 2008], [Huang et al., 2010], [Ding et al., 2013], [Haddadou et al., 2013], [Liu et al., 2016], [Kerrache et al., 2016]. Além disso, alguns trabalhos oferecem revisões sobre propostas já apresentadas, como [Zhang, 2011], [Ma et al., 2011], [Zhang, 2012], Mejri et al. [2014], [Soleymani et al., 2015], [Sengar, 2016], [Dwivedi and Dubey, 2016]. Nesta seção, alguns dos trabalhos mais relevantes são apresentados.

No modelo proposto em [Minhas et al., 2010] usa diversos critérios para julgar se uma mensagem é confiável ou não. Ele utiliza uma combinação de confiança baseada em função (por exemplo, viaturas policiais são automaticamente mais confiáveis) e confiança baseada em experiência (baseada em interações anteriores). Além disso, uma mensagem é considerada mais confiáveis quando sua origem estava próximo do evento sendo relatado por ela. Quando múltiplas mensagens sobre o mesmo evento são recebidas, um nó pode optar por considerar as que foram enviadas por nós mais confiáveis, ou ponderar um consenso baseado em diversas opiniões alheias. Porém, este modelo depende apenas de interações diretas, e confiança não é propagada pela rede.

Em [Chen et al., 2010], os autores propõem avaliar mensagens com um método que utiliza grupos. Nós são separados em grupos e, cada vez que um deseja enviar uma mensagem, os outros membros do grupo oferecem suas opiniões sobre o emissor. Finalmente, um dos nós, designado como líder do grupo, coleta as opiniões e decide se a mensagem é válida de acordo com o consenso. Porém, é incerto como o modelo funcionaria em redes esparsas, manter grupos em uma rede altamente dinâmica pode ser uma tarefa de alto custo e um grupo todo pode ser comprometido se o líder não for confiável.

O modelo ART [Li and Song, 2016] busca um modelo robusto e resistente a ataques. Ele tem dois passos principais: coleta de dados e detecção de nós maliciosos. Utiliza a teoria de evidências Dempster-Shafter para agregar dados vindos de outros nós. Então, usa uma métrica baseada em cosseno para comparar vetores de confiança de dois nós (cada vetor é uma sequência de opiniões que um nó tem sobre outros). Nós com vetores de confiança próximos confiam uns nos outros. O problema dessa solução é a dependência em cálculos custosos que podem atrapalhar o desempenho em situações que exigem baixa latência.

Os autores de [Chen and Wang, 2017] propõem uma solução de confiança baseada em nuvem, que exige um gerenciamento de confiança via internet. A vantagem disso é simplificar diversas dificuldades de redes veiculares, como redes esparsas e altamente dinâmicas. Contudo, o modelo é problemático em regiões com pouco ou nenhum sinal de comunicação celular e o sistema todo é suscetível a instabilidades no serviço.

Por fim, é importante notar que nenhum dos trabalhos acima oferece análises de custo e complexidade de seus algoritmos. Portanto, manter uma baixa complexidade é um objetivo chave do modelo TruMan.

Projeto e Implementação do TruMan

TruMan é um modelo de gerenciamento de confiança para redes veiculares, possibilitando a detecção de nós maliciosos em uma rede e a disseminação dados de confiança para outros nós. TruMan busca gerenciamento de confiança eficiente em redes altamente dinâmicas, mantendo baixo custo computacional e um modelo simples de entender e implementar. Esta seção apresenta os fundamentos e algoritmos por trás de TruMan, assim como detalhes de sua implementação.

TruMan é baseado no algoritmo MaNI [Vernize, 2013], que sugeriu o uso de componentes fortemente conexos e de coloração de grafos para a detecção de nós maliciosos em uma rede. Porém, o MaNI foi desenvolvido para redes estáticas e é executado por um agente externo à rede, tornando-se inapropriado para redes veiculares. Para funcionar em redes dinâmicas, TruMan roda iterativamente em intervalos pré-determinados. Além disso, o algoritmo roda de forma descentralizada, com uma instância rodando em cada membro da rede.

Cada nó u armazena um grafo direcionado de confiança $T = (V, E)$ que é uma abstração da rede real e começa apenas com $V = u$. Cada nó em V representa um membro da rede e cada aresta em E representa uma relação de confiança entre dois nós. Como cada nó armazena sua própria representação da rede e essa representação evolui com o tempo, há um $T_i^u = (V_i^u, E_i^u)$ para cada nó u e iteração i.

No começo de cada iteração, nós coletam informações sobre seus vizinhos. Um pré-requisito deste passo é a existência de um teste que classifica um nó adjacente como benigno ou malicioso. Tal teste é um problema grande por si próprio, e sai do escopo deste trabalho. Estudos sobre isso podem ser encontrados em [Golle et al., 2004], [Li et al., 2016], [Kerrache et al., 2016].

Cada vez que um nó vizinho v é identificado como benigno, o valor de confiança armazenado em $u \to v$ aumenta, e o grafo $T_i - 1^v$ é unido com o grafo armazenado por u. Após coletar informações de todos os seus vizinhos naquele instante, um novo grafo T_i^u é formado, que é utilizado para os próximos passos.

Em seguida, T_i^u é separado em componentes fortemente conexos usando o algoritmo de Tarjan [Tarjan, 1972], de forma que cada par de nós em um componente é conectado por um caminho de confiança. Ou seja, todos os nós de um mesmo componente confiam uns nos outros direta ou indiretamente. Portanto, em termos de confiança, nós dentro de um mesmo componente podem ser considerados como um só: se um deles é confiável, pode-se assumir que todos são. Os componentes tornam-se nós do grafo $C_i^u = (V_i'^u, E_i'^u)$.

O algoritmo de coloração de grafos [Mittal et al., 2011] é usado como heurística para classificar nós como benignos ou não. Após a execução do algoritmo, a cor cujos nós em C_i^u representam a maior quantidade de nós em T_i^u é classificada como correta, e as outras cores são classificadas como incorretas. Isto funciona porque, em uma rede na qual a maior parte dos nós são benignos, estes tendem a formar poucos componentes grandes, enquanto os nós maliciosos pertencem a componentes pequenos. Assume-se que a maior parte dos nós seja sempre benigna – caso contrário, a rede como um todo está comprometida e perde completamente sua função.

A complexidade do algoritmo pode ser calculada ao somar as operações mais custosas. Para cada iteração i e nó u, e sendo n o número de vizinhos de u, o cálculo é o seguinte:

$$TruMan = Tarjan + Coloração + (União \times (n_i^u))$$

Como discutido acima, o algoritmo de Tarjan tem complexidade de $O(|V| + |E|)$ para o grafo de confiança T. Já o algoritmo de coloração tem complexidade de $O(|E'|)$ para o grafo de componentes C. A parte mais custosa do algoritmo é a união de grafos que acontece após a comunicação entre nós confiáveis. A complexidade desse processo é de $O(|E|)$ para vizinho que um nó tem em uma determinada iteração; o número de vizinhos é, no máximo, $|V|$. A complexidade total do Truman é, portanto:

$$O(|V| + |E|) + O(|E'|) + O(|V| \times |E|)$$

Porém, $|E'| \leq |E|$ é sempre verdade, porque o grafo C é uma redução do grafo T. Além disso, $|V| + |E| \leq |V| \times |E|$ também é verdade, com a exceção do cenário irrelevante no qual $|V| \leq 1$ ou $|E| \leq 1$. Portanto, a complexidade do algoritmo TruMan pode ser simplificada como:

$$O(|V| \times |E|)$$

Avaliação do TruMan

Para testar o TruMan, uma implementação do algoritmo foi feita usando Python. Grafos com mobilidade de nós foram gerados no simulador ONE [Keränen et al., 2009], usando o *Working Day Movement Model* [Ekman et al., 2008] para providenciar mobilidade próxima ao do mundo real, e um mapa de parte da cidade de Helsinki, Finlândia. Imagens da topologia da rede foram salvas a cada 10 segundos simulados, e essas imagens foram usadas como entrada para o algoritmo TruMan. O comportamento de nós maliciosos é aleatorizar suas opiniões sobre seus vizinhos.

A maioria dos parâmetros da simulação no ONE foram tiradas do artigo do Working Day Movement Model [Ekman et al., 2008]. Alguns parâmetros diferentes foram usados, mostrados na Tabela 4.1. Todos os nós da simulação são carros; para o propósito deste trabalho, nenhum outro tipo de veículo foi considerado. Uma parte pequena dos nós movimenta-se aleatoriamente, para simular veículos que não seguem padrões de movimento diários.

O raio de transmissão dos nós varia de 10m a 50m, ilustrando as diferenças entre diferentes densidades de rede. A densidade de rede (δ) é um valor que abstrai o volume e a frequência de conexões em redes veiculares, estimando a cobertura da rede pelo ambiente. Para o TruMan, densidades mais altas trazem melhores resultados, pois nós podem adquirir informações mais rapidamente. A densidade é calculada usando o raio de transmissão (ρ), o número de nós (η) e a área da simulação (α).

A cobertura de um único nó é o círculo ao redor dele formado pelo raio de transmissão. Esse valor é dividido por dois para compensar círculos sobrepostos, e então multiplicado pelo número de nós na rede. Por fim, esse valor é dividido pela área total do ambiente. A fórmula de densidade da rede é a seguinte:

$$\delta = \frac{\frac{\rho^2\pi}{2} \times \eta}{\alpha}$$

As densidades de algumas simulações realizadas são exibidas na Tabela 4.2. Já a Tabela 4.3 mostra as densidades de rede hipotéticas de algumas cidades do mundo, usando dados geográficos reais e supondo um raio de transmissão de apenas 10m. É possível ver que, mesmo com um raio de transmissão pequeno, as cidades oferecem densidades de rede maiores do que as das simulações, assumindo que uma parcela substancial de seus automóveis seja equipada com dispositivos de conexão.

Para validar o desempenho e a corretude de TruMan, diversas simulações foram executadas.

Nas simulações com 10% dos nós agindo maliciosamente, com raio de comunicação entre 10m e 50m, é possível ver como o aumento do raio de comunicação melhora bastante os resultados: com 10m, quase 8000 iterações são necessárias para atingir um bom resultado, enquanto com 50 são apenas cerca de 1000.

Em simulações com raio de comunicação de apenas 10m e até 30% de nós maliciosos, os resultados são bons. Porém, com 40% de nós maliciosos, uma parte pequena desses nós não são detectados. Já com 50% de maliciosos, os resultados são erráticos, pois o controle da rede é dividido entre os nós benignos e maliciosos.

A maioria das simulações foram feitas com o limiar de confiança $h = 0.5$, que significa que, para um nó confiar em outro, o valor de confiança deve ser acima de 0.5. Simulações com $h = 0.3$ e $h = 0.7$ demonstram o impacto de mudar esse limiar. É possível observar que o impacto não é muito significativo, porém, com $h = 0.7$, os resultados são um pouco melhores.

Quando o algoritmo é executado durante 7 dias simulados, a maioria dos nós maliciosos é detectada ao fim do primeiro dia, e a rede é completamente descoberta pouco tempo depois. Com o tempo, o número de falsos positivos cai, até tornar-se um número insignificante.

Por fim, as simulações nas quais um nó passa a ser malicioso na metade do tempo mostram como TruMan reage a um possível ataque. O parâmetro m determina quantas iterações o algoritmo leva para descartar arestas antigas, e, portanto, afeta a agilidade do modelo ao detectar um nó convertido. Com m muito baixo, informações são descartadas muito rapidamente e o número de falsos positivos aumenta drasticamente.

Conclusão

Nos próximos anos, comunicação veicular será uma importante ferramenta para segurança e eficiência em transportes. Porém, elas também serão um alvo de atacantes e usuários maliciosos. Confiança é um conceito poderoso para evitar a disseminação de dados falsos entre membros de uma rede. Neste projeto, um novo modelo de confiança para redes veiculares, chamado TruMan, foi apresentado. TruMan combina algoritmos eficientes para providenciar gerenciamento de confiança rápido para redes altamente dinâmicas.

Enquanto nós (veículos) viajam pela rede, eles adquirem mais informações sobre outros membros da rede, e podem usar essas informações para detectar nós maliciosos. A utilização de componentes fortemente conexos permite que o grafo da rede seja simplificado em um menor, no qual cada nó é uma abstração de diversos membros da rede. Então, com um algoritmo de coloração de grafo, membros maliciosos se destacam ao ter cores diferentes da maioria dos outros nós.

TruMan foi testado usando dados gerados com o simulador ONE e o Working Day Movement Model. Os experimentos mostram que veículos podem formar uma abstração suficiente da rede para detectar nós maliciosos ao redor deles. Em geral, quanto mais tempo durar a simulação e quanto maior for a densidade de nós na rede, melhores são os resultados.

Em comparação aos trabalhos relacionados, TruMan satisfaz a maior parte das propriedades desejáveis para modelos de mobilidade em redes veiculares, e, ao mesmo tempo, permite que as outras propriedades sejam implementadas de outras formas. O foco do TruMan é eficiência e é o primeiro modelo de confiança para redes veiculares a apresentar claramente a complexidade de seu algoritmo.

Há diversas opções para trabalhos futuros relacionados ao TruMan, como, por exemplo: testes em uma variedade maior de cenários, usando dados reais de mobilidade; aproveitar mais as propriedades sociais de redes veiculares; a integração com veículos de transporte público e infraestrutura; testes de resistência contra ataques conhecidos; testes em redes móveis reais; experimentação em outros tipos de rede.

Chapter 1

Introduction

As computers grow in power and shrink in size, more aspects of everyday life can be enhanced by adding processing units to common devices. While many of these applications focus on conveniences, such as home automation [McCole, 2016] (the collection of connected and smart devices is dubbed the Internet of Things or IoT [Morgan, 2014]), the integration of computers with other objects and devices can also be important to save time and save lives [Real-Time Innovations, 2014]. One way of achieving this is by adding computers and wireless transmitters to vehicles — such as cars, buses, and trains — so they can share data which may increase traffic efficiency or reduce the chance of accidents [Saini et al., 2015].

In 2013, an estimated 1.25 million people lost their lives due to traffic accidents globally [World Health Organization, 2013]. While this number has greatly reduced over the past decades [Johnson, 2010] [Insurance Institute for Highway Safety, 2016] thanks to better safety features (seat belts, air bags, ABS, etc.) and stronger laws (drunk driving, motorcycle helmets, speed limits, etc.), it may still rise as a major cause of death in the years to come [World Health Organization, 2015], so further actions are necessary. Furthermore, as the car population increases, congestions consume even more time of the daily commuter, peaking at over 100 hours per year for the residents of Los Angeles, CA [INRIX, 2017]. Moscow, New York, Bogotá, São Paulo, London and Paris are also among the ten most congested cities in the world [INRIX, 2017].

Smart vehicles and vehicular networks are ways that technology can aid both of the aforementioned problems. Through the use of sensors and wireless communications, these vehicles are able to avoid accidents by alerting distracted drivers [Lee et al., 2004], or by knowing in advance another vehicle's position and speed [Hafner et al., 2011]. By communicating, they can also collaborate to offer driving and route suggestions, therefore reducing the possibility of traffic jams [Knorr et al., 2012].

Today, certain vehicle manufacturers already include the on-board technology required to enable vehicular communications in the real world [IEEE Connected Vehicles, 2015]. However, this technology only becomes truly useful when there are other vehicles or infrastructure with whom to communicate, so, at the moment, the benefits are notable but limited [Cadillac Pressroom, 2017].

When dealing with safety or traffic-efficiency applications, it is crucial that network communications occur with low latency (approximately 100 milliseconds [CAMP Vehicle Safety Communications Consortium, 2005]). Current cellular technology, such as LTE, could be used to connect vehicles to the Internet, but the delay added by the transmission would make safety applications unfeasible or unreliable [Mangel et al., 2010]. Cellular connections also have other problems: the connection would require an active subscription with a carrier; the connection depends on available infrastructure; the wireless frequency would be shared with phones and other mobile devices, increasing the possibility of interference and congestion; server-side issues could impact the vehicles' communications.

For these reasons, ad-hoc solutions are preferred over centralized ones. An ad-hoc network is one that has no reliance on pre-existing infrastructure (such as routers or access points) [Wu and Stojmenovic, 2004]. Instead, each node is able to communicate directly with others and a routing protocol allows for messages to be forwarded until they reach their destinations. Every time a node wants to send a message and the recipient is not a direct neighbor, it must choose which nearby node is the most likely one to get the message to its destination. Routing techniques can use either the network's topology or geographical coordinates [Saini et al., 2015] to choose which node should be the next hop.

These issues — the additional safety and efficiency as well as the low-latency communications — can be tackled through the use of a vehicular ad-hoc network (VANET), in which vehicles share data amongst themselves without relying on external devices, an Internet connection or server availability. Neighboring vehicles can share their position and velocity data at high frequencies, allowing, for example, for autonomous vehicles to plan a platooning approach to traffic [Amoozadeh et al., 2015]. In the case of a collision or other event, nearby nodes can broadcast alerts, which other nodes pick up and forward [Li and Wang, 2007], as illustrated by Figure 1.1. That way, an alert can travel long distances in little time, allowing approaching vehicles to safely slow down or pick alternative routes.

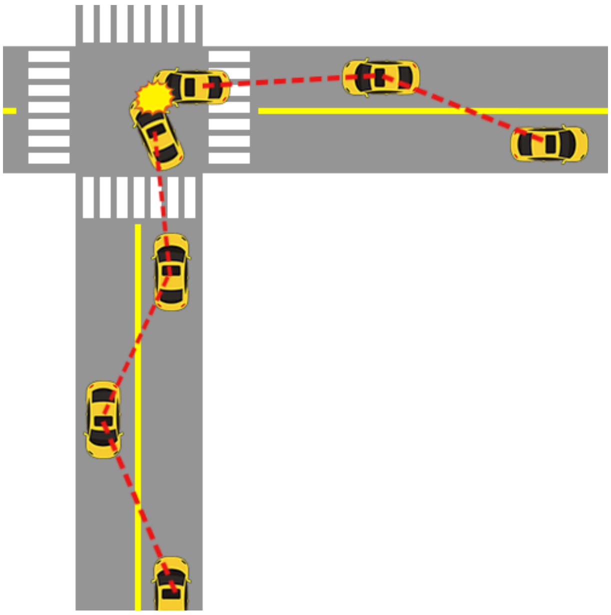


Figure 1.1: Propagation of a collision alert in a VANET

As is the case with most new technologies, VANETs are expected to be a notable target of attacks for a diversity of reasons [Isaac et al., 2010]. A local malicious user might alter the data his or her vehicle broadcasts in order to manipulate traffic conditions, while remote attackers could invade vehicles' computers and obtain partial control of the network [Garip et al., 2015]. These attacks can vary from time-consuming annoyances to life-threatening, so it is important that real-world implementations of vehicular networks are prepared to handle them.

In ad-hoc networks, one way to mitigate a number of attacks is through each node using data collected from previous experiences to filter out incoming messages that seem to be malicious, incorrect, or irrelevant. A node's degree of confidence that some data is correct and useful is called *trust*. For instance, in the example of a single malicious user broadcasting false data, nodes receiving these messages can use their own sensors to verify whether or not the data was correct, and update the *trust value* of the sender vehicle. In case the trust value of a sender is too low, a receiver node can choose to ignore the data contained in a message, as it concluded that the sender is not trustworthy. Trust allows for better cooperation of nodes in a network, since incorrect messages might be detected and discarded.

Furthermore, once nodes form their own opinions about others, they can propagate pre-existing trust values when necessary. For example, if two nodes are not direct neighbors and do not have any pre-existing trust information about each other, they can ask intermediary nodes for their opinions on the other node [Wang et al., 2009]. The management of trust values (i.e. how one node acquires and updates trust values) and the use of these values to derive further information (such as designating nodes as malicious or not) is called *trust management* [Ma et al., 2011]. The effective use of trust management allows for the detection of malicious, misbehaving or faulty nodes in the network, and for such information to be shared amongst the benign participants. Throughout this document, trust and trust management may be used interchangeably.

While the detection of incorrect nodes and/or messages is an important aspect of security and safety in vehicular networks, it does not address all of the problems. Trust solutions are not viable without a solid identity verification scheme, for instance, since nodes would not be able to form trust opinions without being sure of the others' identities (these schemes often use a Public Key Infrastructure [Wasef et al., 2010]). They also do not address issues such as driver and passenger privacy when using the facilities of a VANET. Furthermore, cryptography must be used in order to guarantee that the secrecy of messages is not violated. Therefore, trust and trust management must be viewed as an important aspect of vehicular ad-hoc networks, but not as a definitive solutions to all of the related concerns.

In order to study the implications of trust in vehicular networks, it is interesting to first take a closer look at trust in other kinds of networks. VANETs are a subset of technological networks, therefore it is useful to consider how the Internet or other ad-hoc networks handle trust. Furthermore, VANETs contain several features often found in social networks, which can be directly tied to how nodes form and develop trust relationships with each other.

The study of networks is tied to graph theory, since graphs are a useful way to generate models of networks, and therefore many concepts of the two fields overlap. Mathematical, computational and statistical concepts developed for graphs can be translated to be useful for a variety of different types of networks. Therefore, this document utilizes notation from graph theory in order to represent various kinds of networks.

This work introduces TruMan, a trust management solution for vehicular networks. TruMan combines solutions to two classic graph theory problems, strongly connected components and graph coloring, in order to provide an efficient approach to identifying malicious nodes in a dynamic and decentralized network. This is based on a previously existing algorithm, called MaNI [Vernize et al., 2015], which is limited to centralized networks.

In order to function in a dynamic and decentralized environment, TruMan takes advantage of features that allow the development of trust relationships between members of the network. Since there is no unifying agent supervising the network, trust relationships are built over time, as nodes move around the network and meet other nodes.

These features are discovered by tracing analogies with social networks. For example, certain groups of nodes might come into communication range of each other with predictable frequency, such as vehicles which belong to family members, neighbors or work colleagues as their owners perform their daily activities. By considering such features, TruMan allows the formation of strong trust relationships, which in turn facilitate the discovery of malicious members of the network.

The remainder of this document is organized as follows. Chapter 2 explains the broad study of complex networks and the importance of trust in social and technological networks, then goes into details regarding VANETs and the importance of trust solutions in the field, presenting previous studies made on the subject. Chapter 3 describes the goals of TruMan, its underlying hypothesis and the work done to achieve those goals, as well as the previously existing algorithms which form parts of TruMan. Next, Chapter 4 explains the tools used to validate TruMan's functionality and presents the results of the experiments that were performed. Finally, Chapter 5 presents the final thoughts on the project and concludes this document.

Chapter 2

Background and Related Work

In order to develop a new trust management scheme for vehicular networks, a review of the study of networks as a whole is required. This chapter introduces the concept of complex networks, digging into the meaning and importance of trust in the cases of social networks and technological networks. Then, vehicular ad-hoc networks are explained in details, with the current state of the art, unique properties compared to other networks, desired properties for a vehicular network trust model and details on the most relevant works in the subject.

2.1 Complex Networks

Complex networks can describe many systems which are observed in nature and society through a collection of *vertices* (or *nodes*) and *edges* [Newman, 2010]. They can be comprised of palpable components (such as computers and cables), somewhat abstract entities (such as the World Wide Web's collection of webpages and URLs), or both (like the people and relationships that form a social network). Complex networks are generally divided into four categories [Newman, 2010]:

1. **Technological Networks** are grids purposefully engineered to provide services to consumers and/or citizens. The primary examples of these networks are the Internet, the telephone network, power grids, transportation and delivery networks. A commonly studied type of technological network are Mobile Ad-hoc Networks (MANETs). Although not of widespread use, MANETs can provide a way to create a network without pre-existing infrastructure, as long as each device is equipped with the proper hardware and software. Trust issues in technological networks and MANETs are detailed in Section 2.3. VANETs, which are special types of MANETs, are introduced in Section 2.4, along with several details regarding trust in those types of networks.

2. **Social Networks** are formed of relationships between people, or groups of people. These relationships can be familiar, friendships, acquaintance, etc. For the purposes of this work, the most relevant type of relationship is that of trust. The details surrounding trust relationships in social networks are shown in Section 2.2.

3. **Information Networks** are the ones in which nodes are pieces of data or information and the edges are the connections between those pieces. Often, information networks are directly associated with technological or social networks. For instance, while the World Wide Web is an information network (in which the nodes are webpages and the edges are the links that users click on to navigate), it relies on the Internet, as it contains the physical

infrastructure that makes the web possible. Online social networks can also be classified as information networks, since their nodes are actually information about people rather than the people themselves. Trust in information networks can be observed in some instances, like peer-to-peer networks, although its usages are not relevant for this work.

4. **Biological Networks** are the networks found in nature. Their nodes can be chemicals, cells, animals, groups of lifeforms, and more. An example is the brain, which contains a neural network formed by neurons; connections in the network represent signals that are sent from one neuron to another. Another instance of biological networks are food chains, categorized as ecological webs. Species of animals are the nodes, while the predation of other species form the edges. In biological networks, it is difficult to clearly define trust, since nodes may not have any sort of awareness or intelligence (such as cells or proteins). Regardless, the study of trust in biological networks is not relevant for this project.

In most networks, trust can be a useful tool to aid the security and safety of its members. Therefore, the study of the concept and applications of trust is an important part of the study of networks.

Trust is a concept studied in fields such as psychology and economics, with specific definitions. In complex networks, under the perspective of computer science, trust is a measure of how much confidence one member of a network has that another member of the same network will behave properly and provide valid and/or meaningful data [Sherchan et al., 2013]. What follows is the basis of how a network can be modelled using a graph and how a trust model can be applied to it.

Consider an undirected graph $G = (V, E_G)$, which models one complex network of any kind. The vertices are the members of the network (computers, humans, etc.) and each edge represents a pair of vertices' ability to exchange data freely. This graph represents the network's *topology*, that is, the basic structure of the network. Then, there is also a directed graph $T = (V, E)$, called a *trust graph*. T contains the same vertices as G, although its edges represent the degree of trust (or opinion) each node has towards another.

There are two main ways to describe the edges in E: they can be binary, either existing when there is trust or not otherwise, or they can hold a specific trust value within a certain range. In some cases, an edge $a \to b$ with value 0 indicates lack of trust, meaning a has no reason to trust b. In others, it indicates *negative trust*, meaning a has reason to believe b is malicious.

Since G and T represent different types of information, the shape of T is not necessarily similar to that of G, even though they share the same vertices. For instance, two people can have contact with each other (therefore sharing an edge in G) but not maintain a trust relationship (therefore not sharing an edge in T), thus altering the layout of the trust graph compared to the network topology.

Trust can be hard to define in the context of biological networks, as these networks are often formed by members who have no distinctive properties to categorize as trustworthy or not and, when such properties do exist, the network is probably closer to a social network than a biological one. The same applies to information networks, which are formed by pieces of data that do not need to maintain active relationships with each other. Again, an information network that requires trust is likely closer to the category of technological networks.

In the following two sections of this chapter, trust is further explained in the contexts of social and technological networks. Both types of network can be fitted into the model above, but contain distinct features that demand closer examination. Furthermore, features of both are relevant when analyzing network structure and trust graphs for vehicular networks, which are expanded upon in Section 2.4.

2.2 Trust in Social Networks

There are two types of social networks: real-world ones formed by relationships between people, and online ones that attempt to abstract the former into a digital environment. Examples of the first one are all around, present in any family, workplace, school or group of friends [Newman, 2010]. Online social networks started by connecting people who already knew each other and giving them an additional form of interaction (analogous to what telephones and email did before), but, today, it is not unusual for people to form relationships with others whom they have only met online.

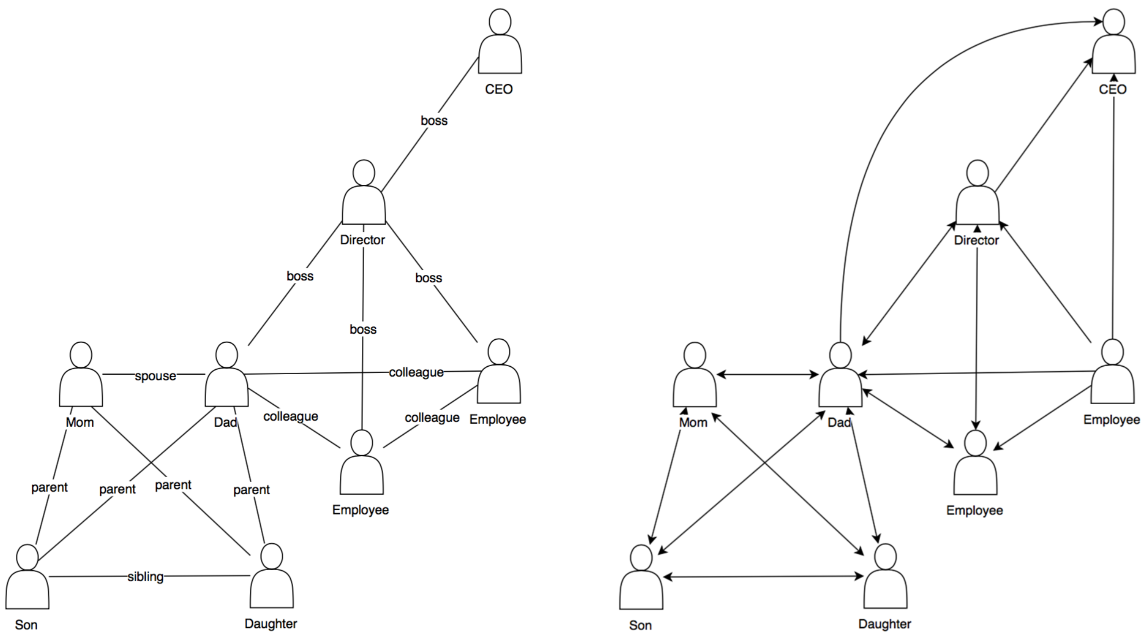
In a traditional social network, it is simple to perceive how trust is relevant and how it works, since trust relationships between people are used on a daily basis to make decisions. When adapted to a digital environment, these social relationships can be used to automatically increase the relevance of certain information. For instance, upon reading an online review for a certain product, a user will be more likely to accept the review's conclusion if it was written by a close friend than if it were written by a stranger. Social trust is a way of estimating how much a certain recommendation will lead to a positive outcome [Golbeck and Hendler, 2006].

The absence of trust or the presence of distrust have consequences as well. Both in the real world and online, information which comes from a stranger is received with uncertainty; there is no reason to trust the sender, so the data itself must be analyzed and compared to other sources in order to judge whether or not it may be trusted. When one person actively distrusts another (that is, the person believes the other is malicious or uninformed), receiving data from the untrustworthy source will be actively avoided. In online social networks, for example, one user can "block" another in order to avoid seeing anything from the other.

Social networks also have the property of carrying trust from one relationship to another: information shared by a close friend of a person might be considered almost as trustworthy as some collected by the person him or herself. Therefore, it is possible to model social trust relationships as a graph, in which nodes represent people and edges represent a certain degree of trust [Newman, 2010]. Expanding on that property, there is the concept friends of friends [Boissevain, 1974]. If, for example, nodes a and b have mutual trust and are considered friends, then it is reasonable to assume that some of a's trust for b carries over to other nodes that enjoy mutual trust with b. In other words, a friend of a friend can be considered more trustworthy than the average stranger. This property is similar, although not identical, to transitivity, since trust is diminished for each extra step an origin node needs to reach a destination, and there is also the possibility that one node distrusts another even if they share a mutual friend. Naturally, social trust is not commutative (a trusting b does not imply that b trusts a).

In general, social networks' topology and trust graphs are mostly static. Although friendships are formed and ended frequently (i.e. the topology is dynamic), those connections do not disrupt the general shape of the network, because members of the network will usually have other friends whose relationships remain stable. Even if a certain person's trust integrity is compromised due to a specific incident, that person's friends are not necessarily deemed untrustworthy, preserving part of the trust graph. While positive trust is often tied to the social topology, it is not always the case: one example is two work colleagues who may have a professional relationship, but wouldn't trust each other on other matters; another is the trust people place in authority figures without necessarily having met. Figure 2.1 shows an example of a small social network containing a family and an office.

In Section 3.2, the argument is made that VANETs can be considered social networks in several occasions and how this can be used to develop trust in vehicular networks.



(a) Topology graph (b) Trust graph

Figure 2.1: Example of a topology graph and a trust graph in a social network.

2.3 Trust in Technological Networks

In conventional technological networks, such as the Internet, trust is defined and applied quite differently from social networks since, generally, it is very centralized through services that offer security to users. Examples of this can be an IP filtering scheme to avoid distributed denial of service (DDoS) attacks or web browser extensions that block requests to domains in a blacklist; a central agent, be it a hosting provider or the extension's publisher, must maintain and update a list of untrustworthy IP addresses or domains. This means that, in the context of the Internet, trust is often derived from a secondary source: end users and their computers can't be expected to maintain their own blacklists, so they rely on external parties which may provide these lists along with other security services. Similarly, when a user visits an e-commerce website, they must have some degree of trust on the website or the vendor; in this case as well, third-party services are used to certify the legitimacy of the transaction, based on feedback from other customers.

While the centralized trust solutions above serve their purpose on the security and privacy of Internet users, they would be too slow to be viable in a dynamic ad-hoc network, which cannot rely on a back-end infrastructure to distribute those lists. The most common instance of mobile ad-hoc networks, or MANETs, is using mobile devices, such as smartphones, being carried by humans. Although these networks are dynamic, their mobility is relatively low in relation to the wireless range of the devices — if two people are walking in opposite directions, their phones may communicate for several seconds before they leave each other's range. Trust solutions for MANETs can use this property to their advantage, since it allows one node to test another and check several of the messages sent between them.

Ad-hoc networks require a decentralized approach to trust management; each member of a network has its own opinions about other members, and these opinions can change over time. For these opinions to be generated and updated, two nodes must have had previous contact with each other, or derive trust from a third, intermediary, node. Hence, there is the correlation between the trust graph and the topology graph of the network. Since MANETs are dynamic, the graph that represents a network's topology is frequently changing and, with that, the opportunities to create and update trust relationships also changes. In networks in which nodes can meet more than once, it might be valuable to store information from previous encounters to use in the future, although this process can be too slow or resource-consuming to be viable in certain devices; by doing this, the trust graph maintains edges between nodes that are no longer connected in the topology.

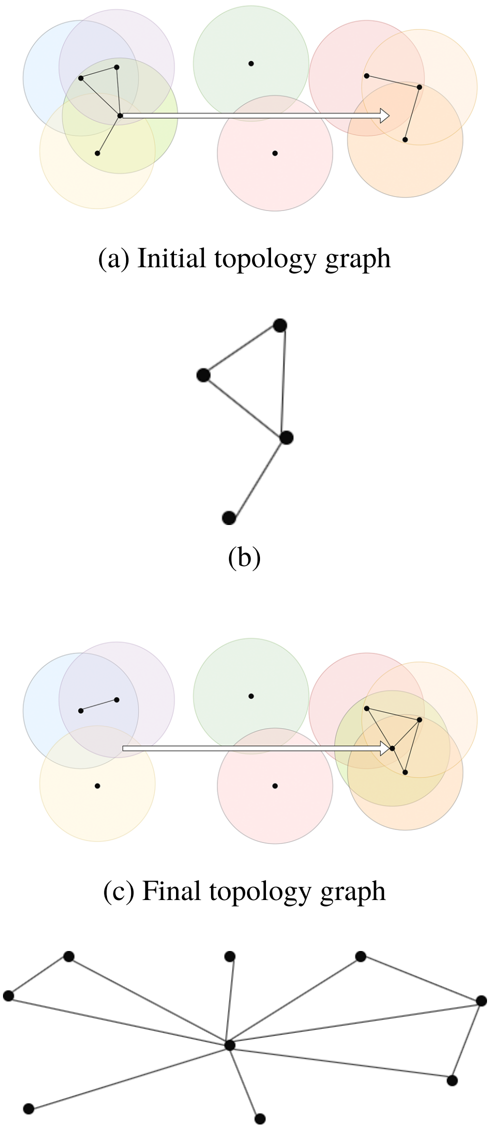
Another important aspect of trust in ad-hoc networks is that information is, generally, uncertain and incomplete [Baras and Jiang, 2005]. That is, since nodes form their own model and opinions of the surrounding network, it is unlikely that this data will always be certain and accurate with reality. For this reason, it is also possible to use data gathered by neighboring nodes to complement the model. Data received from neighbors is also subject to the trust evaluation of whoever requests it, but it is crucial to have better knowledge about other members of the network. Incompleteness is an inherit trace of MANETs, since it is entirely possible for nodes to be too distant to communicate, and only occasionally come into contact.

Finally, MANETs must consider the processing and battery limitations of the devices that integrate it. Nodes may disable wireless communications to save power and therefore become uncooperative, or it may be too slow to be a reliable source of information.

There are few examples of MANETs implemented for consumer devices. Two examples are networks created to quickly share data between devices using Wi-Fi or Bluetooth [Krochmal et al., 2014], or ones that allow for multiplayer gaming sessions amongst multiple nearby devices [Sasaki and Kuwahara, 2011]. In both cases, there is no need for a complicated system-level trust model, since those activities involve active participation from the users wielding the device (that is, the user chooses whether or not to communicate with other devices); rather, the trust relationship occurs socially amongst the users themselves.

Figure 2.2 is an example highlighting the difference between the topology graph and the trust graph. The topology graph changes as one node moves across the network, with edges being added and removed as nodes move into and out of each other's ranges. The trust graph, however, is constantly being built, maintaining past relationships even if the nodes are no longer in communication range.

Naturally, VANETs are an instance of MANETs and therefore share some of the same features. However, the topology of vehicular networks is very different from standard ad-hoc networks, and possible trust solutions are accordingly also distinct.



(a) Initial topology graph

(b)

(c) Final topology graph

(d) Final trust graph

Figure 2.2: Example of the changes node mobility causes to the topology and trust graphs.

2.4 Trust in Vehicular Ad-hoc Networks

Today, most premium vehicles come equipped with hardware that allow for connectivity features; it is expected that, by 2022, many standard vehicles will also come with such features built-in, accounting for a substantial share of the automotive industry's revenue [Viereckl et al., 2016]. Although these features can be useful tools to aid drivers, reducing traffic and risk of accidents, they are merely a gateway to the long-term goal of truly autonomous vehicles, which might become a reality within the next decade; many automakers and technology companies have laid out their plans for the upcoming years [Stewart, 2016]. However, the proper functioning and utility of both connected and fully autonomous vehicles rely on technologies, protocols and applications that allow for the fast communication between vehicle's on-board computers.

Vehicular ad-hoc networks, which are a special instance of MANETs, are a much-studied solution to the problems in the way of smart and autonomous vehicles. In these networks, all nodes are related to traffic; they can be vehicles equipped with on-board computers, or stationary units placed near roads. By quickly sharing data with neighboring vehicles, without the need of an Internet connection, smart vehicles can alert their drivers of important road conditions [Barba et al., 2012], while autonomous vehicles can synchronize their movements to maximize traffic throughput [Amoozadeh et al., 2015].

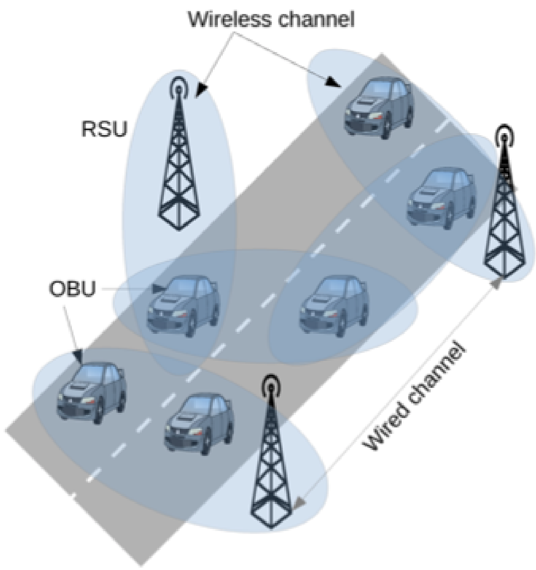


Figure 2.3: Basic elements of a VANET: OBUs and RSUs. [Saini et al., 2015]

Several current efforts to make VANETs viable in cities are centered around the IEEE 802.11p standard, also called Wireless Access in Vehicular Environments (WAVE) [Jiang and Delgrossi, 2008]. Among other aspects of the wireless technology, the WAVE standard describes two types of nodes for vehicular networks: on-board units (OBUs) and road-side units (RSUs). On-board units are computers placed within each vehicle which monitor the vehicle's data and are able to communicate with other nodes using wireless signals. Road-side units are placed in static locations near roads; they may also have wired interfaces with other RSUs and the Internet, so it is possible to use them as anchor points for Internet access for passing vehicles. When referring to the communication between two OBUs, the term vehicle-to-vehicle (V2V) communication is used [Yang et al., 2004]; when both an OBU and an RSU are involved, it is called vehicle-to-infrastructure (V2I) communication [Chou et al., 2009]. Although this nomenclature is important to understand other studies on the subject of VANETs, this study does not consider RSUs and focuses only on vehicles themselves as nodes, so references to VANETs and vehicular networks are exclusively tied to V2V communications.

In traditional networks (ad-hoc or not), routing protocols usually use the topology to choose where to forward packets; in other words, the primary metric used is the number of hops required to reach the destination. This metric is not as useful in vehicular networks, since the high mobility causes the topology to change frequently. Instead, most VANET routing protocols use geographical coordinates to forward packets [Saini et al., 2015], that is, the physical distance between two nodes is used as the primary metric. The implication is that, even if a packet requires more hops to reach its destination, it always travelling in the generally correct geographical direction.

As expected for a new technology being introduced, vehicular communications can become an appealing target for malicious users and attackers. These are some examples of possible issues in a VANET:

1. Vehicles with faulty GPS modules, speedometers or other sensors. If a vehicle is broadcasting incorrect data (perhaps unknowingly) because of a hardware or software fault, it can be a serious hinderance to efficiency and safety applications. It might behaving appropriately according to protocol, but the data it sends is not reliable [Isaac et al., 2010].

2. Vehicles might be deliberately broadcasting false data. In this case, there might be a specific purpose (by either the vehicle's driver or a remote attacker), like altering traffic or even cause an accident [Golle et al., 2004].

3. Attackers with control of several vehicles can propagate junk data in an attempt to flood the network, causing a distributed denial of service (DDoS) attack. Alternatively, the data propagated might have some reasoning behind it, like lying about road conditions in order to divert traffic [Garip et al., 2015].

4. Instead of sending data, some vehicles might try to eavesdrop on others' communications. The hop-based routing protocols used in VANETs facilitate this, since any node can be asked to be a hop. If the intermediary node is malicious, it may attempt to extract data contained in messages or refuse to forward them. Related to this, there is the Sybil attack [Isaac et al., 2010], in which a node lies about its position in order to seemingly alter the physical topology of the network and be chosen as a hop [Leinmüller et al., 2005].

5. Malicious vehicles may use signal jammers or other devices in order to affect other vehicles' sensors and communications [Isaac et al., 2010]. That can cause other vehicles to broadcast incorrect data, therefore obfuscating the origin of the attack.

6. A malicious user or remote attacker can monitor messages shared across the network in an attempt to stalk one specific vehicle [Isaac et al., 2010].

Each of these possible attacks requires a unique approach, though there are some broader ways to help the security and safety of VANET users. Trust can be an important feature in vehicular networks, especially when attempting to filter out malicious or incorrect messages. It does not, however, avoid all possible attacks, such as a signal jammer or stalking. Rather, different mechanisms must be explored in order to avoid most problems.

Like in other types of networks, the proper functioning of a VANET depends on the reliability of its participants. If a node is malicious or faulty, it can spread incorrect data that may compromise the network's utility. Once the concept of VANETs was established, researchers have been attempting to predict ways in which malicious users might use the network to their advantage. Examples include triggering false alarms about inexistent accidents, lying about the average speed in a road to make it less desirable for others, and falsifying geolocation data to exploit location-based routing algorithms. Therefore, the concept of trust must be established in the vehicular network context, allowing for nodes to judge the validity of information transmitted by others and share those conclusions with other nodes.

There is an important distinction between a malicious node and a faulty one; both of them may be sharing false data, but for different reasons and with different consequences. For example, a malicious node may lie about its location in order to make routing protocols use it [Leinmüller et al., 2005], in order to try to store or alter messages, while a faulty GPS module may cause an accident because its position data was incorrect. However, that distinction can be hard to make, because a close inspection is necessary to determine whether the incorrect data is erratic or deliberate. Since both types of nodes are problematic to the proper functioning of a network, malicious and faulty nodes can be treated as the same in a trust model.

In general, trust management solutions for VANETs use *data-oriented trust*, *entity-based trust*, or a combination of the two. The solutions that use data-oriented trust (or *data-centric trust*) [Raya et al., 2008] focus on validating messages instead of entities. This is important when vehicles share messages about a specific event, such as a collision, which must be quickly validated by neighbors and distributed to other nodes within a relevant area. In this scenario, vehicles sharing the same road might be complete strangers to each other, and therefore would not have any trust relationship, so neighboring nodes must decide if a message is true by its contents and by other nodes' observations of the event. On the other hand, when dealing with frequent messages which contain basic information such as geolocation and speed (used for traffic-diminishing solutions), it is too costly to judge each individual message. Therefore, *entity-based trust* becomes more appealing, since benign nodes can quickly identify a malicious node and isolate it from the network. Within entity-based trust, there are also two often-used methods of establishing trust: first, there is *role-based trust*, which is the static trust of pre-authenticated vehicles such as police units; second, there is *experience-based trust*, which is built through previous encounters shared between pairs of nodes. The model proposed in this work utilizes entity-based and experience-based trust, as it is based on the possibility of nodes meeting more than once and, therefore, being able to form a long-term trust relationship with each other.

2.4.1 Special properties of VANETs

VANETs feature several unique properties which distinguish them, and the behavior of its members, from other types of networks [Yousefi et al., 2006]. Some of these properties include:

1. Rapidly changing topology. Since the nodes are vehicles, they move frequently and at relatively high speeds. Each node's wireless communications also have a certain range, so the other nodes within that range (and, therefore, network neighbors) can change very quickly.

2. Node mobility is constrained to a pre-existing grid of roads. Within those roads, nodes usually travel in predictable directions according to local laws and historical data. The spaces in the grid, like city blocks, provide a challenge to communication both because of distance and because buildings can cause obstructions to radio transmissions.

3. VANETs are prone to fragmentation, since a gap in the network topology can make two parts of it unable to communicate with each other. Combined with the property above, this fragmentation can appear and disappear frequently, depending on the node density.

4. Due to the changing topology and possible disconnection, connection with distant nodes is not reliable. Therefore, the effective diameter of the network is relatively small for important applications.

5. Compared to devices like smartphones, vehicles have no notable power constraints.

6. In certain locations and/or moments, large vehicle density results in a large-scale network, since there are many nodes concentrated in a relatively small space.

7. The topology is susceptible to driver behavior. First, this means the topology can occasionally change in unpredictable ways. Second, contents of a message sent through the network can alter the driver's behavior and therefore change the topology.

Some of these properties provide advantages or disadvantages when developing trust models for vehicular networks, although all of them must be considered.

2.4.2 Desired properties for VANET trust models

The analysis of related work is based on [Zhang, 2011], which proposes eight desired properties for a trust management model for VANETs. In this section, these properties are briefly described.

1. Decentralized trust establishment: nodes must be able to form their own trust values about other nodes without the aid of an Internet connection or centralizing agents. Nodes may or may not use information from other trustworthy nodes to build trust values (in other words, trust might be transitive).

2. Coping with sparsity: the model still functions when there are few nodes populating the network. Due to the dynamic nature of vehicular networks, it is possible that nodes will find themselves with few other nodes in range. In such scenarios, a trust model should be able to establish trust even if there are few neighbors with whom to share data.

3. Event/task and location/time dynamics: the model reacts to different situations depending on what, where and when events happen. The event or task dynamics involve managing different situations in different ways. Messages can carry different types of alerts, and not all of them need to be addressed with the same urgency. A message about a nearby crash, for example, requires a much quicker reaction than one about an upcoming change in weather; malicious nodes that broadcast false information about critical events are especially important to detect. Similarly, in order to satisfy location and time dynamics, nodes might behave differently according to where and when certain messages are received. To do this, messages about events must contain timestamp and geolocation data attached; nodes close to the event in space and in time could be considered more trustworthy. Furthermore, by attaching timestamp data to messages, it is possible to age information, allowing the model to consider only data that is recent enough to be relevant.

4. Scalability: the model can work on very large networks at high speeds. This is very important in vehicular networks, since, at certain times or locations, there might be a very large number of vehicles very close to one another. In the case of a model that allows transitive trust, a high volume of nearby vehicles can be advantageous because it allows nodes to share a lot of recent data with each other.

5. Integrated confidence measure: allows nodes to estimate how useful the output of the algorithm is. Along with the information of whether or not a node *a* trusts *b*, there should also be information regarding *how sure a* is of its trust in *b*. Generally, a higher confidence measure is the result of more and/or better evidence.

6. System level security: requires authentication of nodes participating in the network. There should be an infrastructure in place in order to avoid identity falsification from potentially malicious nodes as well as verifying which node is the sender of a given message.

7. Sensitivity to privacy concerns: avoids eavesdropping and stalking by malicious nodes. A message should only be received by the nodes it was meant for, avoiding eavedropping of its contents. Additionally, it should not be possible to track the activity of a node based on the messages it sends.

8. Robustness: the model's resistance to attacks. There are already some studied attacks for vehicular networks, such as the Sybil Attack, Newcomer Attack and Betrayal Attack. Models must show that they function in the event of such attacks.

2.4.3 Existing trust models for VANETs

Several models have been proposed to solve the problem of trust in vehicular networks. In this section, the most relevant ones are described, considering the time in which they were proposed, the advantages they bring and their contributions to later study. None of them provide a complete solution, but serve as pieces of a puzzle that is still incomplete. Many trust management solutions for VANETs have been proposed over the years, such as [Patwardhan et al., 2006], [Gerlach, 2007], [Raya et al., 2008], [Huang et al., 2010], [Ding et al., 2013], [Haddadou et al., 2013], [Liu et al., 2016], [Kerrache et al., 2016]. There are also some review and/or survey articles on the subject of VANET trust models, such as [Zhang, 2011], [Ma et al., 2011], [Zhang, 2012], [Mejri et al., 2014], [Soleymani et al., 2015] [Sengar, 2016], and [Dwivedi and Dubey, 2016].

[Dotzer et al., 2005] is one of the earliest examples of VANET trust models, establishing a system called VARS, based on the reputation of nodes and messages throughout the network. The authors use what they call *opinion piggybacking*, which means that, for each hop between the origin and the destination of an event-related message, the forwarding node includes its opinion of the message's contents and the message's sender. In other words, when a node *a* receives a message about a certain event from *b*, it calculates a new opinion considering it rebroadcasts the message to other nearby nodes, but with its own opinions about the event and about node *b* attached. This process adds credibility to a message through validation by nodes in a decentralized fashion. It combines aspect of data- and entity- based trust, since nodes share their opinion of the data as well as their opinion of the sender. An interesting observation is setting higher trust values for certain vehicles based on their familiarity with the region (vehicles that reside in a given city may have more experience with certain types of events than newcomers). However, opinion piggybacking has its own share of problems. First, it allows forwarding nodes to access (at least some of) the contents of a message so it can form an opinion on it, diminishing privacy; a malicious forwarding node could even attempt to alter those contents. Second, since each new opinion appended to the message considers the previously appended opinions, the first nodes to forward the message to have a substantially greater impact over the final opinion than the later ones. Finally, there is an issue with scalability, since appending new information to a message on each hop may add a significant overhead to the transmission. Additionally, the authors provide little to no experimentation or proof that their approach would be sound in a real-world network.

The model proposed in [Minhas et al., 2010] uses several criteria to judge whether or not a received message is trustworthy. First, nodes are classified by their roles and previous experience with them. Roles are used for vehicles which should be more trustworthy than the average: government official cars, traffic report vans, buses, cabs, etc. Nodes also store their experience each time an event message is received (if one neighboring node reported an event which did not turn out to be true, its trust value is reduced). Additionally, messages are considered more reliable when their senders were direct witnesses to an event (i.e. were close to the event when it happened). When several messages about the same event are received, a node can either choose the *n* most trustworthy senders, according to the priority of the event. For example, events that require a fast response might cause a decision to be made using fewer messages about it, which diminishes precision. The model considers both role-based trust and experience-based trust; although the work proposed here does not use role-based trust, the authors provide a useful method of calculating and updating an experience-based trust value, which might be used or adapted. However, their model relies only on direct interaction between pairs of nodes, so no form of indirect trust (that is, trust values received from other nodes) is considered.

In [Chen et al., 2010], the authors propose to evaluate messages utilizing a cluster-based trust model. By separating nodes into clusters with their geographical neighbors, it is possible to efficiently distribute the evaluation of messages using previously formed opinions. When a node sends a message, one node in the cluster (the leader) must aggregate the other nodes' opinions on that message. Afterward, the message is only forwarded to another cluster if that aggregate opinion is above a certain threshold; furthermore, nodes that receive the message only act upon it if the overall trust on it is above another threshold, which can be different according to the nature of the message. However, it is unclear how the model behaves when the network is too sparse – nodes are few and/or far apart – to form relevant clusters, neither do the authors inform how the aforementioned thresholds are decided. Furthermore, maintaining clusters in a highly dynamic network is a costly job and, if the node chosen to be the leader of a cluster happens to be malicious, all the information from that cluster could be compromised.

The trust model in [Park et al., 2011] takes advantage of daily commutes. In this article, the focus is on the early stages of VANETs, in which a very small percentage of vehicles are equipped with OBUs. To make trust viable in such a scenario, the authors rely on RSUs to store reputation information from passing vehicles. Each vehicle must have an "Agent RSU", which is in charge of storing and sharing that vehicle's trust data to other passing vehicles and connected RSUs. It must also keep the data updated when the vehicle approaches it again. To make this viable, the properties of daily commutes are used: it is assumed that the vehicle is near its Agent RSU with reasonable frequency because it is located within the driver's home-to-work route. The main problem with this model is that it relies on the existence of a reasonably large number of RSUs, which might not always be viable due to infrastructure costs. It also does not make it clear what should happen when a vehicle stops using a route or does not have a daily predictable path (it does, however, handle occasions in which a vehicle chooses an alternate route or is absent for some days such as weekends and holidays).

The authors of [Huang et al., 2014] take special note of two characteristics from social networks that can also be found in many VANET trust models: *information cascading* and *oversampling*. That is, information reported by a number of original nodes (i.e. the ones that witnessed an event) may be diluted as nodes that forward it append their own opinions on the matter. Following the VARS [Dotzer et al., 2005] approach, an algorithm is proposed to diminish that effect by assigning higher weights to the opinions of nodes that were witnesses to an event and lower weights to forwarding nodes. However, the authors conclude that the optimal scenario is to assign no weight at all to forwarding nodes, therefore allowing each node to form an opinion based only on the original nodes' reports. Furthermore, the authors are quick to dismiss the validity of entity-based trust, instead opting for a pure data-oriented approach, considering only the contents of a message and disregarding who sent it. Although it is true that data-oriented trust is efficient for events, which is what their model is based on, it is not ideal for sharing data quickly and frequently. When a collision or other major incident occurs, it is useful to judge each message on its own, since not all members of the network will have existing trust relationships with each other. However, when sharing location and velocity data several times per second, it is not reasonable to expect that each message will be analyzed so carefully; rather, it makes sense to form an opinion about the sender of the message and use the resulting trust value to choose which messages are relevant or not.

The Attack-Resistant Trust Management Scheme (ART) from [Li and Song, 2016] proposes to resist three types of attacks: simple attacks, in which malicious nodes do not cooperate with the network; bad-mouth attacks, in which malicious nodes perform simple attacks but also share false information about benign nodes; and zig-zag attacks, in which nodes vary their behaviour from benign to malicious in order to be harder to detect. It works in two main steps: data gathering and malicious node detection. For data gathering, it uses the Dempster-Shafter theory of evidence, which establishes *belief* and *plausibility* values, both real numbers ranging from 0 to 1. The former refers to the amount of evidence indicating the truthfulness of a hypothesis (for example, a node sending false data corroborates the hypothesis of it being malicious), while the latter is 1 minus the amount of evidence that supports the possibility of the hypothesis being false. These evidences are acquired through observations by a node and through data received from other nodes. The resulting probability is the basic trust value, which is stored in a trust vector (a series of trust values regarding other nodes). The malicious node detection step uses a Cosine-based metric to compare two nodes' trust vectors. When two nodes share similar opinions about other nodes, they will consider each other trustworthy. The downside of this model is that each step is mathematically costly, requiring several intensive calculations in order to achieve its goals. This likely increases the complexity of the algorithm, which is not detailed by the authors. Because of this, it is uncertain how the model would scale to large networks, while it might also underperform in small networks in which there is little evidence to collect.

The authors of [Chen and Wang, 2017] propose a cloud-based solution for a trust model, which requires an Internet-based global trust manager. This has the advantage of simplifying properties such as handling sparsity and scalability, but also makes the system slower in general, especially in situations in which mobile communication is slow or unreliable. In general, it goes against several established concepts for vehicular networks, such as being completely decentralized and ad-hoc. It also makes the system somewhat unreliable, since the whole system collapses if the global trust manager is attacked, or could leave the network members unaided in the event of a server or connection failure.

2.5 Discussion

Analyzing the previous work in the subject of trust management for vehicular networks, it is possible to observe that many interesting solutions to the problem have been attempted, but all of them show at least one major deficiency when thinking about real-world implementation. Table 2.1 shows how well each studied trust model satisfies the eight properties described in Section 2.4.2, which are based on [Zhang, 2011]. The numbers used in the table are the same ones from Section 2.4.2.

Table 2.1: Properties of the related work

Property	1	2	3	4	5	6	7	8
[Dotzer et al., 2005]	✓	-	✓	-	-	-	✓	-
[Minhas et al., 2010]	✓	✓	✓	✓	✓	✓	✓	-
[Chen et al., 2010]	✓	✓	✓	✓	✓	✓	-	-
[Park et al., 2011]	-	-	-	-	-	✓	✓	✓
[Huang et al., 2014]	✓	-	✓	✓	✓	-	-	-
[Li and Song, 2016]	✓	-	-	-	✓	-	-	✓
[Chen and Wang, 2017]	-	✓	-	✓	-	✓	✓	-

Desired properties

1. Decentralized trust establishment
2. Coping with sparsity
3. Event/task and location/time dynamics
4. Scalability
5. Integrated confidence measure
6. System level security
7. Sensitivity to privacy concerns
8. Robustness

Furthermore, vehicular ad-hoc networks are a special kind of network, but, despite the unique properties explained in Section 2.4.1, they still share many features with mobile ad-hoc networks and even with social networks.

With this information in mind, a new trust model is introduced, taking advantage of social features in order to establish long-term trust relationships between participants of a vehicular ad-hoc network. This new trust model emphasizes efficiency while maintaining correctness and most of the desired properties for such a model. In the following chapter, the model is explained in detail.

Chapter 3

Design and Implementation of TruMan

This work introduces an efficient solution to trust management in dynamic networks such as VANETs. In order to make this possible, it is necessary to identify features in VANETs that show that nodes can share a long-term relationship, as is the case for social networks. Through these long-term relationships, it then becomes feasible for nodes to store trust data and share it with other nodes. By combining a node's own opinions about familiar nodes and trust information received from its neighbors, it is possible to create a model of the surrounding network. This model includes a trust graph, showing the trust relationships between pair of nodes, which can then be used in conjunction with other algorithms in order to classify nodes as correct or malicious.

In this chapter, the reasoning behind TruMan and details of how it works are presented. First, it is shown how vehicles can form long-term relationships and trust one another in a similar way to social networks. Then, two algorithms are introduced: Tarjan's strongly connected components algorithm [Tarjan, 1972] and an efficient algorithm for graph coloring [Mittal et al., 2011]. Next, the Malicious Node Identification Algorithm (MaNI) [Vernize, 2013] is explained, because it is the work that suggests the usage of strongly connected components and graph coloring for malicious node detection. Finally, TruMan itself is detailed, showing how it combines features from existing algorithms and adapts them to a dynamic environment, enabled by the social properties found in vehicular networks.

3.1 Goals

Building from the foundation set by MaNI [Vernize, 2013], TruMan strives to enable efficient trust management in highly dynamic networks such as VANETs. In addition to efficiency, it is desirable that the trust model is both simple to understand and to implement, making it appealing for real-world usage.

Furthermore, the desired properties of a VANET trust model described in [Zhang, 2011] (explained in detail in Section 2.4.1) were considered, so TruMan attempts to fulfill or enable as many of those properties as possible.

3.2 Social Networks and VANETs

Some proposed trust models for vehicular networks, such as [Huang et al., 2014], state that the likelihood of two nodes meeting each other twice is too low to be relevant. However, it stands to reason that, throughout the course of several days, many drivers take similar routes at similar times of day (e.g. to commute to work) and, therefore, their vehicles are in similar locations each day. Additionally, many cities rely on main roads to serve as backbones to their traffic, meaning there is a high density of vehicles on those roads during rush hours. Since that is true for a notable percentage of a city's fleet, it can also be assumed that those vehicles may frequently encounter each other during their commute. While two vehicles that share a commute route may not be direct neighbors every day, they are likely to be relatively close to each other most days, meaning few hops separate them in the ad-hoc network. Furthermore, certain pairs of vehicles are bound to be within communication range of each other nearly every day. Examples of these include vehicles whose owners are neighbors or coworkers. Such vehicles' trust relationship should become steady over time and, in the case of positive trust, they can use each other's information to learn more about other nodes in the network.

Most cities also have one or more types of mass transit systems (buses or trains). Those vehicles can also be part of a VANET and communicate with private cars. Buses share the same roads as cars, but instead of having specific destinations, they travel a predefined route during the whole day, usually tied to a tight schedule. Trains travel on rails, so their contact with cars is less frequent, but it can also happen on railroad crossings; they travel long distances in relatively short amount of time, which helps the dissemination of data in a VANET. In the same way that cars have a high probability of meeting more than once during their commutes, it is also very likely that they meet the same buses and/or trains frequently.

In [Cunha et al., 2013], [Cunha et al., 2014b], and [Cunha et al., 2014a], the authors attempt to find features usually attributed to social networks in vehicular networks. By using a data set from the city of Zurich, Switzerland, they show that some metrics, such as clustering coefficient and number of encounters, have peaks during the rush hours. They note that, during rush hours, the diameter of the graph decreases to around 6 hops; additionally, the frequency of total encounters between pairs of nodes in the network increases during those hours. Although the authors do not quantify the encounters between specific pairs of nodes, these numbers support the idea that daily commutes do indeed cause vehicular networks to exhibit social network features.

3.3 Tarjan's strongly connected components algorithm

The use of Tarjan's strongly connected components algorithm [Tarjan, 1972] is an important aspect of TruMan's efficiency. This allows a large graph to be abstracted into a smaller one, which therefore reduces the input for further steps. Given a directed graph $T = (V, E)$, a strongly connected component is defined as a group of nodes in which, for any pair of nodes $u, v \in V$, there is a path from u to v and a path from v to u. For the purposes of trust management, this definition is extended to accept only paths of edges with weight above a predetermined threshold h. Every node of the input graph T must belong to a component.

The algorithm works by performing a depth-first search, adding nodes to a stack as they are visited. If two nodes are present on the stack, then there is a path from the first node to the second one (in the order they were added to the stack).

Each node has two attributes assigned to it during the execution of the algorithm: $index$ is used to number the nodes in the order they are visited, while $lowlink$ is the lowest indexed node reachable from each node. In the implementation used, $index$, $lowlink$, $count$ and $stack$ are global variables accessed from every call of the function. $index$ and $lowlink$ are arrays indexed by unique node identifiers, $count$ is an integer and $stack$ is a last-in-first-out data structure.

In the call that visits a node u, the algorithm must loop through each node v trusted by u (that is, $u \to v$ exists and has value greater than h). If node v has not yet been visited, the algorithm is called for v. The $lowlink$ of u is then calculated as the smallest value between

$lowlink[u]$ and $lowlink[v]$, because any node reachable from v is also reachable from u. After the loop, if $lowlink[u]$ is equal to $index[u]$, it means that u is the lowest indexed node reachable from itself and that it is the root of a component. Therefore, nodes must be popped from the stack until u is found. Each node popped, including u, is a member of a strongly connected component.

The number of components is, at most, $|V|$: in a worst-case scenario, each node is placed into its own component. The complexity of the algorithm is $O(|V| + |E|)$ for a graph $T = (V, E)$. Algorithm Algorithm 1 shows the general structure of Tarjan's algorithm [Tarjan, 1972].

Algorithm 1 Tarjan's strongly connected components algorithm

```
function TARJAN(vertex u)
    index[u] = count
    lowlink[u] = count
    count ← count + 1
    push u to stack
    for v in neighbors of u do
        if weight of u → v < h then
            continue
        if index[v] = −1 then    // v has not been visited yet
            Tarjan(v)
        lowlink[u] ← min(lowlink[u], lowlink[v])
    if lowlink[u] = index[u] then
        repeat   // unstack nodes until u is found
            pop w from stack
            add w to component
        until w = u
```

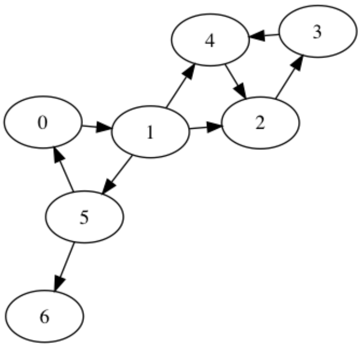
Figure 3.1 illustrates the execution of Tarjan's algorithm. The algorithm starts from node 0, with $index[0] = 0$ and $lowlink[0] = 0$. With a depth-first search, the algorithm traces the path $0 \to 1 \to 2 \to 3 \to 4 \to 2$. Since node 2 has already been visited and nodes 4 and 3 have no further outgoing edges, $lowlink[4]$ and $lowlink[3]$ receive the value 2 and the function calls return back to the first visit of node 2. At this point, nodes 2, 3 and 4 all have 2 as the smallest reachable index and, therefore, they form a strongly connected component.

Continuing from node 1, the algorithm traces $1 \to 5 \to 0$, but stops there since node 0 has already been visited. Continuing from node 5, the algorithm traces $5 \to 7$. Node 7 has no outgoing edges, so it forms a strongly connected component by itself. Once the function calls return to node 0, a strongly connected component is formed with nodes 0, 1, and 5, since they all have 0 as their $lowlink$ value.

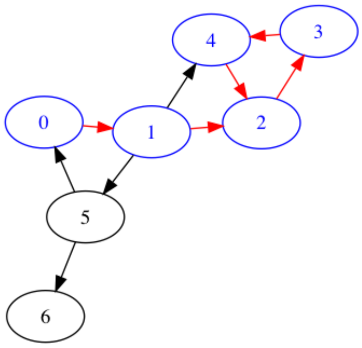
3.4 Graph coloring with minimum colors

Graph coloring is one of the possible heuristics suggested by MaNI to detect malicious nodes after the generation of the component graph using Tarjan's algorithm. Out of the tested heuristics, it presents the best results, so it has been chosen as the heuristic for TruMan.

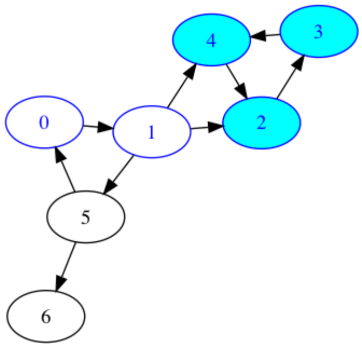
The process of graph coloring consists of giving each node a label so that no two neighboring nodes share the same label. When visualizing a graph, labels are represented by colors, although they can be any type of data. This problem has been studied in Computer Science since, at least, 1972 [Karp, 1972] and has been studied as a classic mathematics problem



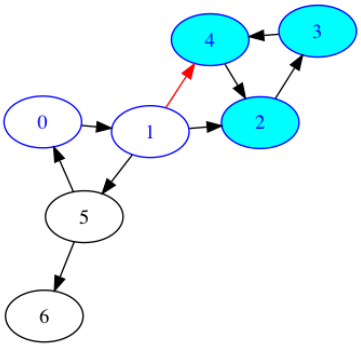
(a) Initial graph.



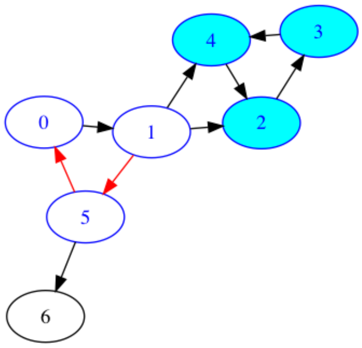
(b) The search follows the path $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 2$.



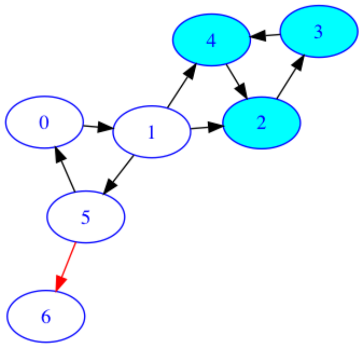
(c) Since $lowlink[2] = 2$, nodes 2, 3 and 4 are added into a component.



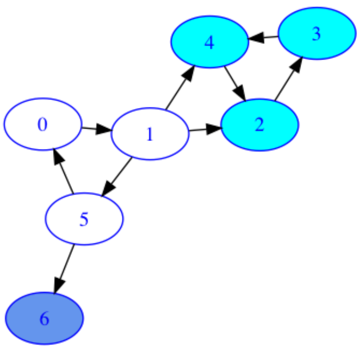
(d) The search follows the path $1 \rightarrow 4$, but node 4 has already been visited.



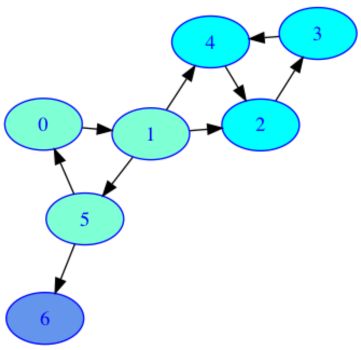
(e) The search follows the path $1 \rightarrow 5 \rightarrow 0$.



(f) The search follows the path $5 \rightarrow 6$.



(g) Node 6 does not have outgoing edges and is added to a component.



(h) The algorithm returns to 0 and, since $lowlink[0] = 0$, adds 0, 1 and 5 to a component.

Figure 3.1: Example of an execution of Tarjan's strongly connected components algorithm.

for even longer [Kempe, 1879]. It has been proven mathematically that any planar graph can be colored with at most four colors [Appel et al., 1976], but discovering the smallest number of colors necessary to color an arbitrary graph (called the graph's chromatic number) is an NP-hard problem [Sánchez-Arroyo, 1989].

In [Mittal et al., 2011], the authors present an efficient approach to graph coloring using the minimum possible amount of colors. Although they do not prove that their algorithm always uses the smallest possible amount of colors, the output is always a correct coloration and the algorithm is nevertheless efficient. For the purposes of trust management, it is not necessary to prove that the coloring algorithm's output uses the minimum possible number of colors.

The complexity of the algorithm is $O(|E'|)$ for a graph $C = (V', E')$. As a comparison, the DSATUR algorithm for graph coloring has complexity $O(|V|^2)$ [Brélaz, 1979]. Algorithm Algorithm 2 shows the general structure of the graph coloring algorithm [Mittal et al., 2011].

Algorithm 2 Graph coloring with minimum colors

function COLORING(graph G)
    color all nodes of G **with** 0
    $d \leftarrow 0$
    for $e = (u, v)$ **in** edges of G **do**
        if u and v have the same color **then**
            if $color[v] = d$ **then**
                $d \leftarrow d + 1$
            $color[v] \leftarrow d$

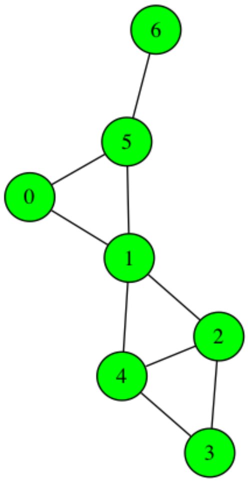
A limitation of this algorithm is that the edges must be sorted according to node indexes. It doesn't matter which nodes get assigned which indexes, but once they are assigned those numbers, the algorithm must follow the edges in numerical order. This is demonstrated in [Vernize, 2013].

Figure 3.2 illustrates the execution of the graph coloring algorithm. It is notable how the algorithm takes few iterations to fully color the graph. However, it is also possible to observe that the result does not use the minimum amount of colors. By coloring node 4 as cyan and node 3 as magenta, the sample graph could have been colored with only three colors instead of four. As described above, this is not a problem for the usage of the algorithm in TruMan.

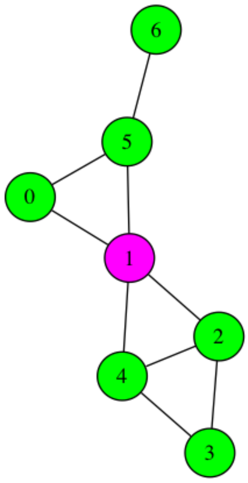
3.5 Malicious Node Identification Algorithm

The basis of TruMan is the Malicious Node Identification Algorithm (MaNI) proposed in [Vernize et al., 2015], which suggests the use of strongly connected components and graph coloring for malicious node detection. This article presents a malicious node identification scheme based on strongly connected components and graph coloring. The model is proposed for complex networks in general, but is not suited for VANETs because it is designed only for static networks. Furthermore, the algorithm is executed by a global observer which has information about the complete network.

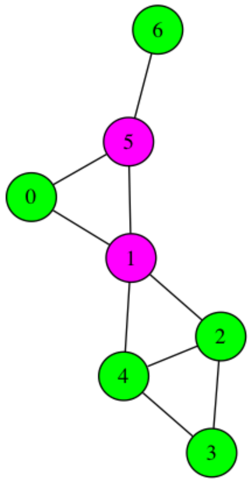
The input graph $T = (V, E)$ is a static, connected, and directed graph containing all trust relationships in the network. Such relationships are binary, so there are no varying degrees of trust: either one node trusts another completely (edge value is 1), or it distrusts the other completely (edge value is 0). The relationships are also directed, meaning that if the value of $A \rightarrow B$ is 1, $B \rightarrow A$ is not necessarily 1.



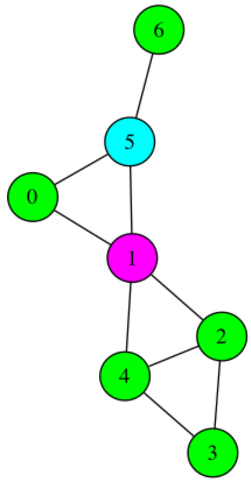
(a) Initial graph with all nodes labeled 1. $d = 1$ (green).



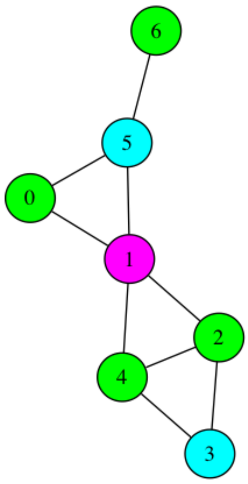
(b) Edge $(0, 1)$ is checked and node 1 gets a new color. $d = 2$ (magenta).



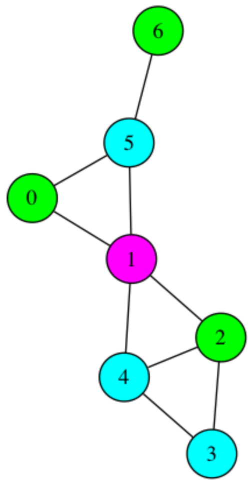
(c) Edge $(0, 5)$ is checked and node 5 gets the current value of d.



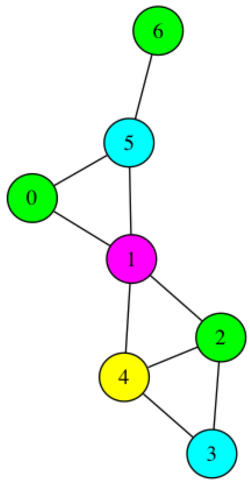
(d) Edge $(1, 5)$ is checked and node 5 gets a new color. $d = 3$ (cyan).



(e) Edge $(2, 3)$ is checked and node 3 gets the current value of d.



(f) Edge $(2, 4)$ is checked and node 4 gets the current value of d.



(g) Edge $(3, 4)$ is checked and node 4 gets a new color. $d = 4$ (yellow).

Figure 3.2: Example of an execution of the graph coloring with minimum colors algorithm.

The process for identifying malicious nodes within T is as follows:

First, T is separated into strongly connected components using Tarjan's algorithm [Tarjan, 1972], which is described in detail in Section 3.3. In each of these components, there are paths formed by edges of value 1 connecting each pair of nodes. In other words, within a single component, all nodes trust one another directly or indirectly; nodes which do not do not meet this criteria are separated into different components. Each of these components becomes a node of a component graph $C = (V', E')$.

The creation of the graph C simplifies the remaining computation. Since each node of C is a vertex $v' \in V'$ and each vertex v' is a component of T in which all nodes trust each other directly or indirectly, for the purposes of identifying malicious nodes, all nodes within each of those components can be treated as one. They can either be benign nodes which legitimately trust one another, or malicious nodes colluding with each other. After the formation of C, one or more heuristics can be used to classify the nodes as benign or malicious.

In the experiments performed by the authors of MaNI, the coloring heuristic shows the most promising results, identifying a high ratio of the malicious nodes in the network. The coloring heuristic uses a graph coloring algorithm, such as DSATUR [Brélaz, 1979] or the algorithm detailed in Section 3.4. Other heuristics were experimented with, but were either less effective in detecting malicious nodes, provided too many false positives, or were not efficient enough.

After running a graph coloring algorithm with graph C as input, the color whose nodes in C represent the most nodes in T is classified as correct, and all others are classified as malicious. Once this information from C is brought back to graph T, it is trivial to label the nodes in T as either benign or malicious based on their components' classifications.

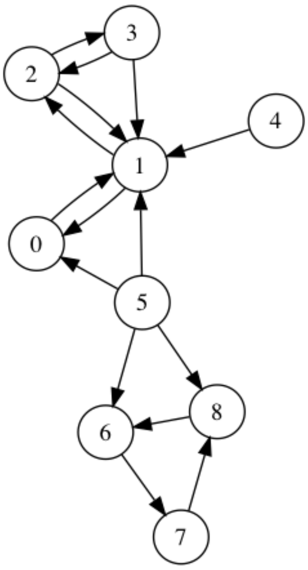
Two types of experiments were made in each network: first, all malicious nodes inverted the edge weights leading to their neighbors; second, malicious nodes randomly inverted or not the weights. In the first scenario, the results show excellent precision in most networks, detecting nearly every malicious node. Experimenting with the second scenario, the results are less precise, however still promising: with up to 20% of malicious nodes in the network, the error rate is under 7%, while with the worst case, 50% of the network being malicious, the error rate is approximately 15%.

The authors suggest running the algorithm repeatedly after removing the malicious nodes from the network. By doing this a small number of times, nearly all malicious nodes are detected by it even when randomly changing edge weights.

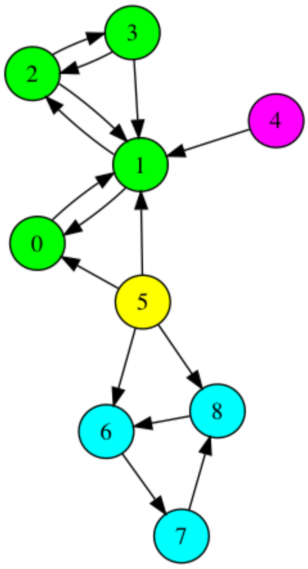
Figure 3.3 illustrates the execution of the MaNI algorithm. With the starting graph T, whose edges represent the trust relationships between nodes, Tarjan's strongly connected component algorithm is executed. Figure 3.3(b) shows nodes colored according to their placement in a strongly connected component. The strongly connected components form a graph C according to edges present in T. In Figure 3.3(c), component 0 is the one containing nodes 6, 7 and 7; component 1 contains node 5; component 2 contains nodes 0, 1, 2 and 3; and component 3 contains node 4. The graph coloring with minimum colors algorithm is executed on C, producing the coloration shown in Figure 3.3(d). Finally, each node in T is colored according to which color its component received in C. The color with the most nodes is deemed benign, while the others are considered malicious.

3.6 The TruMan algorithm

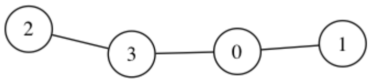
TruMan is based on the MaNI algorithm [Vernize et al., 2015], which suggested the use of Tarjan's algorithm and the graph coloring algorithm. However, MaNI was developed for static



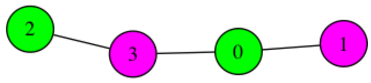
(a) Initial graph T.



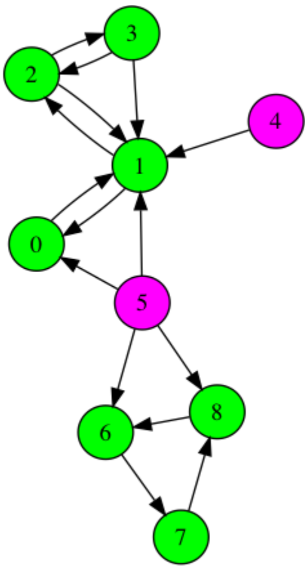
(b) Tarjan's algorithm is used to identify the strongly connected components.



(c) The graph C is formed from the components of T.



(d) The coloring algorithm is used to label the nodes of C.



(e) The colors from C are used to label nodes in T as correct or malicious.

Figure 3.3: Example of an execution of the MaNI algorithm.

networks such as social networks, and is executed by an external supervising agent (i.e. outside of the network), making it unsuitable for a vehicular network.

In order to work with dynamic networks, the TruMan algorithm runs iterations at predetermined intervals. Furthermore, the algorithm runs in a decentralized fashion, meaning each node in the network runs its own instance of the algorithm. Each node starts knowing information only about itself and maintains its own abstraction of the network surrounding it. Every node u stores a representation of the network in the form of a static, connected and directed trust graph $T^u = (V^u, E^u)$, in which V^u is the set of nodes node u is aware of and E^u is the set of trust relationships (opinions) u knows of between members of V^u. Since each node has its own network representation and it changes over time, there is a $T_i^u = (V_i^u, E_i^u)$ for every node u and iteration i.

At first, the node collects and organizes information. A prerequisite of this step is a test that correctly classifies a neighboring node as benign or malicious. Testing the correctness of neighboring nodes is a problem in and of itself, which is beyond the scope of this project, but studies on this topic can be found on [Golle et al., 2004], [Kerrache et al., 2016], [Li et al., 2016].

Every time a neighboring node v is tested as benign, the value of $u \rightarrow v$ increases. Additionally, node u performs an union between its trust graph and v's trust graph, forming a new graph $T_i^u = T_{i-1}^u \bigcup T_{i-1}^v$, which is used for the remaining steps. Algorithm Algorithm 3 shows the basic interaction between two nodes, while algorithm Algorithm 4 details the steps of the graph union.

Algorithm 3 Interaction between two nodes u and v

if $v \notin T^u$ **then**
    add v **to** T^u
    add $u \rightarrow v$ **to** T^u
    $T^u(u \rightarrow v).trustvalue = 0.5$
    $T^u(u \rightarrow v).timestamp =$ **now**
u **tests** v
if v is benign **then**
    $T^u(u \rightarrow v).trustvalue$ **increases**
    $T^u \leftarrow$ **Union**(T^u, T^v)
else
    $T^u(u \rightarrow v).trustvalue$ **decreases**

Algorithm 4 Graph union

function Union(graph T^u, graph T^v)
    $T \leftarrow T^u$
    for $a \rightarrow b$ **in** edges of T^v **do**
        if $a \notin T$ **then**
            add a **to** T
        if $b \notin T$ **then**
            add b **to** T
        if $a \rightarrow b \notin T$ **then**
            add $a \rightarrow b$ **to** T
    return T

After the collection of data, T_i^u is separated into strongly connected components using Tarjan's algorithm [Tarjan, 1972], although the implementation of the algorithm slightly differs from the one used in MaNI. Since MaNI uses binary trust, Tarjan's algorithm only checks whether edges have value 0 or 1; in TruMan, each edge stores a trust value $t \in [0, 1]$.Therefore, a threshold h is defined so Tarjan's algorithm can consider only edges represent a significant trust relationship when forming strongly connected components. So, for each node in a component, there is a path formed by edges of weight higher than the threshold h to each other node in the same component. Each of these components becomes a node of a component graph $C_i^u = (V_i'^u, E_i'^u)$.

Since each vertex $v' \in V_i'^u$ is a component of T_i^u in which all nodes trust each other, for the purposes of identifying malicious nodes, all nodes within each of those components can be treated as the same. They can either be benign nodes which legitimately trust one another, or malicious nodes colluding with each other. After the formation of C_i^u, a heuristic is used to classify the nodes as benign or malicious.

The coloring heuristic is used to classify nodes, which uses the algorithm described in Section 3.4 [Mittal et al., 2011], although other heuristics may be considered. After running the graph coloring algorithm with graph C_i^u as input, the color whose nodes in C_i^u represent the most nodes in T_i^u is classified as correct, and all others are classified as malicious. Once this information from C_i^u is brought back to graph T_i^u, it is trivial to label the nodes in T_i^u as either benign or malicious based on the classifications of their components.

In a network in which malicious nodes are a minority (under 50%), it is expected that the benign nodes will form components with large numbers of nodes, because these benign nodes will share their networks with each other and it is easy to form trust paths between pairs of benign nodes. Malicious nodes, on the other hand, do not send their own networks of false information to benign nodes, and might not always trust other malicious nodes, causing them to become isolated in small strongly connected components (in some cases, these contain a single malicious node). The result is that most benign nodes become members of a small number of large components; when the component graph is colored, these components are likely to receive the same color, because two benign components are almost never adjacent [1]. Because of this, the coloring heuristic works as a classification method. A large scale collusion attack (in which malicious nodes form nearly half the network or more) could affect this heuristic, as malicious nodes would form large components and distinguishing these components from the benign ones would be a challenge.

In summary, every node u runs the following steps in each iteration to detect malicious nodes in the network:

1. Node u checks which are its neighbors (nodes within its communication range). New discovered nodes and new formed edges are added to T_i^u. Edges are created with weight 0.5.

2. Node u tests all its neighbors to discover which ones can be directly trusted or not. New trust values are computed for the edges using the average between the previous value and either 1 (if the neighbor is trustworthy) or 0 (otherwise).

3. If a neighbor v is trustworthy, u performs a union with T_{i-1}^u and T_{i-1}^v, establishing T_i^u.

[1] When a node is building its local representation of a network, new nodes are almost certainly identified as malicious. This happens because there is a trust edge going from an already established node to the new one, but the returning edge is not there yet. Therefore, two adjacent benign nodes are separated into different components. This causes inaccurate results in the very beginning of execution, and becomes a minor detail afterwards.

4. Tarjan's algorithm is executed to identify the strongly connected components of T_i^u, resulting in a component graph C_i^u.

5. The graph coloring algorithm is executed on C_i^u and nodes are classified as benign or malicious.

3.6.1 Information aging

In order to make TruMan resistant to attacks, it is necessary to age the information nodes store about the network, so old information does not affect the identification of malicious nodes.

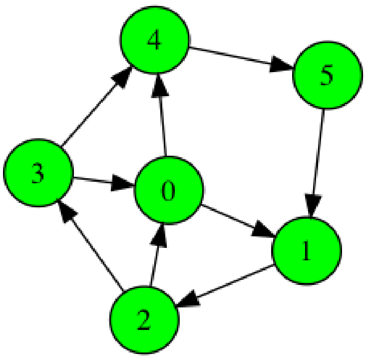
To do this, each edge stores a timestamp value s in addition to the trust value t. When two nodes u and v interact with each other, the edge $u \rightarrow v$ in T_u stores the timestamp s, which is set according to node v's internal clock.

Then, when node u interacts with another node w in the future, u's opinion of v comes with the timestamp attached, so w has the information to know how much time has passed since the interaction between u and v happened. Once w performs the graph merge procedure with T_u and T_w, but T_w already has an edge $u \rightarrow v$ stored, it checks the two timestamps and only updates the edge if the incoming information is more recent than the node already had stored. Since the timestamp is always set to the destination node of the edge, it can be used for comparison regardless of which nodes handled the information. Algorithm Algorithm 5 shows how the union function was changed to accommodate timestamps and information aging.

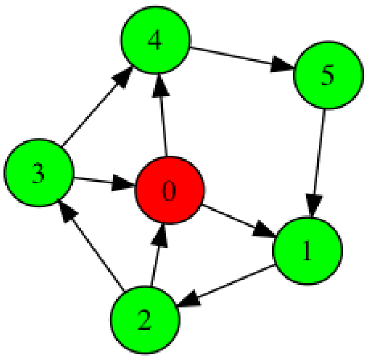
Algorithm 5 Graph union with timestamps

function UNION(graph T^u, graph T^v)
    $T \leftarrow T^u$
    for $a \rightarrow b$ **in** edges of T^v **do**
        if $a \notin T$ **then**
            add a **to** T
        if $b \notin T$ **then**
            add b **to** T
        if $a \rightarrow b \notin T$ **then**
            add $a \rightarrow b$ **to** T
        else
            $s_u \leftarrow T(a \rightarrow b).timestamp$
            $s_v \leftarrow T^v(a \rightarrow b).timestamp$
            if s_u **is more recent than** s_v **then**
                $T(a \rightarrow b).trustvalue \leftarrow T^v(a \rightarrow b).trustvalue$
                $T(a \rightarrow b).timestamp \leftarrow T^v(a \rightarrow b).timestamp$
    return T

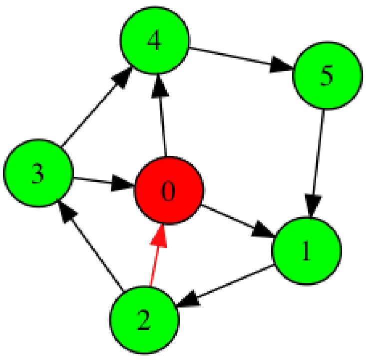
For example, when u interacts with v on and establishes that v is benign, it creates the edge $u \rightarrow v$ with $t \geq 0.5$ and $s = s_0$, where s_0 is the timestamp of the moment in which the interaction occurred according to v. Then, when w interacts with u and establishes that u is benign, it receives network information from u, including the edge $u \rightarrow v$. Since w didn't have that edge in its graph before, it is added maintaining the original timestamp s_0. Later, when w interacts with u again or with another trustworthy node that recently interacted with u, it receives information about the edge $u \rightarrow v$ again. If this new information includes the timestamp s_1 which is more recent than s_0, w updates its own graph so that the edge $u \rightarrow v$ stores the new timestamp and the new trust value.



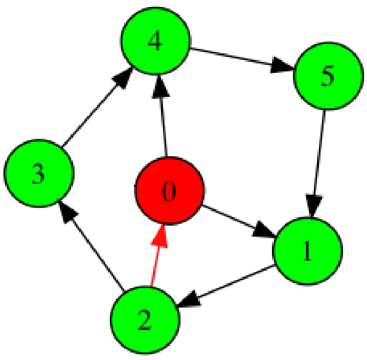
(a) All nodes are benign and belong to the same strongly connected component.



(b) Node 0 becomes malicious, but still belongs to the component.



(c) Node 2 updates its opinion of node 0, identifying it as malicious.



(d) 3's opinion becomes too old and is discarded.

Figure 3.4: Example of what happens when a node becomes malicious.

Furthermore, after each iteration, nodes go through the edges they have stored and discard the ones that are too old. This is done by setting a maximum age m for edges; once an edge is older than m, it is discarded. A low value for this setting allows for faster detection of converted nodes, however also increases the likelihood of good information being thrown away (therefore increasing the likelihood of false positive detections).

Information aging is necessary because, when a node converts from benign to malicious, old information might still indicate that it is benign, as shown in Figure 3.4. Immediately after node 0 becomes malicious, it is still part of a benign strongly connected component, because there are paths from every other node to 0, and from 0 to every other node. Some time later, node 2 interacts with 0 again and identifies it as malicious; other nodes that interact with 2 will receive that updated information. However, the outdated edge $3 \rightarrow 0$ keeps 0 in the component, since other nodes trust 3 and 3 still hasn't updated its own opinion of 0. Once that edge is old enough, it is discarded and node 0 is correctly identified as malicious by all other nodes.

3.6.2 Complexity

The complexity of TruMan must be calculated for each iteration executed and each node in the network. It can be estimated by adding the complexity of most costly operations involved, which are: (i) Tarjan's algorithm, (ii) the graph coloring algorithm, and (iii) the graph union process. Tarjan's algorithm and the graph coloring algorithm are executed once per iteration on each node of the network. The graph union, however, is executed once every time a neighboring node shares information. Each neighbor shares information once per iteration, so the complexity of the graph union procedure is multiplied by n_i^u, the number of neighbors node u has during iteration i.

Therefore, the general complexity equation for each node and each iteration is as follows:

$$TruMan = Tarjan + Coloring + (Union \times (n_i^u))$$

As discussed above, Tarjan's algorithm has a complexity of $O(|V| + |E|)$ and is executed on graph T. Meanwhile, the graph coloring algorithm has a complexity of $O(|E'|)$ and is executed on graph C. The complexity of the graph merge algorithm is $O(|E|)$ and, in a worst-case scenario, n_i^u is at most $|V|$ (every node in the network is a neighbor), resulting in a total complexity of $O(|V| \times |E|)$ for a whole iteration. The total complexity of each iteration of TruMan is, therefore:

$$O(|V| + |E|) + O(|E'|) + O(|V| \times |E|)$$

However, $|E'| \leq |E|$ is always true, because the graph C is a reduction of graph T. Furthermore, $|V| + |E| \leq |V| \times |E|$ is also true, except for the irrelevant scenarios of $|V| \leq 1$ or $|E| \leq 1$.

Therefore, the complexity can be simplified to: $O(|V| \times |E|)$.

3.7 Discussion

This chapter shows the concepts and algorithms that went into the development of TruMan. Using efficient algorithms as a foundation, TruMan is able to efficiently detect and identify malicious nodes operating within a dynamic network.

Although TruMan could be a viable solution for multiple kinds of networks, this work is focused primarily on vehicular networks and, therefore, TruMan was developed with such networks in mind. The following chapter explains how TruMan was evaluated in the context of vehicular networks, including results of the performed experiments.

Chapter 4

Evaluation of TruMan

In order to validate TruMan's functionality and efficiency, several simulations were executed, attempting to replicate real-world scenarios. This chapter includes all information relevant to these simulations, including the tools used, the chosen movement model, the parameters and methodology, and the results.

4.1 Tools

SNAP library

The simulations necessary to validate the project require a robust library to handle graph data structures. The Stanford Network Analysis Platform (SNAP) library [Leskovec, 2016] was chosen primarily because it is memory efficient. The simulations require multiple graphs that share the same set of nodes (because each node in the network has its own knowledge of the surrounding network), and the SNAP library uses pointers to nodes and edges, saving memory by not having to duplicate the entire data structure. It is written in C++, but, for these simulations, the Snap.py Python library was used.

The ONE Simulator

The Opportunistic Network Environment simulator [Keränen et al., 2009] [Keränen, 2015] is a Delay-Tolerant Network simulator, used in this study to generate mobility patterns used as input for simulations. It was chosen for the simulations of TruMan primarily because it already includes integration with the Working Day Movement Model.

The ONE Simulator comes with a usable map of the city of Helsinki, Finland, so the city was chosen as the map for the simulations of TruMan.

In order to use data from the ONE simulator as input for TruMan, a new report module had to be created for it. The `AdjacencySnapshotReport` module creates a report consisting of all adjacencies in the network every x number of simulated seconds. That is, at a given timestamp t, all pairs of nodes that are within communication range of each other are added to the report. This report is then used as input to TruMan, which uses the adjacencies to build the topology graph for each iteration of the algorithm. The `AdjacencySnapshotReport` module has been submitted to the ONE repository as a pull request [Greca, 2017].

4.2 Working Day Movement Model

Most VANET trust models use the Random Waypoint mobility model for simulations, i.e. each node has an origin point, chooses a random location, gets to that location, then chooses another random location and goes there, and so forth. While this model is efficient for testing trust protocols, it doesn't truly represent vehicle mobility in the real world.

To make use of the properties described in Section 3.2, it is important to choose a mobility pattern that properly represents the way vehicles move on a daily basis in the real world. Therefore, the Working Day Movement Model [Ekman et al., 2008] (WDMM) is useful. The model, developed for use in Delay-Tolerant Network (DTN) simulations, includes many of the features that are necessary to simulate the daily movement of a vehicular network.

As the name implies, the Working Day Movement Model abstracts people's movement from their homes to their offices and back. Each node has a home and a workplace and they need to travel back and forth between those locations on a daily basis. Occasionally, nodes can also go to other locations for leisure. As mentioned above, many drivers have routes they travel on daily, so the Working Day Movement Model is a reasonably accurate representation of daily movement in a city.

4.2.1 Original model

The Working Day Movement Model was developed for Delay-Tolerant Networks in which network members are devices (such as smartphones) carried by people. Therefore, the Working Day model represents not only people's movements inside their cars, but also within their offices, walking on foot, or riding a bus.

The model proposed by the authors makes use of several other models for specific tasks. The main mobility model places devices in the network and sets their destinations. Within it, five submodels are used:

1. The **home activity submodel** describes what devices do at night, within their owners' homes. No movement is modeled. Devices can belong to relatives or neighbors, and therefore share the same home location.

2. The **office activity submodel** describes the devices' movement routines within their owners' offices. Devices can move to other locations within the office (such as meeting rooms) and such movement is modeled. Devices whose owners are coworkers share the same office space.

3. The **evening activity submodel** is responsible for mobility outside the devices' owners' standard routines. Devices can be carried by people who meet at certain locations (such as restaurants) and spend a few hours with friends.

4. The **transport submodel** shows how devices move around the city. It includes another tier of submodels, responsible for modeling three different types of transportation: walking, driving, and riding a bus. People who own cars always use them, while the others can decide to walk or ride a bus depending on the distance between the origin and destination and the available bus stops. The walking and driving submodels represent similar types of movement, although at different speeds, while the bus submodel follows cyclical routes and can take or deliver passengers at bus stops.

5. The **map** represents the city in which the simulation runs. Its streets constrain the movement of devices, and all homes, offices and meeting spots must be within the map boundaries. The map can be divided into districts, which increases what the authors define as *locality*. It is possible to configure how many people work in the same district where they reside; devices carried by these people rarely leave their district. People who reside and work in different districts allow information to spread across different parts of the network by carrying their device with them.

4.2.2 Adaptation for a vehicular simulation

By thinking of these submodels for vehicles instead of people, it becomes apparent that the frequency and length of encounters between members of the network are similar in both instances. If two vehicles belong to family members or neighbors, they likely spend most of the night within communication distance, while coworkers' cars spend the office hours close by. Cars can also meet each other frequently if their drivers are friends who go out together after work. In the vehicular case, there is an added layer of encounters: cars can communicate frequently with buses and other cars that take the same route daily, even if their drivers are complete strangers.

To adapt the Working Day Movement Model to a VANET environment, a few changes had to be made so the network members are vehicles instead of people (or the devices they carry). Rather than altering the model itself, all of these changes were implemented as parameters for the model. The changes are as follows:

1. The office activity submodel no longer needs to model movement within the office and can be identical to the home submodel. In both, a node can move a small amount once after reaching the office or home, to simulate parking. This was done by setting the `officeSize` parameter to 1, so vehicles do not move around while their drivers are at work.

2. The walking submodel needs to be disabled, since all nodes are cars. By setting the `ownCarProb` parameter to 1, mobility is always done by car.

3. While the bus submodel could be used for a vehicular simulation, this was not used in this evaluation.

One important topic raised in the Working Day Movement Model article is the use of two metrics for a movement model: *inter-contact times* and *contact duration*. Inter-contact time is the average time it takes for two nodes to meet repeatedly in the network. For example, two vehicles who belong to neighbors might have an inter-contact time of about 12 hours, since that is how long they are apart before connecting again. Meanwhile, the contact duration is the time nodes spend connected when they do meet. In the case of the two vehicles owned by neighbors, the contact duration might also be about 12 hours, while their owners are at home and leave the vehicles close to each other.

The choice of the Working Day Movement Model for evaluations is more strongly related to inter-contact times. For reasons explained in Chapter 3, relatively short inter-contact times is important for TruMan's functionality. Contact duration time is an important metric to measure how much data can be exchanged during each encounter, although, for this evaluation, it was not considered.

4.3 Simulation parameters and methodology

In order to test the TruMan trust model, simulations were made using an implementation of the algorithm in Python. To generate the input graphs with node mobility, the ONE simulator [Keränen et al., 2009] was used in conjunction with the Working Day Movement Model [Ekman et al., 2008], which provides a strong similarity with vehicle movement in real life.

Snapshots of the network were taken every 10 simulated seconds using the `AdjacencySnapshotReport` module for the ONE simulator. However, a few experiments showed that it was not necessary to run iterations of TruMan that frequently; therefore, iterations run at an interval of 100 simulated seconds and only use one tenth of the snapshots saved.

Malicious nodes in the simulation misbehave by randomizing their opinions of neighbors. This means that an edge from a malicious node a to another node b is not reliable; its trust value can be anything regardless of the behavior of b. In collusion attacks, malicious nodes intentionally trust other malicious nodes, but such situations were not considered for the evaluation of TruMan.

Table 4.1: Simulation parameters

Parameter	**Value**
Duration	86400 seconds
Work day length	28800 seconds
Std. dev. departure time	7200 seconds
Node velocity	7 to 10 m/s
Simulation area	14,689,750 m^2
Number of nodes	150 (WDMM) + 10 (random)
Office size	1

Most of the parameters for the ONE simulator were taken from the article detailing the Working Day Movement Model [Ekman et al., 2008]; the most important parameters are shown in Table 4.1. The simulation ran for 86400 seconds (24 hours), with a work day length of 28800 seconds (8 hours) and a standard deviation of departure time of 7200 seconds (2 hours). Nodes move between 7 and 10 m/s in an area of approximately 14.7 km^2 based on a section of the map of Helsinki.

There is a total of 160 nodes, 150 of which follow the Working Day Movement Model, and 10 that follow the random waypoint mobility model to simulate vehicles that do not follow daily patterns. Since this simulation is for vehicles instead of pedestrians, there are no buses in the model and every node is guaranteed to own a vehicle and travel by car. This configuration was chosen in order to make the network homogenous, but buses and pedestrians could be considered as an addition to the scope.

Aside from the office size parameter, which is 1 in order to inhibit in-office mobility, the parameters regarding offices, meeting spots and shopping were kept intact. A small part of nodes move randomly to simulate vehicles that do not follow daily patterns. The transmission range of nodes varies from simulation to simulation, for reasons explained in Section 4.3.1.

4.3.1 Network Density

The communication range of nodes varies from 10m to 50m, to illustrate the impact of different network densities. The network density (δ) is a value which abstracts the volume and frequency of connections in a vehicular network by estimating how much of the environment is covered by the network. For TruMan, higher densities yield better results, since nodes can construct and update their models of the network faster (this is demonstrated in Chapter 4). It is calculated using the transmission range of the nodes (ρ), the amount of nodes (η), and the total area of the simulation (α, in m^2).

The coverage of a single node is the circumference around it formed by the transmission radius. This is divided by two to compensate for overlapping circumferences, then multiplied by the number of nodes in the network to estimate the maximum coverage area. Finally, the value is divided by the total area of the environment. The network density formula is as follows:

$$\delta = \frac{\frac{\rho^2\pi}{2} \times \eta}{\alpha}$$

For example, in a simulation with ρ = 30m, the calculation is as follows:

$$\delta = \frac{\frac{30^2\pi}{2} \times 160}{14,689,750} = 0.0154$$

In Table 4.2, a few densities for different values of ρ, η and α are shown. Simulations for TruMan have densities between 0.0017 (ρ = 10m) and 0.0428 (ρ = 50m).

Table 4.2: Simulation densities

Range (ρ)	**Nodes (η)**	**Area (α)**	**Density(δ)**
10 m	160	14,689,750	0.0017
30 m	160	14,689,750	0.0154
50 m	160	14,689,750	0.0428
100 m	160	14,689,750	0.1604
150 m	160	14,689,750	0.3609
200 m	160	14,689,750	0.6416

Table 4.3: Calculated densities of major cities

City (country)	**Nodes (η)**	**Area (α)**	**Density (δ)**
Helsinki (FI)	250,000 [Helsinki, 2011]	214,250,000 [NLSF, 2018]	0.1833
São Paulo (BR)	8,603,239 [Detran SP, 2017]	1,521,110,000 [IBGE, 2016]	0.8884
New York (US)	2,162,349 [NY DMV, 2016]	777,934,030 [US Census, 2017]	0.4366
Los Angeles (US)	8,050,850 [CA DMV, 2016]	1,213,820,883 [US Census, 2017]	1.041
Tokyo (JP)	3,159,455 [Statistics Japan, 2017]	2,191,000,000 [Tokyo MG, 2017]	0.2265
London (UK)	3,091,393 [UK DfT, 2016]	1,572,100,000 [GLA, 2011]	0.3089

As a comparison, the city of São Paulo (Brazil) has a fleet of over 8 million vehicles [Detran SP, 2017] in an area of 1,521.11 km^2 [Instituto Brasileiro de Geografia e Estatística, 2016], and thus has a density of 0.8884 at ρ = 10m, a much higher value than what is necessary for a satisfactory performance of the algorithm. Table 4.3 shows the densities of a few major cities around the world, using $\rho = 10$m for all of them. All data is taken from local government sources regarding the number of licensed vehicles in each city; these numbers do not include vehicles from the larger metropolitan areas that surround these cities.

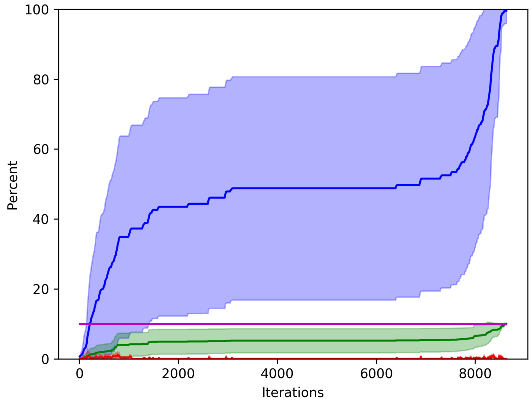
Simulations of TruMan were performed with densities as low as 0.0017, a value much lower than even the real world density of Helsinki, which is a city with relatively few vehicles for its size. It can be expected that the model will perform even better in real-world scenarios in cities with even higher densities, which is common, as exemplified by Table 4.3.

4.4 Results

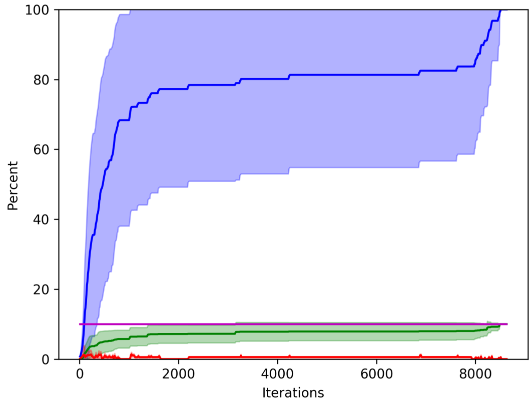
To improve readability, all figures in this section follow the same format:

- The X axis shows the results of sequential iterations, ranging from 0 to 8639 in most cases;
- The Y axis shows a percentage of all nodes in the network, ranging from 0 to 100;
- The blue line represents the percentage of nodes detected out of the complete network;
- The magenta line is the percentage of malicious nodes in the network (ground truth);
- The green line represents the nodes correctly identified as malicious (true positives);
- When present, the cyan line represents the undetected malicious nodes (false negatives);
- The red line represents the benign nodes incorrectly identified as malicious (false positives);
- The lines represent average values between all benign nodes on the network, while the colored regions represent the standard deviation present in the data.

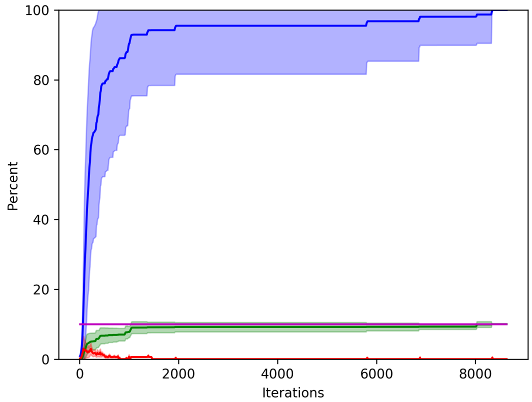
It is expected that the results are very inconsistent at the beginning of the simulations, since nodes are still building their models of the network and have relatively little information to use.



(a) ρ = 10m.



(b) ρ = 30m.



(c) ρ = 50m.

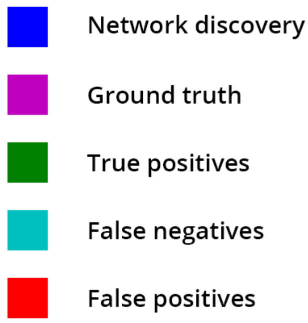
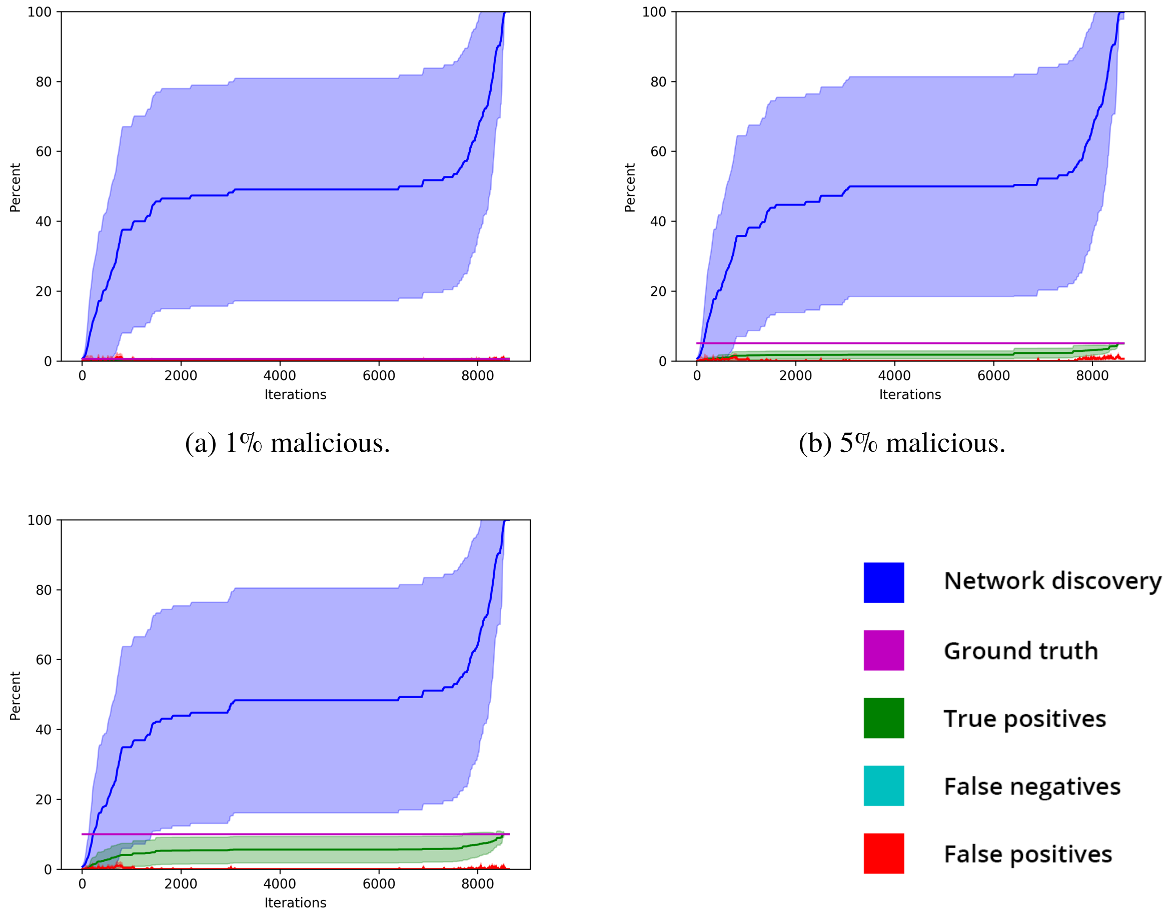


Figure 4.1: Simulation of TruMan with 10% malicious nodes and varying values of ρ.

Figure 4.1 shows the results of simulations running with 10% of nodes acting maliciously, with communications range varying from 10m to 50m. It is possible to see how the increase in the range allows the algorithm to produce better results. At ρ = 10m and δ = 0.0017, a large amount of time is spent with only about half of malicious nodes being accurately identified. Results with ρ = 30m and δ = 0.0153 are better, but still less than ideal; during this simulation, there were more false positive detections happening, although it was still a small amount. At ρ = 50m and δ = 0.0427, results are solid before the 2000th iteration of the algorithm, since at that point over 90% of the network has been acknowledged by most nodes. The amount of false positive detections is extremely low, and, by the end of the simulation, all benign nodes have identified 100% of malicious nodes.



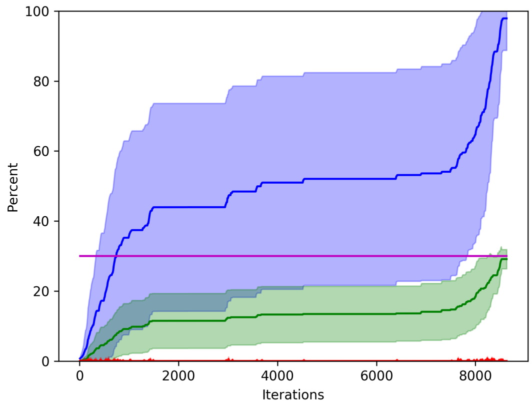
(a) 1% malicious.

(b) 5% malicious.

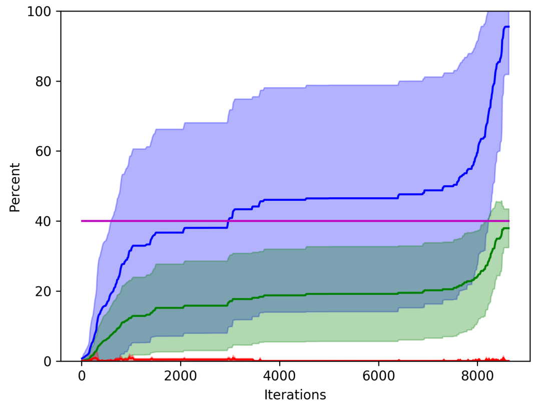
(c) 10% malicious.

Figure 4.2: Simulation of TruMan with ρ = 10m and varying percentages of malicious nodes (1%, 5% and 10%).

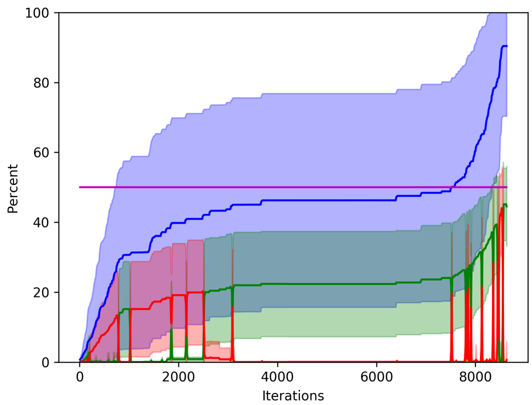
Figure 4.2 and Figure 4.3 show the variation of results for different amounts of malicious nodes in the network. By the end of one day, the algorithm is able to detect all malicious nodes when they are up to 30% of the network, although there is still a small amount of false positive detections. At 40%, a small part of malicious nodes are yet to be detected and the standard deviation is larger overall. At 50%, as expected, the results are inconsistent as the network is completely split between benign and malicious nodes; at this point, the network is considered completely compromised. True positive and false positive detections are often inverted, because whenever there are more malicious nodes than benign ones in a certain node's network model, the algorithm will classify the malicious ones as correct. The amount of malicious nodes also affects how quickly nodes are able to acquire information about the network, since nodes do not trust information from malicious neighbors.



(a) 30% malicious.

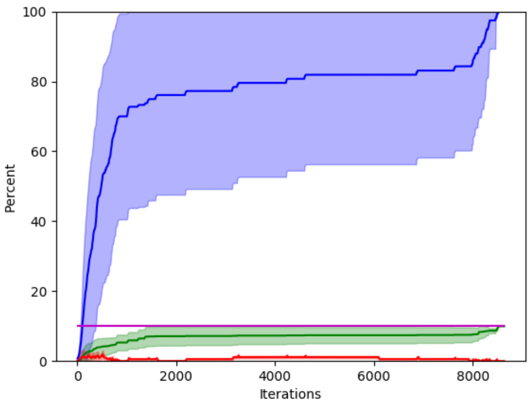


(b) 40% malicious.

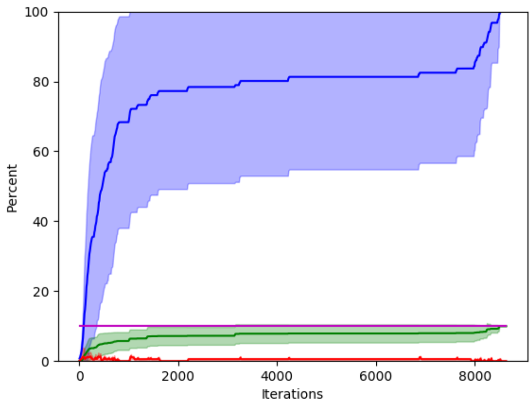


(c) 50% malicious.

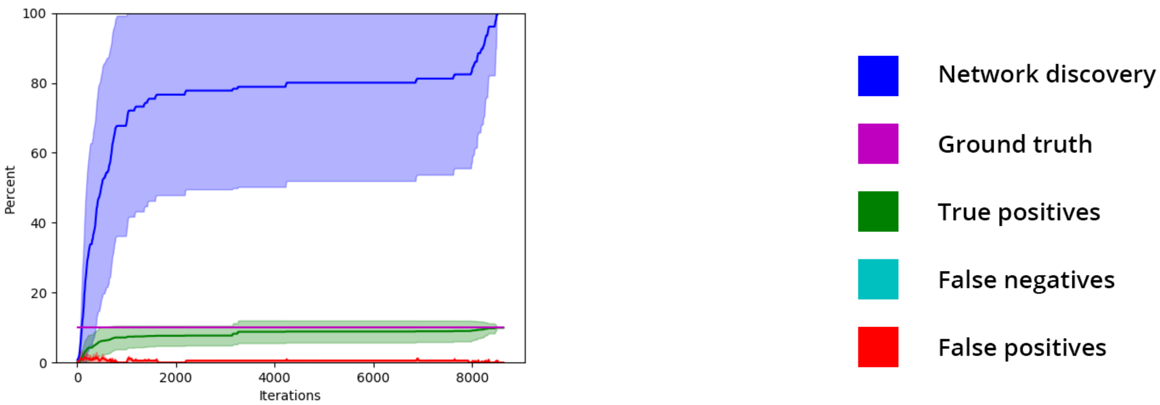
Figure 4.3: Simulation of TruMan with ρ = 10m and varying percentages of malicious nodes (30%, 40% and 50%).



(a) h = 0.3.



(b) h = 0.5.



(c) h = 0.7.

Figure 4.4: Simulation of TruMan with 10% malicious nodes, ρ = 30m and varying values of h.

All simulations were performed with trust threshold h = 0.5, except for the ones in Figure 4.4, performed to demonstrate the impact of different threshold values. The plots illustrate that there is not a significance change in results depending on the different thresholds. The results are slightly better at h = 0.7, however not in a significant way. It still takes over 8000 iterations to detect all malicious nodes and there are still some false positive detections.

Figure 4.5 shows the execution of the algorithm over the course of 7 days. Most malicious nodes are identified by the end of the first day, before iteration 9000. However, not all nodes have finished building their model of the network and therefore there are still a number of false positive detections occurring. Around iteration 20000, all nodes finish building their models and the false positive detection rate drops to an insignificant amount.

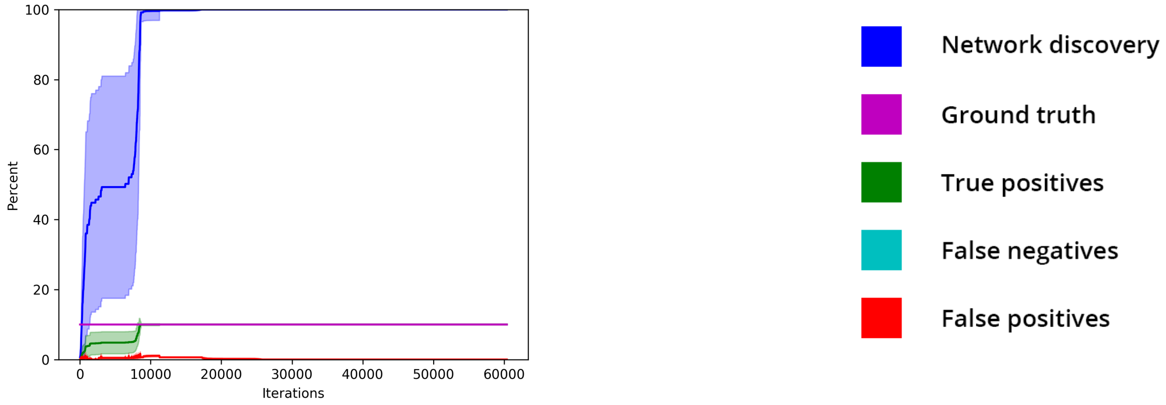
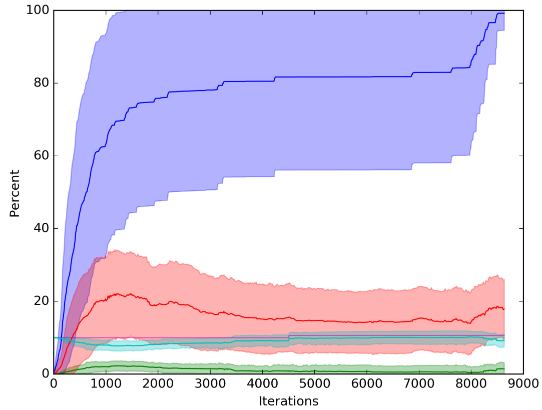
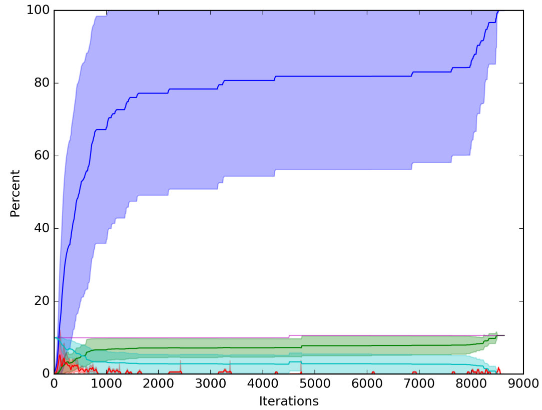


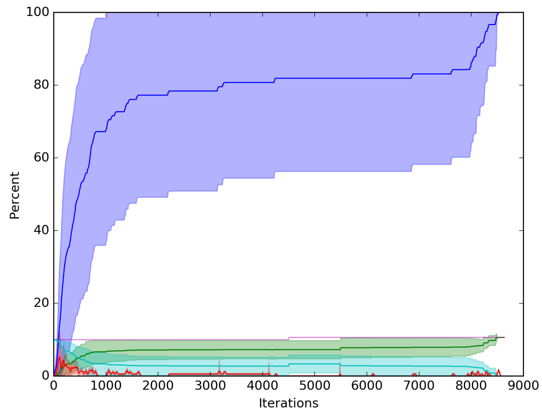
Figure 4.5: 7 days scenario: 10m range and 10% malicious nodes.



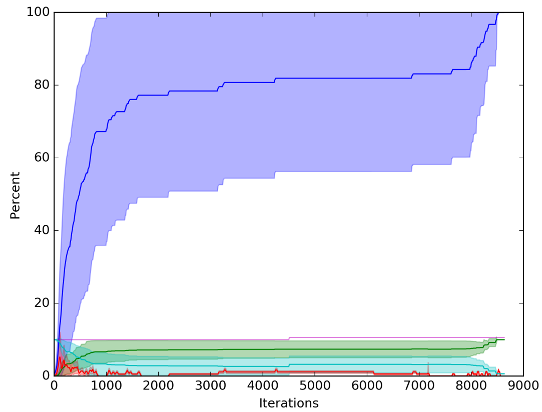
(a) m = 10.



(b) m = 250.



(c) m = 1000.



(d) m = 5000.

Figure 4.6: Simulation with information aging, with different maximum age values (m = 1000, 5000).

Figure 4.6 shows how TruMan reacts to one node converting from benign to malicious. These were the only simulations to consider information aging. Here, several different values for the maximum age, m, were tested. For the simulation, information timestamps are set as the iteration in which they were added to the graph ($s \leftarrow i$), and the age is calculated as the difference between the current iteration and the timestamp ($age \leftarrow i - s$). With $m = 10$, information is discarded too quickly, resulting in highly imprecise results. Starting with $m = 250$, results are reasonable, and the additional malicious node is detected approximately 250 iterations after it converts. With higher values such as $m = 1000$, detection takes longer, but not much else changes. When the value is very high, like in the case of $m = 5000$, the extra malicious node is not detected by the end of one simulated day.

4.5 Satisfaction of desired properties

Eight desired properties for VANET trust models were presented in Section 2.4.1. Here, TruMan's ability to satisfy each property is evaluated. Table 4.4 replicates the one from Section 2.4.1 with the addition of TruMan, providing a full comparison between itself and the related work.

1. Decentralized trust establishment: TruMan is built from the ground up for decentralized systems. Nodes form their own abstractions of the network and the model does not rely on a central observer.

2. Coping with sparsity: The experiments using low density values demonstrate that TruMan works in reasonably sparse networks. Since nodes carry information from previous interactions, the model can also work on temporarily isolated parts of the network.

3. Event/task and location/time dynamics: TruMan can be extended to consider event, task, location and time dynamics when managing trust. However, aside from the time-related information used for information aging, dynamics were not used in the simulations presented here.

4. Scalability: Due to the low complexity of the algorithms used in the model, TruMan is highly scalable, as it does not incur substantial pressure on the vehicles' on-board units. It has also been demonstrated that iterations of the algorithm do not need to run very frequently in order to detect malicious nodes with high accuracy — once every ten seconds is more than enough.

5. Integrated confidence measure: Since nodes using TruMan store trust values as a number between 0 and 1 that increase or decrease the more nodes interact (i.e. the more evidence they gather), this value can be used as a confidence measure of the opinion. Setting the threshold h to a value closer to 1 makes it unlikely that malicious nodes will be incorrectly labeled as benign.

6. System level security: A public-private key solution can be used to verify message integrity, although it is not a requirement for the functioning of TruMan. Any cryptographic solution can be integrated as a separate security model during the transmission of messages without affecting TruMan's efficiency and efficacy.

7. Sensitivity to privacy concerns: TruMan has not been designed with this in mind, as it requires that nodes maintain a constant identity the whole time. However, this does not inhibit other types of privacy protection, which can be implemented in addition to TruMan.

8. Robustness: TruMan satisfies this property, because malicious nodes are quickly and accurately identified, making it difficult for them to perform attacks. Experiments show that, when fewer than 50% of nodes in the network are malicious, TruMan performs as expected. Collusion attacks must be performed by more than half of the entire network, in which case the network is considered compromised. Furthermore, since nodes take into consideration experiences from several trustworthy nodes, a malicious node that occasionally behaves correctly can still be identified. Experiments with information aging show that it is possible to detect nodes that suddenly become malicious, although further experimentation with known attacks is still necessary.

Finally, it is worth noting that, considering the scale of the problem, TruMan's cost is very low without sacrificing completeness and correctness. The model satisfies or permits most of the desired properties of a trust model with low computational complexity, making it viable for real-world use.

Table 4.4: Satisfaction of desired properties compared to related work

Property	1	2	3	4	5	6	7	8
TruMan	✓	✓	✓	✓	✓	-	-	✓
[Dotzer et al., 2005]	✓	-	✓	-	-	-	✓	-
[Minhas et al., 2010]	✓	✓	✓	✓	✓	✓	✓	-
[Chen et al., 2010]	✓	✓	✓	✓	✓	✓	-	-
[Park et al., 2011]	-	-	-	-	-	✓	✓	✓
[Huang et al., 2014]	✓	-	✓	✓	✓	-	-	-
[Li and Song, 2016]	✓	-	-	-	✓	-	-	✓
[Chen and Wang, 2017]	-	✓	-	✓	-	✓	✓	-

Desired properties

1. Decentralized trust establishment
2. Coping with sparsity
3. Event/task and location/time dynamics
4. Scalability
5. Integrated confidence measure
6. System level security
7. Sensitivity to privacy concerns
8. Robustness

Chapter 5

Conclusion

In the coming years, vehicular networks or VANETs will be an important part of safety and security in transportation, optimizing traffic and reducing the number of accidents. However, they are an appealing target for entities with malicious intents, creating the need of robust solutions to maintain the integrity of such networks.

The concept of trust as applied in VANETs is a powerful tool for those seeking to reduce the spread of false information among members of a network as much as possible. In this paper, a new trust model for vehicular networks called TruMan was introduced, which combines the efficiency of previously proposed algorithms in order to generate fast and accurate results. The solution works in a decentralized fashion and is built for the dynamic environment of vehicular networks, although it could also be adapted to other types of networks.

As nodes travel across the network and collect more data from neighbors, they are able to form an abstraction of the network which can be used to detect malicious nodes. By placing nodes into strongly connected components, a network containing a large amount of nodes can be simplified into a much smaller one. Using a simple graph coloring algorithm, most malicious nodes stand out by having different colors than the majority of nodes. This allows for a low complexity approach to malicious node identification in a dynamic network.

TruMan was evaluated using mobility data gathered from the ONE simulator using the Working Day Movement Model, which approximates node mobility to that of real-world vehicles. Several simulations were performed, changing certain parameters to understand how the model performs in different scenarios. The experiments show that vehicles within a network can form a sufficient abstraction of the network in around one day, and with that information they are able to detect nearly every malicious node in the network, with a very small amount of false positives. As the network changes in shape, nodes acquire more information and are able to make even more accurate classifications of malicious nodes around them. With the implementation of information aging, TruMan is also able to detect nodes that start benign and become malicious during the simulation.

In comparison with the related work, TruMan was able to satisfy most of the desired properties for vehicular network trust models, while not inhibiting the properties that were not desired. Most importantly, TruMan put emphasis on efficiency and is the first model that clearly displays the complexity of its algorithm. Furthermore, TruMan begins taking advantage of social network features found in vehicular networks, although more can be done with this idea.

The work done on TruMan was published as a conference paper on the 2018 IEEE Symposium on Computers and Communications [Greca and Albini, 2018].

Several paths could be considered for future work on TruMan, such as:

- TruMan could be tested in more varied scenarios, using different maps and various different amounts of nodes in the simulations. It would be even better to use real-world mobility data.
- TruMan takes advantage of social features found in vehicular networks, but even more could be done with this. For example, vehicles that belong to the same family could have strong ties and share a lot of data with each other. Certain vehicles, such as police cars and ambulances, could have privileged roles within the model.
- The model could be adapted to include public transportation vehicles (trains, buses) with predictable routes, as well as vehicle-to-infrastructure communication.
- Well-known vehicular network attacks, such as the ones in [Isaac et al., 2010], could be used against TruMan, attempting to break it. Such experiments would fully validate TruMan's robustness.
- TruMan should be tested in real-world networks. Although it was designed for vehicular networks, other types of networks with mobility, such as mobile ad-hoc networks, could be used for experiments.
- Finally, it is possible that TruMan might be a valuable tool for more than just vehicular network. A decentralized and dynamic trust management scheme could be useful for social networks, mobile ad-hoc networks, Internet-based peer-to-peer networks, and others. These scenarios should be tested.

Bibliography

Mani Amoozadeh, Hui Deng, Chen-Nee Chuah, H Michael Zhang, and Dipak Ghosal. Platoon management with cooperative adaptive cruise control enabled by vanet. *Vehicular communications*, 2(2):110–123, 2015.

Kenneth Appel, Wolfgang Haken, and John Koch. Every planar map is four colorable. *Bull. Amer. Math. Soc*, 82(5):711–712, 1976.

John S Baras and Tao Jiang. Cooperation, trust and games in wireless networks. In *Advances in Control, Communication Networks, and Transportation Systems*, pages 183–202. Springer, 2005.

Carolina Tripp Barba, Miguel Angel Mateos, Pablo Reganas Soto, Ahmad Mohamad Mezher, and Mónica Aguilar Igartua. Smart city for vanets using warning messages, traffic statistics and intelligent traffic lights. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 902–907. IEEE, 2012.

Jeremy Boissevain. *Friends of friends: Networks, manipulators and coalitions*. Blackwell Oxford, 1974.

Daniel Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22 (4):251–256, 1979.

Cadillac Pressroom. V2V Safety Technology Now Standard on Cadillac CTS Sedans. `http://media.cadillac.com/media/us/en/cadillac/news.detail.html/content/Pages/news/us/en/2017/mar/0309-v2v.html`, 2017. [Online; accessed June 21, 2018].

California State Department of Motor Vehicles. Estimated Vehicles Registered by County. `https://www.dmv.ca.gov/portal/wcm/connect/add5eb07-c676-40b4-98b5-8011b059260a/est_fees_pd_by_county.pdf?MOD=AJPERES`, 2016. [Online; accessed February 28, 2018].

CAMP Vehicle Safety Communications Consortium. Vehicle safety communications project: Task 3 final report: identify intelligent vehicle safety applications enabled by dsrc. *National Highway Traffic Safety Administration, US Department of Transportation, Washington DC*, 2005.

Chen Chen, Jie Zhang, Robin Cohen, and Pin-Han Ho. A trust modeling framework for message propagation and evaluation in vanets. In *Information Technology Convergence and Services (ITCS), 2010 2nd International Conference on*, pages 1–8. IEEE, 2010.

Xiao Chen and Liangmin Wang. A cloud-based trust management framework for vehicular social networks. *IEEE Access*, 5:2967–2980, 2017.

Chien-Ming Chou, Chen-Yuan Li, Wei-Min Chien, and Kun-chan Lan. A feasibility study on vehicle-to-infrastructure communication: Wifi vs. wimax. In *Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on*, pages 397–398. IEEE, 2009.

howpublished = "https://web.archive.org/web/20111211115004/http://www.hel2.fi/tietoa/helbro1.pdf" year = 2011 note = "[Archived online; accessed March 29 2018]" City of Helsinki, title = This is Helsinki.

Felipe D Cunha, Aline Carneiro Vianna, Raquel AF Mini, and Antonio AF Loureiro. How effective is to look at a vehicular network under a social perception? In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on*, pages 154–159, 2013.

Felipe D Cunha, Aline Carneiro Vianna, Raquel AF Mini, and Antonio AF Loureiro. Is it possible to find social properties in vehicular networks? In *2014 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2014a.

Felipe D Cunha, Aline Carneiro Vianna, Raquel AF Mini, and Antonio AF Loureiro. Are vehicular networks small world? In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 195–196. IEEE, 2014b.

Detran SP. Frota de Veículos em SP. https://www.detran.sp.gov.br/wps/wcm/connect/portaldetran/detran/detran/estatisticastransito/sa-frotaveiculos/, 2017. [Online; accessed February 28, 2018].

Qing Ding, Xi Li, M Jiang, and X Zhou. A novel reputation management framework for vehicular ad hoc networks. *International Journal of Multimedia Technology*, 3(2):62–66, 2013.

Florian Dotzer, Lars Fischer, and Przemyslaw Magiera. Vars: A vehicle ad-hoc network reputation system. In *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*, pages 454–456. IEEE, 2005.

Sudha Dwivedi and Rajni Dubey. Review in trust and vehicle scenario in vanet. *International Journal of Future Generation Communication and Networking*, 9(5):305–314, 2016.

Frans Ekman, Ari Keränen, Jouni Karvo, and Jörg Ott. Working day movement model. In *Proceedings of the 1st ACM SIGMOBILE workshop on Mobility models*, pages 33–40. ACM, 2008.

Mevlut Turker Garip, Mehmet Emre Gursoy, Peter Reiher, and Mario Gerla. Congestion attacks to autonomous cars using vehicular botnets. In *NDSS Workshop on Security of Emerging Networking Technologies (SENT), San Diego, CA*, 2015.

Matthias Gerlach. Trust for vehicular applications. In *Autonomous Decentralized Systems, 2007. ISADS'07. Eighth International Symposium on*, pages 295–304. IEEE, 2007.

Jennifer Golbeck and James Hendler. Inferring binary trust relationships in web-based social networks. *ACM Transactions on Internet Technology (TOIT)*, 6(4):497–529, 2006.

Philippe Golle, Dan Greene, and Jessica Staddon. Detecting and correcting malicious data in vanets. In *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, pages 29–37. ACM, 2004.

Greater London Authority. Land Area and Population Density, Ward and Borough. `https://data.london.gov.uk/dataset/land-area-and-population-density-ward-and-borough`, 2011. [Online; accessed February 28, 2018].

Renan Greca. Pull request: Created AdjacencySnapshotReport. `https://github.com/akeranen/the-one/pull/56`, 2017. [Online; accessed February 24, 2018].

Renan Greca and Luiz CP Albini. Truman: Trust management for vehicular networks. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2018.

Nadia Haddadou, Abderrezak Rachedi, and Yacine Ghamri-Doudane. Trust and exclusion in vehicular ad hoc networks: an economic incentive model based approach. In *Computing, Communications and IT Applications Conference (ComComAp), 2013*, pages 13–18. IEEE, 2013.

MR Hafner, D Cunningham, L Caminiti, and D Del Vecchio. Automated vehicle-to-vehicle collision avoidance at intersections. In *Proceedings of world congress on intelligent transport systems*, 2011.

Dijiang Huang, Xiaoyan Hong, and Mario Gerla. Situation-aware trust architecture for vehicular networks. *IEEE Communications Magazine*, 48(11), 2010.

Zhen Huang, Sushmita Ruj, Marcos A Cavenaghi, Milos Stojmenovic, and Amiya Nayak. A social network approach to trust management in vanets. *Peer-to-Peer Networking and Applications*, 7 (3):229–242, 2014.

IEEE Connected Vehicles. First Toyota cars to include V2V and V2I communication by the end of 2015. `http://sites.ieee.org/connected-vehicles/2015/09/30/first-toyota-cars-to-include-v2v-and-v2i-communication-by-the-end-of-2015/`, 2015. [Online; accessed June 21, 2018].

INRIX. Los Angeles Tops INRIX Global Congestion Ranking. `http://inrix.com/press-releases/los-angeles-tops-inrix-global-congestion-ranking/`, 2017. [Online; accessed March 27, 2017].

Instituto Brasileiro de Geografia e Estatística. Áreas dos Municípios - São Paulo. `https://www.ibge.gov.br/geociencias-novoportal/organizacao-do-territorio/estrutura-territorial/2225-np-areas-dos-municipios/15761-areas-dos-municipios.html?&t=destaques`, 2016. [Online; accessed February 28, 2018].

Insurance Institute for Highway Safety. Overview of Fatality Facts. `http://www.iihs.org/iihs/topics/t/general-statistics/fatalityfacts/overview-of-fatality-facts`, 2016. [Online; accessed June 21, 2018].

Jesús Téllez Isaac, Sherali Zeadally, and José Sierra Camara. Security attacks and solutions for vehicular ad hoc networks. *IET communications*, 4(7):894–903, 2010.

Daniel Jiang and Luca Delgrossi. Ieee 802.11 p: Towards an international standard for wireless access in vehicular environments. In *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 2036–2040. IEEE, 2008.

Teddi Dineley Johnson. U.S. traffic deaths drop to lowest level since 1949. http://thenationshealth.aphapublications.org/content/41/4/E17.full, 2010. [Online; accessed March 27, 2017].

Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

Alfred B Kempe. On the geographical problem of the four colours. *American journal of mathematics*, 2(3):193–200, 1879.

Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The one simulator for dtn protocol evaluation. In *Proceedings of the 2nd international conference on simulation tools and techniques*, page 55. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

Chaker Abdelaziz Kerrache, Abderrahmane Lakas, and Nasreddine Lagraa. Detection of intelligent malicious and selfish nodes in vanet using threshold adaptive control. In *Electronic Devices, Systems and Applications (ICEDSA), 2016 5th International Conference on*, pages 1–4. IEEE, 2016.

Ari Keränen. The ONE. http://akeranen.github.io/the-one/, 2015. [Online; accessed March 27, 2017].

Florian Knorr, Daniel Baselt, Michael Schreckenberg, and Martin Mauve. Reducing traffic jams via vanets. *IEEE Transactions on Vehicular Technology*, 61(8):3490–3498, 2012.

M.J. Krochmal, C.J. Edmonds, C.C. Jensen, and A. Prats. Discovery of nearby devices for file transfer and other communications, December 11 2014. URL https://www.google.com.br/patents/US20140362728. US Patent App. 14/037,272.

John D Lee, Joshua D Hoffman, and Elizabeth Hayes. Collision warning design to mitigate driver distraction. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, pages 65–72. ACM, 2004.

Tim Leinmüller, Elmar Schoch, Frank Kargl, and Christian Maihöfer. Influence of falsified position data on geographic ad-hoc routing. In *European Workshop on Security in Ad-hoc and Sensor Networks*, pages 102–112. Springer, 2005.

Jure Leskovec. Stanford Network Analysis Project. http://snap.stanford.edu, 2016. [Online; accessed November 30, 2017].

Fan Li and Yu Wang. Routing in vehicular ad hoc networks: A survey. *IEEE Vehicular technology magazine*, 2(2), 2007.

Wenjia Li and Houbing Song. Art: An attack-resistant trust management scheme for securing vehicular ad hoc networks. *IEEE Transactions on Intelligent Transportation Systems*, 17(4): 960–969, 2016.

Wenjie Li, Francesca Bassi, Davide Dardari, Michel Kieffer, and Gianni Pasolini. Defective sensor identification for wsns involving generic local outlier detection tests. *IEEE transactions on Signal and Information Processing over Networks*, 2(1):29–48, 2016.

Zhiquan Liu, Jianfeng Ma, Zhongyuan Jiang, Hui Zhu, and Yinbin Miao. Lsot: A lightweight self-organized trust model in vanets. *Mobile Information Systems*, 2016, 2016.

Shuo Ma, Ouri Wolfson, and Jie Lin. A survey on trust management for intelligent transportation system. In *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, pages 18–23. ACM, 2011.

Maanmittauslaitos. National Land Survey of Finland. `http://www.maanmittauslaitos.fi/sites/maanmittauslaitos.fi/files/attachments/2018/01/Suomen_pa_2018_kunta_maakunta.pdf`, 2018. [Online; accessed March 29, 2018].

Thomas Mangel, Timo Kosch, and Hannes Hartenstein. A comparison of umts and lte for vehicular safety communication at intersections. In *Vehicular Networking Conference (VNC), 2010 IEEE*, pages 293–300. IEEE, 2010.

Michael McCole. How to Make the Amazon Echo the Center of Your Smart Home. `https://www.wired.com/2016/01/iot-cookbook-amazon-echo/`, 2016. [Online; accessed April 20, 2017].

Mohamed Nidhal Mejri, Jalel Ben-Othman, and Mohamed Hamdi. Survey on vanet security challenges and possible cryptographic solutions. *Vehicular Communications*, 1(2):53–66, 2014.

Umar Farooq Minhas, Jie Zhang, Thomas Tran, and Robin Cohen. Towards expanded trust management for agents in vehicular ad-hoc networks. *International Journal of Computational Intelligence: Theory and Practice (IJCITP)*, 5(1):03–15, 2010.

Amit Mittal, Parash Jain, Srushti Mathur, and Preksha Bhatt. Graph coloring with minimum colors: An easy approach. In *Communication Systems and Network Technologies (CSNT), 2011 International Conference on*, pages 638–641. IEEE, 2011.

Jacob Morgan. A Simple Explanation Of 'The Internet Of Things'. `https://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/`, 2014. [Online; accessed April 20, 2017].

New York State Department of Motor Vehicles. Vehicle Registrations in Force. `https://dmv.ny.gov/statistic/2016reginforce-web.pdf`, 2016. [Online; accessed February 28, 2018].

Mark Newman. *Networks: an introduction*. Oxford University Press, 2010.

Soyoung Park, Baber Aslam, and Cliff C Zou. Long-term reputation system for vehicular networking based on vehicle's daily commute routine. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pages 436–441. IEEE, 2011.

Anand Patwardhan, Anupam Joshi, Tim Finin, and Yelena Yesha. A data intensive reputation management scheme for vehicular ad hoc networks. In *Mobile and Ubiquitous Systems: Networking & Services, 2006 Third Annual International Conference on*, pages 1–8. IEEE, 2006.

Maxim Raya, Panagiotis Papadimitratos, Virgil D Gligor, and J-P Hubaux. On data-centric trust establishment in ephemeral ad hoc networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 1238–1246. IEEE, 2008.

Real-Time Innovations. How the internet of things can save 50,000 lives a year. 2014.

Mukesh Saini, Abdulhameed Alelaiwi, and Abdulmotaleb El Saddik. How close are we to realizing a pragmatic vanet solution? a meta-survey. *ACM Computing Surveys (CSUR)*, 48(2): 29, 2015.

Abdón Sánchez-Arroyo. Determining the total colouring number is np-hard. *Discrete Mathematics*, 78(3):315–319, 1989.

Tetsuya Sasaki and Masato Kuwahara. Wireless network system and wireless communication program, December 6 2011. US Patent 8,073,923.

Jitendra Singh Sengar. Survey: Reputation and trust management in vanets. *International Journal of Grid and Distributed Computing*, 9(1):201–206, 2016.

Wanita Sherchan, Surya Nepal, and Cecile Paris. A survey of trust in social networks. *ACM Computing Surveys (CSUR)*, 45(4):47, 2013.

Seyed Ahmad Soleymani, Abdul Hanan Abdullah, Wan Haslina Hassan, Mohammad Hossein Anisi, Shidrokh Goudarzi, Mir Ali Rezazadeh Baee, and Satria Mandala. Trust management in vehicular ad hoc network: a systematic review. *EURASIP Journal on Wireless Communications and Networking*, 2015(1):146, 2015.

Statistics Japan. Automobiles Registered. `http://stats-japan.com/t/kiji/10786`, 2017. [Online; accessed February 28, 2018].

Jack Stewart. Tesla's Self-Driving Car Plan Seems Insane, But It Just Might Work. `https://www.wired.com/2016/10/teslas-self-driving-car-plan-seems-insane-just-might-work/`, 2016. [Online; accessed April 11, 2017].

Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2): 146–160, 1972.

Tokyo Metropolitan Government. Geography of Tokyo. `http://www.metro.tokyo.jp/ENGLISH/ABOUT/HISTORY/history02.htm`, 2017. [Online; accessed February 28, 2018].

United Kingdom Department for Transport. Licensed Vehicles - Type, Borough. `https://data.london.gov.uk/dataset/licensed-vehicles-type-0`, 2016. [Online; accessed February 28, 2018].

United States Census Bureau. 2017 U.S. Gazetteer Files. `https://www.census.gov/geo/maps-data/data/gazetteer2017.html`, 2017. [Online; accessed February 28, 2018].

Grazielle Vernize. *Identificação de nós maliciosos em redes complexas baseada em visões locais*. MSc dissertation, Universidade Federal do Paraná, 2013.

Grazielle Vernize, André Luiz Pires Guedes, and Luiz Carlos Pessoa Albini. Malicious nodes identification for complex network based on local views. *The Computer Journal*, 58(10): 2476–2491, 2015.

Richard Viereckl, Dietmar Ahlemann, Alex Koster, Evan Hirsh, Felix Kuhnert, Joachim Mohs, Marco Fischer, Walter Gerling, Kaushik Gnanasekaran, and Julia Kusb. Connected car report 2016: Opportunities, risk, and turmoil on the road to autonomous vehicles. `http://www.strategyand.pwc.com/reports/connected-car-2016-study`, 2016. [Online; accessed April 11, 2017].

Jian Wang, Yanheng Liu, Xiaomin Liu, and Jing Zhang. A trust propagation scheme in vanets. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 1067–1071. IEEE, 2009.

Albert Wasef, Rongxing Lu, Xiaodong Lin, and Xuemin Shen. Complementing public key infrastructure to secure vehicular ad hoc networks [security and privacy in emerging wireless networks]. *IEEE Wireless Communications*, 17(5), 2010.

World Health Organization. Number of road traffic deaths. `http://www.who.int/gho/road_safety/mortality/traffic_deaths_number/en/`, 2013. [Online; accessed March 27, 2017].

World Health Organization. Road traffic injuries fact sheet. `http://www.who.int/mediacentre/factsheets/fs358/en/`, 2015. [Online; accessed March 27, 2017].

Jie Wu and Ivan Stojmenovic. Ad hoc networks. *COMPUTER-IEEE COMPUTER SOCIETY-*, 37(2):29–31, 2004.

Xue Yang, L Liu, Nitin H Vaidya, and Feng Zhao. A vehicle-to-vehicle communication protocol for cooperative collision warning. In *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, pages 114–123. IEEE, 2004.

Saleh Yousefi, Mahmoud Siadat Mousavi, and Mahmood Fathy. Vehicular ad hoc networks (vanets): challenges and perspectives. In *ITS Telecommunications Proceedings, 2006 6th International Conference on*, pages 761–766. IEEE, 2006.

Jie Zhang. A survey on trust management for vanets. In *2011 IEEE International Conference on Advanced Information Networking and Applications*, pages 105–112. IEEE, 2011.

Jie Zhang. Trust management for vanets: challenges, desired properties and future directions. *International Journal of Distributed Systems and Technologies (IJDST)*, 3(1):48–62, 2012.