Modeling trust in multiagent mobile vehicular ad-hoc networks through enhanced knowledge exchange for effective travel decision making

by

John Finnson

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2012

© John Finnson 2012

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis explores how to effectively model trust in the environment of mobile vehicular ad-hoc networks. We consider each vehicle's travel path planning to be guided by an intelligent agent that receives traffic reports from other agents in the environment. Determining the trustworthiness of these reports is thus a critical task. We take as a starting point the multi-dimensional trust model of Minhas et al. That work had a two-phased approach: i) model trust and ii) execute an algorithm for using that trust modeling, when deciding what route to take. The framework presented in this thesis aims to clarify i) the messaging that should be supported, ii) the internal representation of the messaging and the trust information and iii) the algorithms for sending and receiving information (thus updating knowledge) in order to perform decision making during route planning. A significant contribution is therefore offered through clarification and extension of the original trust modeling approach. In addition we design a comprehensive, extensive simulation testbed that is used to validate the effectiveness and robustness of the model. This testbed supports a variety of metrics and is able to perform testing in environments with a large number of cars. This constitutes the second significant contribution of the thesis.

Overall, we present a valuable model for knowledge management in mobile vehicular ad-hoc networks through a combination of trust modeling, ontological representation of concepts and facts, and a methodology for discovering and updating user models. Included is a representation and implementation of both a push-based and pull-based messaging protocol. We also demonstrate the effectiveness of this model through validation conducted using our simulation testbed, focusing first on a subset of the multi-faceted trust model in order to highlight the value of the underlying representation, decision making algorithm and simulation metrics. One very valuable result is a demonstration of the importance of the combined use of the different dimensions employed in the trust modeling.

Acknowledgements

First and foremost I offer my sincerest gratitude to my supervisor, Dr. Robin Cohen and co-supervisor, Dr. Grant Weddell for being outstanding advisors. Their invaluable guidance, encouragement, and support from the initial research to the final stages enabled me to make this thesis successful. Furthermore, I am deeply indebted to Dr. Peter van Beek and Dr. Richard Trefler for serving on the thesis committee.

I warmly thank Amirhossein Vakili and John Champaign for their valuable advice and guidance to make my graduate studies successful. Thanks as well to Paul Kates, Jie Zhang, Umar Farooq Minhas, and Thomas Tran for providing me with valuable feedback on my research.

I am very grateful to all of my colleagues and friends: Michael Cormier, Adam Hartfiel, Rhiannon Rose, Tyrel Russell, John Doucette, Hadi Hosseini, Ayman Shalaby, Tomasz Rozdeba, Rajesh Kumar, Dan Attrell, Nea Powell, and Sarah Ruffell for being such great friends and making my life much more enjoyable throughout my stay at Waterloo.

I thank the David R. Cheriton School of Computer Science for providing the support and equipment I needed to produce and complete my thesis and NSERC for funding my studies.

Finally, my warmest appreciation goes to family, especially my father Doug Finnson, who has been supporting me throughout my time preparing and writing my thesis, and my late mother Mary Rose Finnson, who has been a source of inspiration throughout my university career.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

An intelligent agent is a software program designed to assist a user in accomplishing a task using automated reasoning, taking into consideration the user's goals and preferences. A mobile vehicular ad-hoc network is a network of vehicles that may be traveling on a road, where communication between the cars is supported. In this thesis, we assume each vehicle is representing a user and each user's travel path planning is guided by an intelligent agent.

In the work of Minhas et al.([20, 18]) a framework was proposed to support the exchange of information about traffic towards more effective decision making of intelligent agents in transportation environments, through mobile ad-hoc vehicular networks. That model focused on determining the reliability of the knowledge being exchanged, by modeling the trustworthiness of the agent providing the advice. This was achieved by using a multi-faceted approach whereby an agent's trustworthiness depended on a combination of that agent's role, the direct experience of that agent's trustworthiness, a modeling of the time and location of the agent's advice and an aggregation of collective advice from the environoment, in order to model majority consensus. In addition, the framework provided for a limiting of the number of agents to consult, so that only the most trustworthy advice overall would form the basis of the agent's decision making (and so that in scenarios which required quick decision making one could scale back on processing by focusing on a smaller selection of overall advice, whereas longterm planning could enjoy the luxury of extended information gathering).

The overall approach of the framework was to propose an algorithm for integrating the various influences on modeling trustworthiness in order to enable a decision to be made by an agent, following an exchange of information from other agents in the environment. Formulae enabled a proper discounting of advice that was possibly stale in time and location, and a proper increased value to the advice being presented by agents in authority roles, such as police.

The model was validated as effective through its use in a modest simulated traffic environment, where cars made decisions about travel routes based on the advice received and where cars that were less effective in selecting trustworthy advice would suffer from poorer travel speed overall, in the environment.

One central question that was not addressed sufficiently well was simply a proper articulation of the actual communication between the agents in the environment. Exactly what form the messages would take, what knowledge they would yield and how then that knowledge could be represented (in order to reason with it) was simply encapsulated in sample examples such as "Is there heavy traffic on Road X?" expecting a reply of "Yes" or "No".

In this thesis, we present a framework to support effective exchange of existing knowledge between agents through a clearer encapsulation of the messaging that is performed. In order to achieve this goal, we explore a subtopic in the design of effective knowledge bases known as provenance. In this context, one examines propositions being exchanged and one models the authority of the source of the proposition within an overall ontological structure for the set of sources. In so doing, one is able to not only clarify the messaging performed but also to deduce reliability of that knowledge, with an initial provenance measure. Using methods for constructing knowledge bases to support provenance, we propose a specific solution for the messaging and for the representation of the knowledge that will be managed by the agents in our mobile vehicular ad-hoc network (VANET), as the backdrop for their decision making about route planning in the environment.

Once this has been established, the trust modeling framework of Minhas et al.[20] can then be re-examined in order to provide finer distinctions on trust modeling of the sources of advice. This then leads to a framework which is able to offer: i) detailed representation of knowledge to be exchanged which integrates both ontological and propositional elements ii) motivated by knowledge provenance research, an integration of various features of propositions which assist agents in their decision making about route planning iii) representation of trust modeling measures iv) specification of an algorithm for information exchange which clarifies when, where and with whom knowledge is shared, towards effective decision making for the agents in the environment.

This framework is, in turn, validated through a careful implementation in a detailed, expanded simulation of actual traffic flow, employing large numbers of cars making decisions about travel based on knowledge received from others in the mobile environment; an effective modeling of the trustworthiness of those agents; significant metrics for determining whether effective decisions about routes were achieved within the simulation. We are able to demonstrate significant improvements in average speed of cars, average number of paths completed by cars during the simulation, and average path time of cars when using our proposed framework for message exchange, knowledge representation and trust modeling.

At the core, we provide a valuable specification of the context in which a request for information would be conveyed. This leads to a more careful characterization of how the knowledge that is gained from the communication should (and should not) influence the transportation decision of the agent and how that knowledge should be stored and accessed. As such, we also offer insights into a companion question, namely:

when should agents issue requests for information, in order to carry out effective transportation planning? This is provided through an overall proposed algorithm for information gathering from agents in these dynamically changing environments. As such, beginning to clarify more carefully the communication that is flowing between agents leads us to not only specify the format of the messaging, but also the conditions under which communication would occur, when updates to user models would ensue, and some specification of the reason and context of the communication that can become part of the agent's overall decision making process.

Overall, this research culminates in a clearer articulation of the knowledge gathering procedure recommended for agents in mobile vehicular ad-hoc networks (an important transportation application), with respect to what should be asked, to whom, when; this serves to specify not only the format of messages but also the updating frequency, to then lead to evaluations of trustworthiness that are leveraged to result in effective decisions about travel. This results in a framework that enables effective transportation decisions based on proper management of the knowledge exchanged between agents – one that has been validated as valuable through extensive simulations.

1.1 Motivating Scenarios

This section presents a few scenarios in the VANET domain to motivate the need for our proposed framework. These scenarios demonstrate issues in traffic routing and peer trust that can arise in the VANET domain. In these examples, we clarify the value of exchanging information about congestion of roads between vehicles - the primary message that we propose within the framework developed for this thesis. The first section details how vehicle congestion can arise from lack of situational awareness. The second section details how agents could easily be misled if the information being received is not treated critically.

1.1.1 Traffic Congestion

This section outlines a few scenarios where drivers in a city encounter traffic congestion. The congestion delays described could be avoided through situational awareness of congested roads.

1.1.1.1 Inexperienced Agent

An agent (A1) is passing through the city to a destination beyond and is unfamiliar with the city. Relying on a typical GPS system, A1 follows its instructions and attempts to take the GPS's outlined shortest path. The shortest path uses a main road, and results in A1 being delayed for an extended period of time. If A1 had taken any of the side routes then he would have avoided the main road's congestion and saved a lot of time.

1.1.1.2 Experienced Agent

An agent (A2) controlling a vehicle needs to reach a destination in as little time as possible. A2's destination is a parking garage for his work. A2 also has a GPS, which advises a similar route to that of A1's; however, he has been living in the city for a long time and is very familiar with how commuting works. A2 knows that the advised route will be very congested and even though it is the shortest path, he will be delayed a long time. As a result, A2 ignores the GPS and takes a side route to his destination.

During A2's detour, several things can go wrong. For example, while Agent A2 is attempting to take his side route, he could be unaware of a car crash which causes him to be late for work. Agent A2 could also be unaware that the side path has had rising popularity with the veterans of the city. Even though the side route is not as congested as the main road, taking a different route could have saved A2 some time.

1.1.2 Dishonest Agents

This section outlines a few scenarios where drivers in a city communicate with each other about traffic scenarios. The agents in the system are fully trusting of the communicated information, which results in malicious agents taking advantage of the system by sending false information. The use of dishonest information could be avoided in these scenarios through profiling (i.e. trust modeling) and other techniques, such as the modeling of agent roles.

1.1.2.1 Few Dishonest Agents

Agent A2 learned his lesson about situational awareness and now communicates with other cars about traffic scenarios. Another agent (A3) also communicates with other cars; however, he is one of a few agents who are greedy and dishonest, and take advantage of the system by informing all other cars that their current road is very congested, even though it is not. This is not entirely unreasonable to expect, in an environment where self-interested agents prefer to reduce the traffic on their current route. A2 is trusting and always believes information from other agents. During a commute, A2 is informed by A3 that his side road is very congested due to a traffic accident. Using this information, A2 takes the main road which he believes is less congested and is unnecessarily delayed. Even though there are only a few malicious cars, they can have a substantial impact on the overall traffic flow of the city. After A3 broadcasts his dishonest reports, he manages to get to his destination with little delay.

1.1.2.2 Many Dishonest Agents

The congestion communication system has been in place for a while now and the system is now filled with dishonest agents who are trying to emulate the success of A3's dishonesty. The high number of dishonest agents results in the city having congestion issues comparable to what existed before the congestion communication system was used. This is due to most of the roads being reported with similarly bad congestion values, resulting in agents not using traffic information and defaulting to using the route with the shortest distance.

1.1.3 Conclusion

Examining these scenarios, the conclusion is that exchange of information about traffic congestion on roads would be valuable, but that a careful modeling of the trustworthiness of the agents providing these reports is warranted. The remainder of the thesis develops a framework to enable this communication and modeling, together with a comprehensive validation of its effectiveness.

Chapter 2

Background

This chapter describes some terminology and foundational related work.

2.1 Agents and Multiagent Systems

In artificial intelligence, an intelligent agent is defined as an autonomous entity that observes and acts upon an environment and directs its activity in order to achieve its goals [25].

Multiagent systems are defined in [31] as: "... systems in which several interacting, intelligent agents pursue some set of goals or perform some set of tasks.” Multiagent systems (MAS) are used to solve problems that are difficult or impossible for an individual agent to solve.

2.2 Ontologies

In computer science, ontologies formally represent knowledge as a set of concepts within a domain, and the relationships between those concepts. Ontologies are used by intelligent agents to reason about entities within a domain and serve the purpose of describing the domain. An ontology is defined in [12] as a "...formal, explicit specification of a shared conceptualization." Ontologies can be formally understood in the context of a dialect of description logic (DL) [2]. This applies for instance to OWL2 in relation to the semantic web [21].

Ontologies are typically used as structural frameworks for organizing information, in a hierarchical fashion, showing classes and subclasses relationships, instances and properties. The descriptive nature of ontologies has promoted the creation of semantic databases for knowledge representation in areas such as biomedical informatics, systems engineering, the semantic web, and artificial intelligence [27, 24, 28, 13]. DL-based ontologies are especially suited for "open worlds" where the collection of facts may be incomplete.

2.3 Provenance

The term provenance comes from the French word *provenir*. It is defined in [1] as referring to the chronology of the ownership or location of an object. In computer science, provenance is used to determine the source of information, such as traffic data, in order to assess whether it is trustworthy or not.

Databases in computer science often are concerned with provenance to a certain degree and usually contain provenance information, either for tracking ownership or data usage. Ownership usually concerns the source, or creation information, of data while data usage concerns the modification and access of data [6].

2.4 Vehicle Ad-Hoc Network (VANET)

A vehicular ad-hoc network (VANET) is a subset of mobile ad-hoc networks (MANETs), and represents technology that concerns networks of inter-connected cars, typically represented as nodes or agents. Self-configuring infrastructureless ad-hoc networks from these cars arise through some type of wireless communication and the ability for cars to dynamically join or leave the network. These networks are typically used to communicate and/or coordinate for a certain task or purpose.

Real world examples of VANET systems are some taxi companies, which communicate positions and customer requests, and buses, which communicate positions and other important information for staying on schedule. One of the most common types of information communicated within these networks is traffic data, such as road closures or car crashes, so that the cars within these networks won't be delayed by these unforseen circumstances.

2.5 A Multi-faceted Trust Management Framework

The examples of VANET systems of the previous section, involving taxis and buses, described vehicles of relatively small groups which were strictly managed. This section describes the work by Minhas et al.[20, 18], which introduces the idea of larger groups of unmanaged vehicles that communicate traffic information. The research presented in Minhas et al.[20, 18] does not limit the cars that may join or leave the VANET. This allows for communication and coordination for a larger scale of vehicles, including the possibility of all vehicles within a city being connected. The goal of this thesis is to expand upon the research presented in this section and to validate it within an implemented simulation framework.

An immediate issue arises from this type of open network, which is addressed in the work of Minhas et al.[20, 18]. When communicating traffic information, the vehicle you are communicating with may not be trustworthy and may be sending false information for devious or greedy purposes. An example of this was described in the motivating scenario of Section 1.1.2.1. Minhas et al.[20, 18] introduce a multi-faceted trust management framework for dealing with these dishonest agents, providing each agent with the most accurate and trustworthy data.

The driver of each vehicle in our VANET environment is considered to be a user. In order for each vehicle on the road to make effective traffic decisions, information is sought from other vehicles (about the traffic congestion on a particular road). The traffic information communicated might be from a less than reputable source. As a result it becomes important to model the trustworthiness of the other agents that provide information. The multi-faceted trust management framework shown in Figure 2.1 [20, 18] includes role-based trust and experience-based trust in an integrated evaluation metric. This is used for determining trustworthiness of vehicular agents that can in turn be used to determine the reliability of the information shared by these agents.

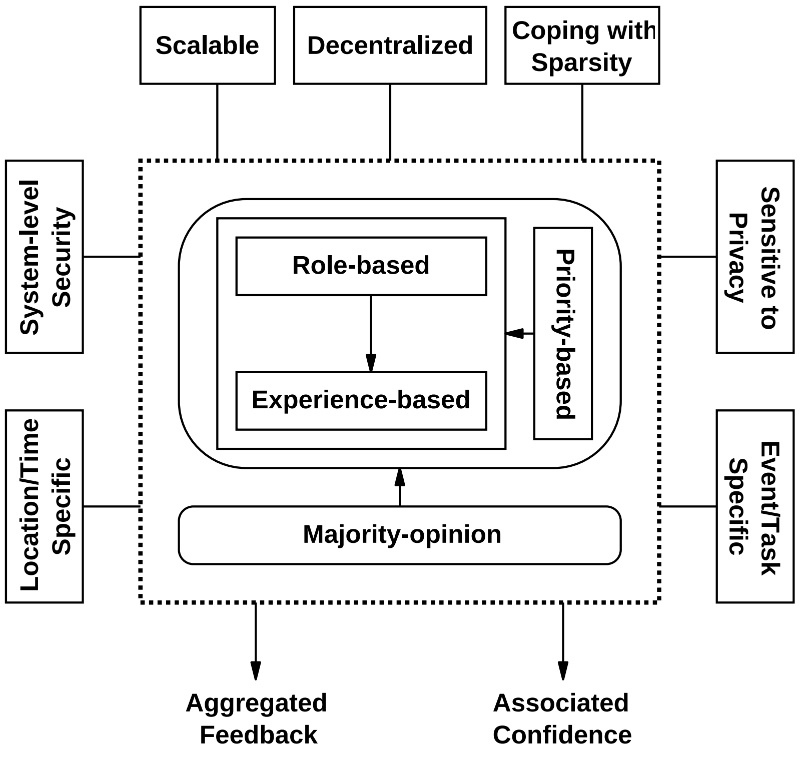

Figure 2.1: A Multi-faceted Trust Management Framework, from Minhas et al.[20]

Role-based trust acknowledges that certain vehicles in the environment may play a particular role and, on this basis, merit greater estimates of trustworthiness. For example, there may be vehicles representing the police, taxis, buses, and other traffic authorities or ones representing radio stations dedicated to determining accurate traffic reports by maintaining vehicles in the vicinity of the central routes. Or there may be a collection of users representing a “commuter pool”, routinely travelling the same route, sharing advice. The proposal for considering roles when reasoning about travel is to group users together according to their designated role, but then to still be able to order each collection of role-based users from the most to the least trustworthy, on the basis of past experience with the users. Role-based trust is also particularly useful for coping with the data sparsity problem where agents may not have much experience with each other in large and open VANET environments.

Experience-based trust allows for an agent to continuously model and update its trust in another based on direct experiences between them. The incorporation of the experience-based trust is to make use of any evidence from direct interactions, whenever available, into the trust calculation. Consideration of any past personal experiences with users allows the model to include any learning about particular users due to previous encounters, specifically modeling trustworthiness each time and adjusting the level of trust to be higher or lower, based on the outcome of the advice that is offered.

Experience-based trustworthiness is represented and maintained according to equations 2.1 and 2.2.

$$T_A(B) \leftarrow \begin{cases} T_A(B) + \alpha(1 - T_A(B)) & \text{if } T_A(B) \geq 0, \\ T_A(B) + \alpha(1 + T_A(B)) & \text{if } T_A(B) < 0, \end{cases} \tag{2.1}$$

$$T_A(B) \leftarrow \begin{cases} T_A(B) + \beta(1 - T_A(B)) & \text{if } T_A(B) \geq 0, \\ T_A(B) + \beta(1 + T_A(B)) & \text{if } T_A(B) < 0, \end{cases} \tag{2.2}$$

where $T_A(B) \in (-1, 1)$ is the trust value indicating the extent to which agent A trusts (or distrusts) agent B according to A's personal experience in interacting with B, $0 < \alpha < 1$ is a positive increment factor, and $-1 < \beta < 0$ is a negative decrement factor. After A follows an advice of B, if the advice is evaluated as reliable, then the trust value $T_A(B)$ is increased by Equation 2.1. Otherwise, if B's advice is evaluated as unreliable, then $T_A(B)$ is decreased by Equation 2.2. These enable agents to update their trustworthiness value for other agents, based on past experiences.

Values of α and β can be set to be event-specific. For example, when asking about a major accident, these values may be set high, to reflect considerable disappointment with inaccurate advice. The framework also incorporates a requirement for users to reveal whether the traffic information they are providing has been directly observed or only indirectly inferred from other reports that user has received. The critical distinction of direct or indirect reporting then influences the values set for α and β, introducing greater penalties for disappointment with direct advice. Minhas et al.[19] discusses at greater length incentives to honesty that are introduced within this framework; for brevity, we omit that discussion in this thesis.

Note that Minhas et al. introduces priority-based trust modeling as well, setting a restricted number of users n to consult for each travel decision. This is clarified in the following subsection.

2.5.1 Majority Consensus

A central calculation to influence the travel decision of each user is the determination of majority consensus amongst the users providing advice about a particular road. We outline below the calculations performed by a user for this reasoning within the framework of Minhas et al.

Step 1: The user maintains, as part of her model of other users, an ordered list of users to ask for advice:

$$\begin{bmatrix} G_1: & u_{11}, & u_{12}, & u_{13}, & ..., & u_{1k} \\ G_2: & u_{21}, & u_{22}, & u_{23}, & ..., & u_{2k} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ G_j: & u_{j1}, & u_{j2}, & u_{j3}, & ..., & u_{jk} \end{bmatrix}$$

This list is ordered from higher roles to lower roles (e.g. G_1 is the highest role), with each group of users of similar roles being ordered from higher experience-based trust ratings to lower ratings. Thus, u_{ij} represents the user in role group i that is at the j^{th} level of experience-based trust[1]. Hence, this ordered list combines role-based trust and experience-based trust into a priority-based approach.

Step 2: The user sets a value n = number of users whose advice will be considered[2].

Step 3: The user asks the first n users from her ordered list the question, receives their responses (reports), and then performs majority-based trust measurement:

Step 3A: Suppose that q of these n users declare that their reports are from direct experience. The asking user determines whether there are sufficient direct witnesses such that she can make a decision based solely on their reports, by comparing q with a threshold N_{min}[3].

Step 3B: If $q \geq N_{min}$, then the asking user will only consider the reports from the q direct witnesses. If a majority consensus on a response can be reached, up to some tolerance set by the asker (e.g. the user may want at most 30% of the responders to disagree), then the response is taken as the advice and followed.

Step 3C: If $q < N_{min}$, then there are insufficient direct witnesses. In this case, the asking user will consider reports from both direct and indirect witnesses, assigning different weight factors to them, computing and following the majority opinion.

Step 3D: Once the actual road conditions are verified, the asking user adjusts the experience-based trust ratings of the reporting users: It penalizes (rewards) more those users who reported incorrect (correct) information in the direct experience case than those users with incorrect (correct) information in the indirect experience case.

[1] There is no need for each group to have the same number of elements. We provide here only a simplified example.

[2] This integrates task-based trust. For instance, a user may set n to be fairly small, say $n \leq 10$, if she needs to make quick driving decision, or set a larger n if she has time to process responses.

[3] See equation (7) in [19] for how to compute this threshold.

Step 3E: If a majority consensus cannot be reached, then requiring majority consensus for advice is abandoned. Instead, the user relies on role-based trust and experience-based trust (e.g., taking the advice from the user with highest role and highest experience trust value).

Step 4: In order to eventually admit new users into consideration, the user will also ask a certain number of users beyond user u_n in the list. The responses here will not be considered for decision, but will be verified to update experience-based trust ratings and some of these users may make it into the top n users, in this way.

The computation of majority consensus adheres to the set of formulae outlined below:

Suppose user A receives a set of m reports $\mathcal{R} = \{R_1, R_2, ..., R_m\}$ from a set of n other users $\mathcal{B} = \{B_1, B_2, ..., B_n\}$ regarding an event. User A will consider more heavily the reports sent by users who have higher level roles and larger experience-based trust values. When performing majority-based process, also taken into account is the location closeness between the reporting user and the reported event, and the closeness between the time when the event has taken place and that of receiving the report. Below C_t represents time closeness, C_l location closeness, T_e experience-based trust and T_r role-based trust. Note that all these parameters belong to the interval $(0, 1)$ except that T_e needs to be scaled to fit within this interval.

For each user B_i $(1 \leq i \leq n)$ belonging to a subset of users $\mathcal{B}(R_j) \subseteq \mathcal{B}$ who report the same report $R_j \in \mathcal{R}$ $(1 \leq j \leq m)$, the effect of its report is aggregated according to the above factors. The aggregated effect $E(R_j)$ from reports sent by users in $\mathcal{B}(R_j)$ can be formulated as follows:

$$E(R_j) = \sum_{B_i \in \mathcal{B}(R_j)} \frac{T_e(B_i)T_r(B_i)}{C_t(R_j)C_l(B_i)W(B_i)} \tag{2.3}$$

In this equation, experience-based trust and role-based trust are discounted based on the two factors of time closeness and location closeness. Note that location closeness $C_l(B_i)$ depends only on the location of user B_i while time closeness $C_t(R_j)$ depends on the time of receiving the report R_j. $C_t(R_j)$ can also be written as $C_t(B_i)$ because it is assumed that each report is sent by an unique user at a possibly different time. $W(B_i)$ is a weight factor set to 1 if user B_i who sent report R_j is an indirect witness, and $W(B_i)$ is set to a value in $(0, 1)$ if user B_i is a direct witness[4].

To consider the effect of all the different reports, the majority opinion is then

$$M(R_j) = \max_{R_j \in \mathcal{R}} E(R_j) \tag{2.4}$$

which implies the report that has the maximum effect, among all reports.

[4]For example, setting $W(B_i) = 1/2$ for the case of direct witnesses indicates that the asking user values direct evidence twice more than indirect evidence.

A majority consensus can be reached if

$$\frac{M(R_j)}{\sum_{R_j \in \mathcal{R}} E(R_j)} \geq 1 - \varepsilon \tag{2.5}$$

where $\varepsilon \in (0, 1)$ is set by user A to represent the maximum error rate that A can accept. A majority consensus can be reached if the percentage of the majority opinion (the maximum effect among different reports) over all possible opinions is above the threshold set by user A.

Finally, a core processing algorithm was proposed for use by each user that seeks advice from other vehicles in the environment, which is summarized in Algorithm 1.

Algorithm 1: Computation Steps

```
while on the road do
    if in need of advice then
        Choose $n$; //number of users to ask for advice
        Prioritize $n$ users;//according to roles and experiences
        Send requests and receive responses;
        if response consensus > acceptable ratio then
            Follow advice in response;
        end
        else
            Follow advice of user with highest role and highest trust value;
        end
    end
    Verify reliability of advice;
    Update users' trust values;
end
```

The trust modeling framework described above clarifies the algorithms that lead to the calculation of the trustworthiness value which would then be stored in each user model. Trip planning decisions of a vehicle would then be made in light of these particular user models.

Elements that require further clarification are the detailed representation of a user model (where trustworthiness information is stored, relative to other information about the user), the world model (where user models and information not specific to users, such as road location and congestion data are stored), a specific proposal for communication protocols (when updates to the user model should occur and what they contain) and the design of a high level agent infrastructure (design of how world and user model, communication protocols, and reasoning components combine with each other). The last element is especially important for preparing this research to be applicable for real world implementation. This thesis presents solutions to these elements, which are elaborated in the chapters that follow.

2.6 Knowledge Provenance

Provenance means the origin, the source of something, or the history of the ownership or location of an object [1]. The concept of provenance is central to this thesis and essential to the Semantic Web. Huang explains that provenance “tells not only what is the information source but also how this information is derived, what is the context of this information, and how this information is used” [14]. All of this information is extremely important for determining the value and integrity of an information resource [3]. This is defined as provenance based judgment. In order to effectively use data from sources, provenance reliability needs to be assessed. This is especially important for the VANET research where other cars might be unreliable sources of information. Certain traffic reporting sites may also be more reliable than others, which can affect which value will be officially used. From this observation, provenance assessment is important for all sources and is therefore used in this thesis on all resources.

As explained, the purpose of knowledge provenance (KP) is to address the problem of how to determine the validity and origin of information/knowledge on the web [9]. The goal of Huang’s thesis [14], was to fully describe, represent, and model provenance in the context of the web.

Provenance information is primarily described by Huang through propositions. A proposition is an atomic piece of information that is annotated with provenance information. Huang represents propositions through an ontological class structure shown in Figure 2.2. Instances of each class are used to represent individual propositions.

The primary concern of Huang’s knowledge provenance research was determining the validity of information through investigating its source. This was done through a process of finding and establishing the source, which defines the provenance, and then assessing the reliability of the provenance. When finding the fundamental source of data, this can possibly involve routing through multiple sites. This is very common on the internet where sites often cite information from other sites, that also do the same.

2.6.1 Distributed Trust Reasoning

This section describes the concept of distributed trust reasoning (DTR) and how it functions in relation to knowledge provenance. DTR attempts to solve the problem of modeling trust in a social network where personal trust judgments are not accessible in a search. The work by Huang proposes a DTR model constructed by using situation calculus [14]. This section describes the terminology and actions for distributed trust reasoning and how they are used.

DTR works with KP as a functional layer on top of it. DTR is used when KP is implemented in a distributed world, with each agent assumed to have its own

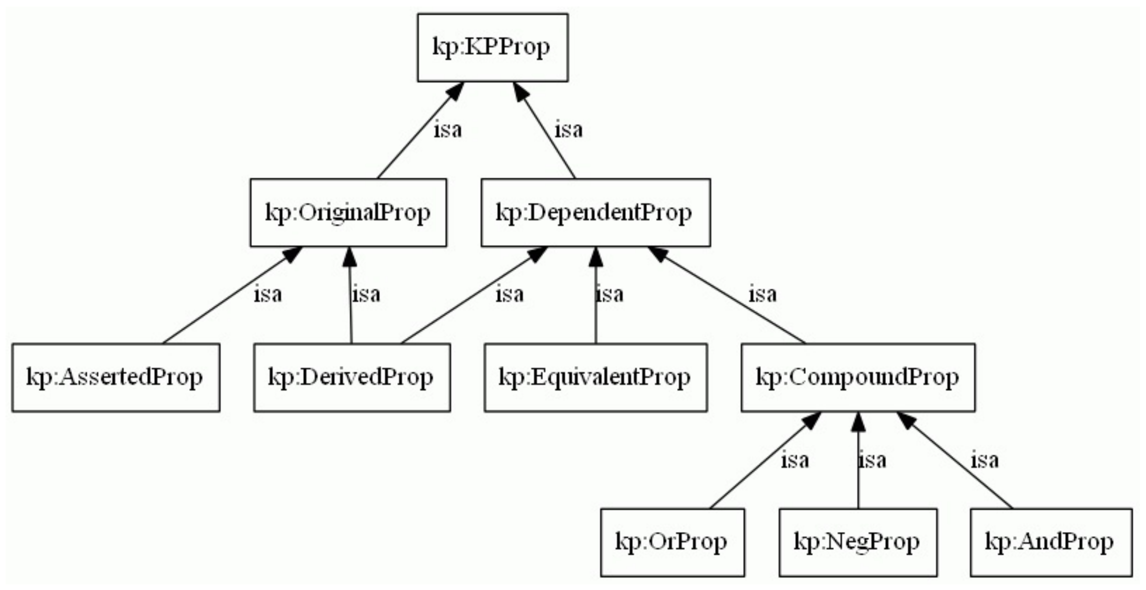

Figure 2.2: Knowledge Provenance Ontology.

Table 2.1: Actions. Reprinted from [14]

Action	Definition/Semantics
request(e,query(e,e’,q))	$\subseteq \mathbf{E} \times \mathbf{D}$ Agent e requests agent e’ whether fluent q holds.
checkAnswer(e,query(e,e’,q),w)	$\subseteq \mathbf{E} \times \mathbf{D} \times \mathbf{D}$ Agent e gets answer w for the query.
acceptQ(e’,query(e,e’,q))	$\subseteq \mathbf{E} \times \mathbf{D}$ Agent e’, accepts query from agent e.
replyQ(e’,query(e,e’,q),w)	$\subseteq \mathbf{E} \times \mathbf{D} \times \{\text{Yes, No}\}$ Agent e’ replies to the query from agent e with w.

knowledge base.

Situation calculus is primarily composed of actions, which can be performed in the world and fluents, which describe the state of the world. Table (2.1) and (2.2), modified from [14], summarize the new actions and fluents needed for the KP world.

These actions and fluents are used to question friends as to whether they trust each other. The friend could subsequently ask their friends if they do not have an answer. Through this process, a transitivity of trust occurs which allows friends and the original requestee to determine the validity of the source and its data [14]. This process is demonstrated in figure (2.3), conveniently cast into the VANET domain.

This thesis focuses on VANET, which uses agents rather than web sites to define the source of data. The proposed framework replaces the KP reasoner in [14] with a new one that is specific to the VANET domain, and will be described in this thesis.

Table 2.2: Fluents. Reprinted from [14]

has_query(query(e,e',q))	$\subseteq \mathbf{D}$ Relational fluent. Query from e to e' has been made.
has_answer(query(e,e',q),w)	$\subseteq \mathbf{D} \times \mathbf{D} \times \mathbf{D}$ Relational fluent. Query from e to e' has answer w.
has_task(e,e',q)	$\subseteq \mathbf{E} \times \mathbf{E} \times \mathbf{F}$ Relational fluent. Agent e has task to answer question q from agent e'.
query(e,e',q)	$\subseteq \mathbf{E} \times \mathbf{E} \times \mathbf{F} \rightarrow \mathbf{F}$ Functional fluent. Represents a query from agent e to e' about fluent q.

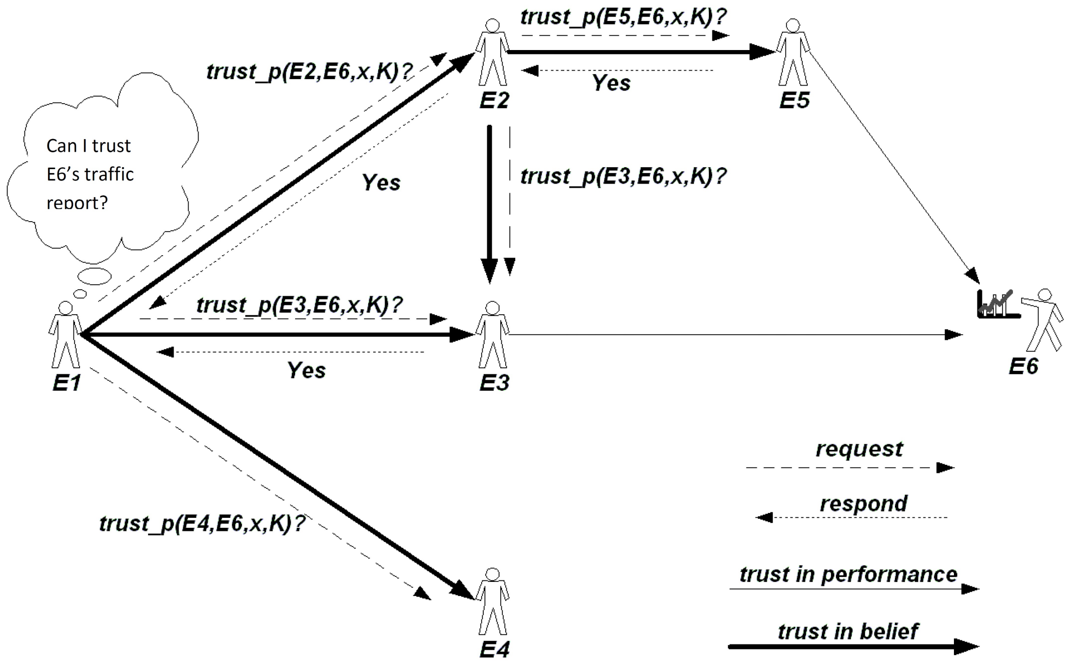

Figure 2.3: Example of distributed trust reasoning in social networks.

Chapter 3

Framework

This chapter and the following chapters present the original work of the thesis, which seeks to expand upon the foundational research presented in Section 2.5 and implement it within a simulation framework.

This chapter introduces the expanded multi-faceted trust management framework by describing the high level infrastructure of an individual agent (within VANET), as well as components which seek to represent and clarify the elements which lacked clarification in Section 2.5.

This thesis adopts a multiagent systems approach where each vehicle in the environment is equipped with an intelligent agent that determines the path for the vehicle to follow, based on reports received from other vehicles in the environment. Due to this structure, messaging between vehicles becomes a central element of the overall framework. Our solution for messaging is motivated by the proposals of [14] for distributed agent communication.

An important result in this chapter is the creation of a framework for knowledge representation to function as a semantic repository for the VANET terms and to serve as the knowledge base used by the VANET agents as part of their reasoning about other agent and paths to follow in the transportation environment. The knowledge base is expressed as an ontology, V-Ont + KP, motivated by the representation used in [14]. This knowledge base fulfills the element roles of a world and user model, which were lacking clarification in Section 2.5.

3.1 Structure Overview

Our overall framework consists of an underlying knowledge representation, proposed messaging, and proposed trust modeling (The Model) together with a simulation testbed (The Simulation). The testbed is used to validate the model as valuable for enabling effective transportation decision making (i.e. route selection) by intelligent agents, sensitive to information about the environment data received from other intelligent agents. Figure 3.1 presents a UML component diagram of our proposed framework. This diagram illustrates all of the major components of the framework and how they fit into the overall model for a single agent. Certain aspects of the diagram are specifically descriptive of the simulation implementation (presented in Chapter 5); however, the overall structure is descriptive of the general model framework.

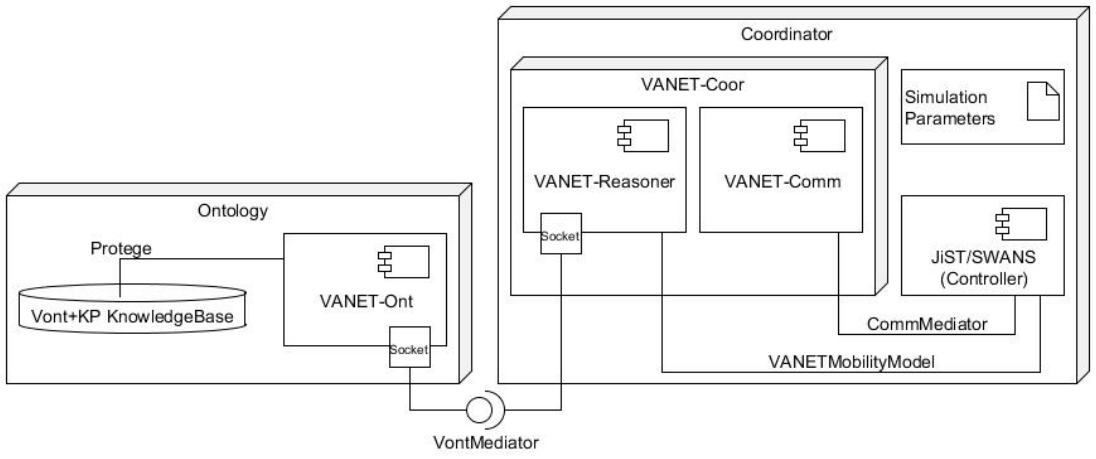

Figure 3.1: UML Component Diagram of the VANET Model.

In analysis of the diagram, the design is separated into two areas, Coordinator and Ontology. This depicts a clear separation between how knowledge is stored and how it is used. The Simulation Parameters artifact is a file of parameters that would be used by the components in the Coordinator node to set up a simulation run of the implementation (discussed further in Chapter 5). The following subsections will describe the Coordinator and Ontology elements and their relative components. Due to Figure 3.1 being indicative of a single agent's framework, the following sections will be described in reference to the resident agent. The resident agent references the native agent which possess the components being discussed. This is an accurate description due to the system being designed for an Ad-Hoc network with no other components existing outside the described framework.

3.2 Coordinator Element

The Coordinator node is separated into two areas, VANET components (VANET-Coor) and a Controller, labelled JiST/SWANS[1], used to issue vehicle commands such as turning at an intersection, and to send messages. VANET-Coor is made up of two components, VANET-Comm and VANET-Reasoner. VANET-Comm is responsible

[1]This specific label is representative of an architecture that serves as the controller. This enables us to connect well to the specific controller used in our Simulation testbed, described in Chapter 5.

for all communication. This includes when to send messages, what messages contain, and how to process them. VANET-Reasoner is responsible for the majority of reasoning and processing of controller information and stored semantic knowledge.

3.2.1 Controller Component

The Controller component is treated as a self contained modular component that acts as a gateway for mediating all low level interactions with vehicles, such as physically sending a message to another car, controlling the speed at which the resident car is traveling, or advising an optimal route for the vehicle.

Due to an open-ended treatment of this component its role can be filled in various ways, such as implementing a 3^{rd} party simulation software or a GPS device that is connected to a real car. The validation of this thesis in Chapter 5 implements the JiST/SWANS simulation software to fill the role of a controller component. The possibility of connecting a GPS device leads to the possibility of real world deployment which is further discussed in the future work section of Chapter 6.

The majority of roles tied to the controller component are autonomous and never need interaction, such as controlling the speed of a car. However, some responsibilities are fundamentally tied to the VANET framework, such sending messages and deciding what path to take. These actions interact with the VANET components through the VANETMobilityModel. The VANETMobilityModel serves as an extension to the controller component and is responsible for communication between the VANET components and the controller.

3.2.2 Communication Component

The VANET-Comm communication component is responsible for sending and managing all messages to and from other agents. This responsibility is divided into two different areas, communication protocols, such as pull and push, and message management, which was inspired from the messaging proposals of [14]. The communication protocols are referred to in this thesis as the pull and/or push protocol. Message management is referred to in this thesis as KP-Comm. The pull and push protocols control *why* and *when* to send messages. KP-Comm is responsible for *how* the messages are structured, according to Section 2.6.1. The Communication component also interacts with the Reasoner component which determines *what* the messages contain. *Who* the messages are sent to is determined either randomly (Push) or from interacting with the Reasoner component (Pull), which then interacts with the Ontology node to retrieve a trustworthy agent's information. The pull and push protocols and KP-Comm are more thoroughly described in Section 4.2.1.

3.2.3 VANET-Reasoner Component

The Reasoner component is responsible for processing and determining the important operations of the other components of the Coordinator and Ontology nodes. These responsibilities can be grouped into two areas, processing messages and acting as a mediator to retrieve traffic information for the Controller component. Processing messages is a complicated task which can involve issuing responses to information requests and processing message information to inform the Ontology node of newly discovered information, such as roads, agents, or congestion data [2].

Interaction with the Controller component is mediated through VANETMobilityModel. Processing of semantic knowledge is done through interactions with the VANET-Ont component through the VontMediator mediator. The diagram's indicated socket connection is specific to the simulation's implementation and is described in Chapter 5.

3.3 Ontology Element

The Ontology node contains two elements, a VANET-Ont component and a Vont + KP knowledge base. The VANET-Ont component is responsible for mediating communication with the VANET-Reasoner of the Coordinator node, performing complex knowledge base interactions, and contains a reasoner for modifying the trust of agents and calculating a majority opinion of congestion data. The Vont + KP knowledge base contains all of the resident agent's knowledge of the world, including congestion data, and profiles of other agents.

3.3.1 Vont and KP Knowledge Base

The Vont and KP Knowledge base is responsible for hosting all semantic data for the resident agent. Each agent will manage its own Knowledge Base (its view of the world). As such, each agent's knowledge base includes models of other agents in the environment. The VANET-Ont component interacts with the knowledge base to complete various tasks such as dynamically creating and adding new instances, updating instances, retrieving semantic data, and querying for collections of related semantic data. This knowledge base is described as *Vont and KP* because it is designed as a modular combination of two ontologies, a VANET ontology (Vont) and a Knowledge Provenance (KP) ontology which are further described in Section 4.1. The KP ontology is a general purpose ontology with classes that represent propositions. The Vont ontology is specific to the VANET domain and is used by the KP ontology to describe propositions. This setup, similar to the Controller component, allows for the Vont ontology to be possibly replaced with another ontology if this framework were to be applied to a different domain.

[2]Depending on whether the agent is honest or dishonest, responses will ether contain truthful or misleading information.

3.3.2 Mediator Component

The Mediator component is contained within the VANET-Ont component and is responsible for all interactions with the knowledge base and the Coordinator node. The knowledge base interactions include knowledge base instance additions, updates, retrievals, and queries. The interactions with the Coordinator node, specifically the VANET-Reasoner component, mostly involve the VANET-Reasoner sending new information to be added to the knowledge base or requesting important information. The requests for information typically involve calculating the majority congestion of a road or information of a trustworthy agent. This requested information is then used by the Controller component and the VANET-Comm component, respectively.

3.3.3 VANET-Ont Reasoner

The Reasoner component is contained within the VANET-Ont component and is responsible for reasoning about and executing the more complex knowledge base interactions. These interactions include the processing of agents for trust updates, and processing new congestion data to form majority opinions, which are then stored in the knowledge base and sent to the Coordinator node. This component interacts with the Mediator component to perform knowledge base interactions and to send messages to the Coordinator node.

3.4 Agent Modeling

A fundamental aspect of the proposed VANET framework is its representation, gathering, management, and use of trust information. This information plays a role in each of the components detailed in Figure 3.1. The VANET-Comm component is responsible for gathering trust information. VANET-Ont is responsible for representing trust information and most of its management, which includes the trust modeling algorithms. VANET-Reasoner is responsible for issuing requests to VANET-Ont in order to perform trust-based decision making for path planning. The Jist/Swans Controller knows the paths that can be taken by the vehicle, in order to direct the car's next move.

The following chapter details the proposed Model, with references to the components of Figure 3.1 and its relevant generalizations, specifically how VANET was redesigned, the messaging design, and the knowledge base. The trust model is also described in greater detail with its implications into the different areas of the proposed model. Chapter 5 details the implementation of the proposed framework, with references to the components of Figure 3.1 and the aspects which are specific to the implementation and their design rationale. That chapter will also detail relevant metrics of the simulations and display their results.

Chapter 4

The Model

This chapter outlines the proposed model of the framework, which defines a generalized system of collecting, storing, and using trust-dependent information. The model can be described as being motivated by the Knowledge Provenance research from [14] and the VANET research from Minhas et al.[20, 18] with extensions and updates that will be detailed in this chapter. This proposed model can be repurposed for other trust-based projects; however, for the purposes of this thesis the VANET project of Minhas et al. (henceforth simply referred to as VANET), its unique problems, concepts, and scenarios from Section 1.1 are used in defining parts of the model. This chapter is separated into three sections: the first defines the use of ontologies to store semantic information, the second defines how this information is collected through distributed agent messaging, and the last defines how the information and types of trust are used to make decisions.

4.1 Knowledge Base

The VANET model makes use of knowledge bases to store relevant information. Each individual agent maintains its own knowledge base that respects the same ontology (as displayed in Figures 2.2 and 4.1) but is populated with unique instances and values. The ontology is designed as the combination of two separate ontologies, a modified version of the Knowledge Provenance (KP) ontology from [14] shown in Figure 2.2, and a new ontology called Vont, which contains all previous and new semantic information from the VANET project and is shown in Figure 4.1.

Knowledge bases were chosen as the primary means of information storage for reasons of simplicity and competency in the intelligent agent domain and their inherent advantage for representing open world information. The Knowledge Provenance work by [14] initially provides an OWL/RDF [17] implementation in that thesis' appendix. Included is the use of classes, individuals, variables, and relationships to describe its data. Classes define types of objects, individuals are unique instances of class objects, and variables are used to describe either basic or complicated relationship information about instances. The chosen knowledge base design is thus reflective of description logics [2], which integrate both a terminological and an assertional component.

4.1.1 Knowledge Base Design

Knowledge bases are particularly useful in agent based systems and our framework. This is because they allow us to formulate an ontology that can be robust and generically descriptive while allowing a population of instances that make each knowledge base unique to the agent it belongs to. An important additional concept that is introduced to the ontologies used by this model is the use of classes that represent propositions, as in the Knowledge Provenance work.

The dual ontology nature of the model allows for the propositions of the KP ontology to use the descriptive language of the other ontology, defined in this thesis as Vont, to describe its contents. The KP and Vont ontologies are combined in a separated design in which no fundamental concepts overlap and create unnecessary complexities or conflicts. Vont instances never reference KP instances. However, KP proposition instances reference Vont instances. An example is shown in Figure 4.5 of Section 4.1.3. References do not directly impact Vont instances, but simply entail that additional related information is provided through KP proposition instances.

The necessity of using Vont to describe the content of KP proposition instances stems from the need for a computer to be capable of understanding the content of a KP proposition for analytical purposes, such as a reasoner analyzing the content of a traffic report proposition to understand the road it's concerning and the reported congestion. The formal hierarchy of these two ontologies can be described as Vont forming a basic descriptive level, which is used by the KP ontology to describe its content, but still remaining fundamentally disjoint. The model's role is to create and populate the combined knowledge base with instances for each ontology, to then have its data used in other components of the model.

4.1.2 Vont Ontology

The VANET ontology (Vont) was created to semantically describe most concepts and relationships from the VANET research described in Section 2.5. The ontology contains many definitive and abstract classes and relationships so that it can be used to describe a wide range of VANET scenarios with minimal confusion. The design of the Vont ontology also was intended to be easily modified so that it can be utilized for future research. Figures 4.1, 4.2, and 4.3 show the Vont Ontology created from the VANET project [20, 18].

Figure 4.1 shows a basic representation of the Vont ontology. The Vont ontology forms a collection of classes and subclasses with inter and intra instance relationships. Classes represent an initial concept at the top, like *v-ont:Location*, and then subclasses of the concept as children, like *v-ont:Road*. Classes of Vont are depicted in Figure 4.1

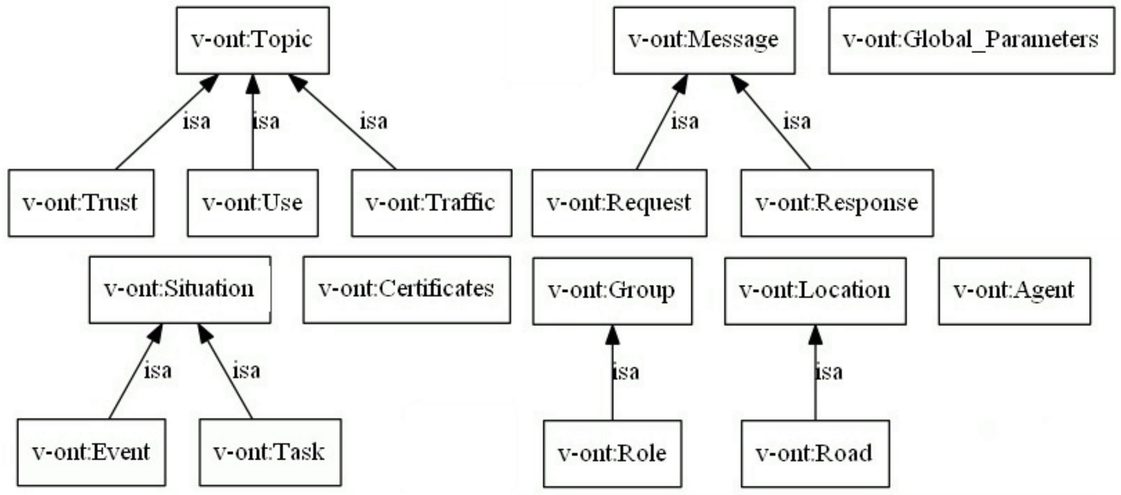

Figure 4.1: VANET Vont Ontology.

with arrows from children pointing to their parent with an *isa* label indicating an *is a* relationship.

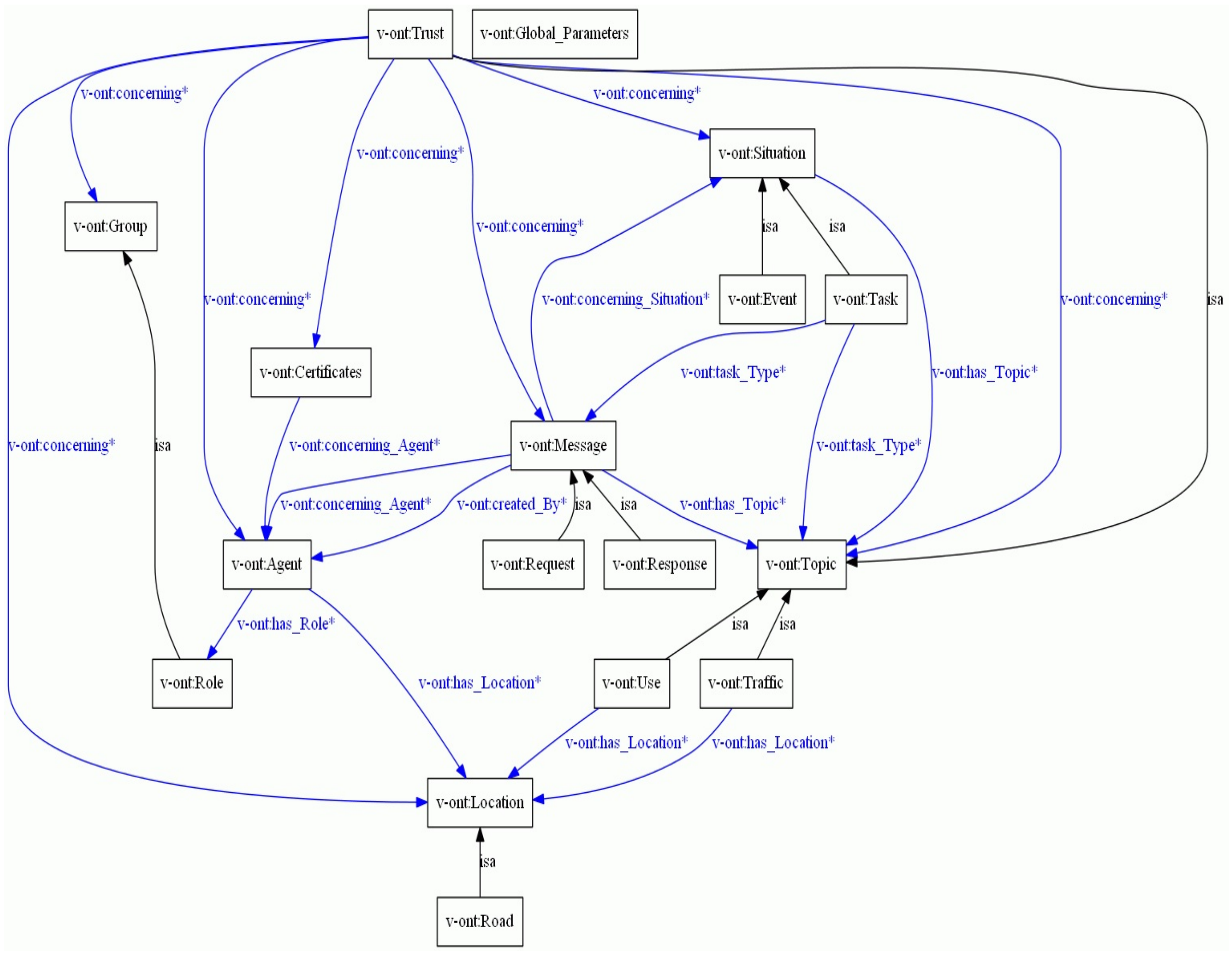

Figure 4.2: VANET Vont Ontology with all Object Properties.

Figure 4.2 shows an expanded diagram of 4.1, which includes all Object Properties. Object Properties are class properties which reference an instance of another class. For example, *v-ont:Agent* has the Object Property *v-ont:has_Location* which points to the class *v-ont:Location*. This Object Property indicates that an instance of *v-ont:Agent* can have the property *v-ont:has_Location*, which references an instance of *v-ont:Location* and thus indicates the last known location of the agent instance.

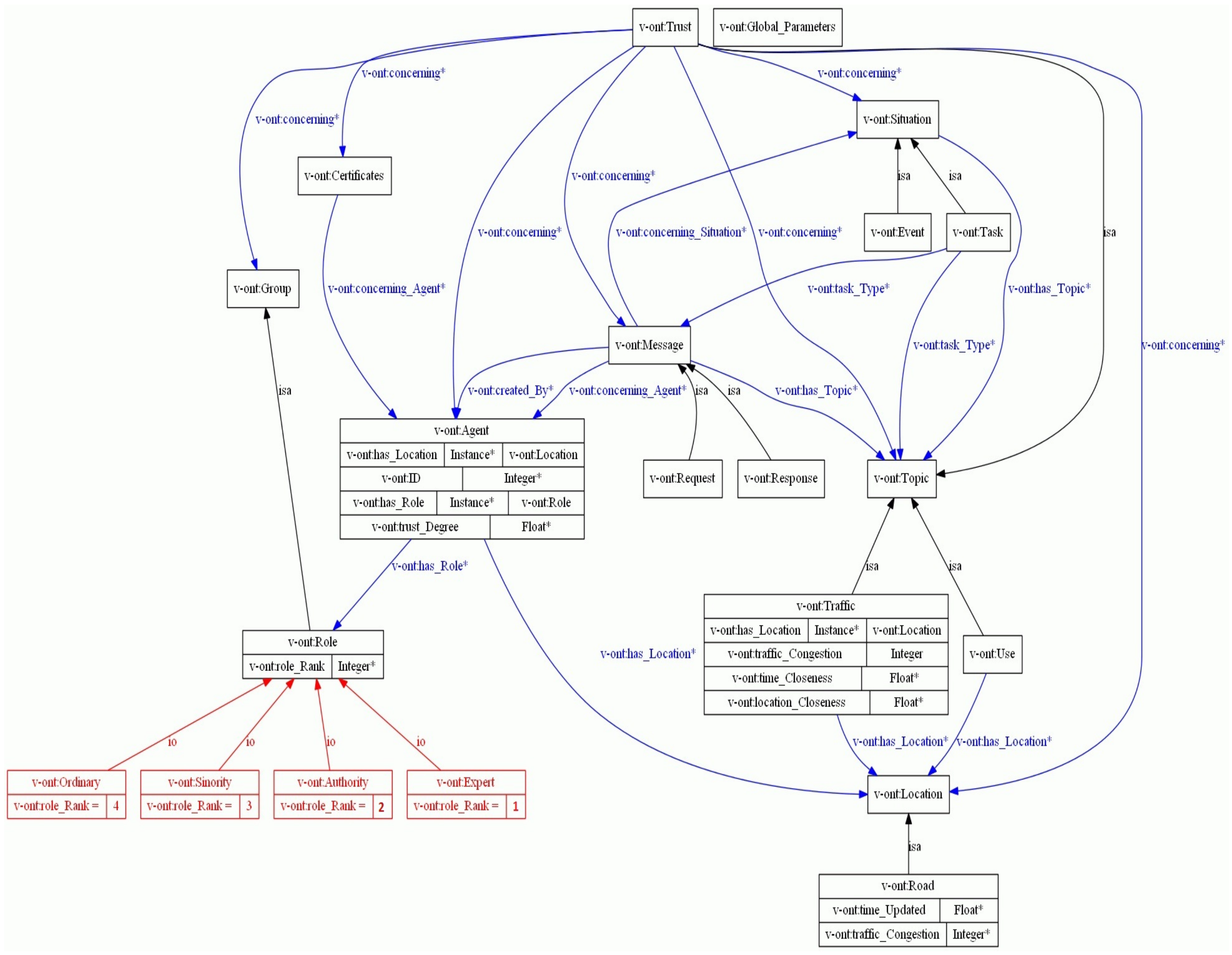

Figure 4.3: VANET Vont Ontology with all Object Properties, initial instances, and important Datatype Properties.

Figure 4.3 shows an expanded diagram of 4.2, which includes some important Datatype Properties. Datatype Properties are class properties which store a basic datatype value. For example, *v-ont:Agent* has the Datatype Property *v-ont:trust_Degree*. This Datatype Property indicates that an instance of *v-ont:Agent* can have the property *v-ont:trust_Degree*, which contains a number that indicates the resident agent's current trust of the *v-ont:Agent* instance. The Datatype Properties of the figure are further described in the sections which are responsible for their processing.

An agent's knowledge base initially has no instances. The exception is the *vanet:Role* class which initially has four instances, indicated in Figure 4.3, which are the four different types of roles an agent can have from Section 2.5. The knowledge base is dynamically populated with instances through requests by the VANET-Reasoner component. The VANET-Reasoner component makes requests to the VANET-Ont component to populate the agent's knowledge base with Vont instances that will be used to create KP propositions, such as locations and foreign agents. These requests are triggered dynamically when new information is encountered, such as encountering a new agent, which the resident agent previously had no contact with, or encountering a new road previously untraveled. The full range of requests may be specific to the implementation. The ones used in our simulation are discussed in Chapter 5.

Instances can also be modified under certain circumstances. The most common type of modification is to agent instances, which define the resident agent's model of a particular agent. Example modifications can be an update to the trust variable as a result of reasoning about advice received from the agent about a particular road, or an update to the agent's location variable after receiving a message from the agent about its current location.

4.1.3 KP Ontology

The Knowledge Provenance ontology (KP) introduces the important concept of propositions to our model's knowledge base. The ontological structure for these propositions can be seen in Figure 2.2 and 4.4. A proposition is an atomic piece of information that is annotated with provenance information. It is through these KP propositions that our model can provide specific information such as the source agent and domain specific data such as, in our application, road congestion values, to be used in trust and certainty evaluations. The hierarchal nature shown in Figure 2.2, and the expanded version shown in Figure 4.4, describes the different types of proposition classes that can be used by our proposed model. The practical use of these propositions is defined by an implementation of this proposed model, and is described in Chapter 5. KP propositions are typically created and then stored for future use. [1]

The different types of propositions also help classify the type of information they are presenting and annotating. The range of propositions and their descriptive nature can be used to describe a wide range of situations. For example, this thesis, for the

[1]They can be modified later on; however the implementation of this thesis in Chapter 5 does not make use of post-creation modifications to propositions.

purposes of the implementation, utilizes Asserted Propositions *kp:AssertedProp* to describe information that was directly observed and Derived Propositions *kp:DerivedProp* to describe information that was indirectly observed or derived from other propositions. Direct and indirect observations are more thoroughly discussed in Section 4.3.8.

The KP propositions help annotate important domain information and present it in a standardized format. This thesis focuses on the VANET domain, with Vont describing its semantics. As discussed in the previous sections, Vont is used by the KP ontology as a descriptive layer, describing the information which is being annotated. An example would be a KP asserted proposition instance referencing an instance of *Vont:Traffic* which represents a traffic report of a location, reported by an agent defined in a variable of the proposition. A detailed example of a KP instance referencing Vont instances is shown in Figure 4.5. The arrows in the figure with the *io* label indicate an *instance of* relationship.

Figure 4.5 demonstrates the use of Vont and KP classes and instances to describe a message received about the congestion of a single road. The figure's contents are representative of a small portion of a single agent's knowledge base at a certain time. Black boxes are representative of classes, red boxes are representative of instances, contents of the boxes represent properties, and arrows represent relations. The figure contains an old instance of the KP Original Proposition class (kp:AssertedProp-10-FromAgent39) whose content property references a Vont traffic report (v-ont:Traffic-10At-21-FromAgent-39), and includes additional provenance information about the creator of the report (v-ont:Agent-39), when it was received, and the associated certainty. The creator of the report references an instance of the Vont Agent class (v-ont:Agent-39); relative properties are shown, such as the agent's role, identification number, and trust degree. The trust of the agent is reflected in the proposition instance. The proposition's referenced traffic report (v-ont:Traffic-10At-21-FromAgent-39) contains a congestion value, a reference to the road instance in question (v-ont:Road-21), and other variables used to calculate the confidence of the report. The road instance shows relative properties, such as last time updated, identification number, and trusted congestion value. The proposition is old because the time stamp for the road instance is higher than that of the proposition instance.

The use of the Vont ontology in conjunction with the KP ontology can semantically define anything from the VANET project. A large portion of this semantic and trust information is gathered from interations with other agents. Section 4.2.1 details messaging within the model. This includes the use of the model's ontology definitions to describe queries within messages.

4.2 Communication

The VANET agents communicate with other agents in order to collect important information that is to be stored in their knowledge base. The previous VANET research [20, 18] lacked a communication protocol or a clarification of the structure of a message's content. In this section, we define and describe these components as well as how messages are processed and used. The communication procedure for what is transmitted and how, is an expansion of the Knowledge Provenance (KP) distributed system communication model from [14].

A major change to the original way VANET approached messages is that previously an agent first had a demand for advice and then proceeded to message other agents. In the version of VANET presented in this thesis, agents are able to constantly receive messages, responses are stored, and then relevant information is used when there is a demand. The design rationale of this update was due to a practical infeasibility of sending, waiting, then receiving advice at the moment it is needed. Additionally, the unbiased nature of sending requests and storing advice allows for the model to profile many more agents even if their advice is not used for travel decisions.

4.2.1 Communication Protocols

The KP communication model [14] was designed with a pull based communication protocol, where agents send requests to other agents for information. In addition to this classic pull oriented design of the KP model, this thesis introduces a push based protocol for broadcasting information. These protocols dictate when communication is initiated and to whom. Either or both of the two protocols can be used for communicating information between agents. Both protocols are used in the simulations of Chapter 5, with a separate simulation demonstrating their performance differences.

This thesis uses two types of messages which are designed for the various protocols: 1) the fundamental and basic message is a location and congestion transmission; 2) the other message is to request the congestion information of a specific road. The following subsections describe each of the protocols, how they send these messages, and their use in combination. The presented framework is capable of functioning with only the pull or push based protocol active, or their use in combination. This will be demonstrated with simulations in Chapter 5.

4.2.1.1 Pull

The pull protocol allows agents (requester) to request information from other agents (requestee). The trustworthiness of the information from the requestee agent is modeled and used to determine what path to follow based on the report produced.

The overall protocol is capable of transmitting and receiving both types of messages. This subsection describes the most basic configuration of the model, which only requests and receives location and congestion information. Section 4.2.1.3 will include a description of the model which utilizes both pull and push protocols and further clarifies possible message types.

Algorithm 2 describes the pull-based protocol and the location and congestion request message to an agent. The algorithm is triggered according to a set communication frequency. The information received generates an update to the knowledge base of the recipient.

Algorithm 2: Pull Based Communication Protocol

```
while on the road do
    if Triggered according to communication frequency then
        Request location and congestion of an agent;
    end
end
```

4.2.1.2 Push

A push protocol allows agents to send information to other agents, even if it were not requested. The trustworthiness of the sender agent is still modeled; this may then be employed during decision making about travel paths. Algorithm 3 describes the push-based protocol and the location and congestion transmission message of an agent. The algorithm is triggered according to a set communication frequency. The information received generates an update to the knowledge base of the recipient.

Algorithm 3: Push Based Communication Protocol

```
while on the road do
    if Triggered according to communication frequency then
        //Broadcast current location and congestion to agents
        Broadcast current location and congestion;
    end
end
```

4.2.1.3 Pull and Push

This subsection describes the complete configuration of the model, which utilizes both protocols and message types. This configuration is used as a default in the simulations of Chapter 5. In this configuration, for scenarios where ongoing broadcasts from vehicles are supported, the push protocol is utilized for transmitting the location and congestion message while the pull protocol is utilized for the second message type, where congestion information is requested for a certain road.

The push based protocol is utilized for transmitting current location and congestion information, instead of the pull based protocol, because it allows for an easier flow of information due to a reduction in the number of messages needed for a basic location and congestion transmission. The pull based protocol requests the current location and congestion of another agent and then receives the response at a later time. Using the push protocol, only one message is needed because each agent can simply message other agents its current location and congestion rather than requiring vehicles to also send an unnecessary request.

The second message type, for retrieving congestion information about a certain road, has two sub messages, a request and a response, and is initiated by a pull protocol. Due to the possibility that the requestee agent may not have any information on the requested road, it is possible that no response is sent and received. The information received from these messages are stored in the requester agent's knowledge base as a proposition, described in Section 4.1.3 and 4.3.2, and may be used or judged when appropriate. The content of these messages are explained in Section 4.2.2.

This thesis encountered an issue with the pull based protocol suggested by the VANET research in terms of requesting specific information. Examining the algorithm proposed by the VANET model of Section 2.5, the messaging proposed was vague. It was suggested that the message content (congestion information about a road) would be a "yes" or "no" response to a question "Is this road congested?" and that this response would be pulled to the requesting agent. When the pulls would occur was also left vague as "in need of advice". As such, which roads were being investigated was also unspecified. This thesis addresses this issue by introducing the concept of priority roads, which are roads that are dynamically designated as important for an agent. The concept of a priority road facilitates messaging (and hence the updating of user models). Roads are placed into priority for an agent through the framework dynamically recognizing a lack of congestion information, which may have otherwise altered the decision of the agent. The subsequent request for congestion information concerning the priority road will fill the agent's earlier gap of congestion information. Note that when a report about a priority road is received, its status may be altered to cause it to be removed from the priority list (if sufficient information on that road has accumulated).

Algorithm 4 describes the push and pull based protocol and how a priority road information request is sent by agents. The algorithm is triggered according to a set communication frequency. We return to briefly discuss priority roads in connection with path decision making in Section 4.3.1.

The rationale in having two types of communication protocols is to allow for an easier flow of information as well as future model development, such as the existence of a global agent which would use the push protocol to send important information such as news of a car crash. This thesis uses only two sets of messages, three unique messages in total. These are outlined in Table 4.1 and described in the following subsection. The model's communication protocols are designed however to be extendable in so that it can host a multitude of different messages. The pull protocol could be used for any type of information gathering, such as a personal inquiry about the receiving agent or less direct inquiry, for example about a third agent.

Algorithm 4: Pull and Push Based Communication

```
while on the road do
    if Triggered according to communication frequency then
        //Pull protocol
        //Get road to request advice about and agent to request from
        if priority road exists then
            Choose highest priority road;
            Get trustworthy agent;
            if Trustworthy agent exists then
                Send request to trustworthy agent for advice concerning the high
                priority road;
            end
            else
                Send request to any agent for advice concerning the high priority
                road;
            end
        end
        //Push protocol
        //Broadcast current location and congestion to agents
        Broadcast current location and congestion;
    end
end
```

4.2.2 Messages

The specific contents of messages are designed around the KP distributed communication work [14]. These messages are composed of two important components, in addition to the requester and requestee. The first is a *query*, specifying the question to be answered. The second is an answer in the form of a *fluent*. Huang defines fluents in his KP thesis as "... a property (of the world) whose value is dependent on situations." The Model's ontological definitions of things such as classes and variables, which are universally understood by agents in the model, are used in the query to define the question.

Table 4.1 details the contents of the three messages discussed in the previous Sections which are utilized in this thesis. The three messages are a transmission of an agent's location and congestion (Location and Congestion Push), a request for congestion information about a specific road (Priority Road Information Pull Request), and a response for congestion information about a specific road (Priority Road Information Pull Response). The queries of messages indicate either what the agent is requesting or sending by transmitting the corresponding ontology classes. The fluents of messages contain the content of the messages, typically a reference to a class instance and/or basic data types which convey some sort of information.

For example, a sample Priority Road Information Pull Request message sent by a requesting agent would reference the *v-ont:Location* and *v-ont:Traffic* classes in the query and then supply a reference to an instance of *v-ont:Location* such as (Highway401(*v-ont:location* instance)). The requestee would receive this message and recognize it as a request for congestion information about Highway401. If the requestee agent is honest it will respond with the Priority Road Information Pull Response message, for example (Highway401(*v-ont:location* instance),23(roads congestion value)) as its fluent. The requesting agent would recognize the fluent and interpret it as a congestion report on the location, and an appropriate KP proposition instance would be created.

In observation of the messages from Table 4.1, they all have the same *Query* but different *Fluents*. This is due to messages between agents consisting only of traffic information. Future work in Section 6.2 in Chapter 6 discusses other types of messages that could be utilized by the proposed model and thesis, such as requests for trust information about other agents.

4.2.3 Processing

This section details the processing of the communication protocols, use of message contents, and the interaction of VANET-Comm with the other components of our framework shown in Figure 3.1.

The Model's messaging system is thoroughly described by Figures 4.6, 4.7, 4.8, and 4.9. The figures detail how the model uses the JiST/SWANS Controller component to send messages between agents, the layered processing of the messages by VANET-

Table 4.1: Pull and Push Message Contents

	Location and Congestion Push	**Priority Road Information Pull Request**	**Priority Road Information Pull Response**
Sending Agent	Resident Agent	Requester Agent	Requestee Agent
Receiving Agent	Foreign Agent	Requestee Agent	Requester Agent
Query	([*v-ont:Location*], [*v-ont:Traffic*])	([*v-ont:Location*], [*v-ont:Traffic*])	([*v-ont:Location*], [*v-ont:Traffic*])
Fluent	((*v-ont:Location instance*), (congestion value))	(*v-ont:Location instance*)	((*v-ont:Location instance*), (congestion value))

Comm, with its use of the KP distributed communication model, and the management and use of the semantic information by the VANET-Reasoner and VANET-Ont components. With the exception of Figure 4.6, the figures are organized into the order they would be processed chronologically. The first, Figure 4.6, is the process in which an agent would send a push message. The second, Figure 4.7, is an agent requesting advice. This figure also signifies the start of a sequence of events and messages which are conveying in the remaining figures. The third, Figure 4.8, is an agent receiving an advice request then sending a response. The last is Figure 4.9, an agent either receiving a response to their original request or receiving a push message. Messages that contain congestion information are processed by adding them to the knowledge base and executing the trust modeling process of Algorithm 7 to calculate a majority opinion. Algorithm 7 is more thoroughly detailed in Section 4.3. The sequence diagrams depict the flow of processing and how components communicate with each other to complete the operation.

The sequence diagram of Figure 4.7 shows the flow of processing for an agent executing a pull protocol request to another agent for advice. This information is to be stored in this agent's knowledge base and used in path planning and agent profiling.

The VANET-Comm: PushProtocol is a continually executing process which is responsible for periodically pushing advice to agents. This process only exits when advice no longer needs to be pushed. The process begins by retrieving an agent and then creating a knowledge provenance fluent, contained within a query to embody the information being pushed. This query is then issued as a task to the part of VANET-Comm which contains the knowledge provenance distributed communication model (per Section 3.3), described in Section 3.2.2, which is responsible for processing all communication tasks. After the task is processed by KP, the message is sent to the requestee agent. After the message is sent, PushProtocol waits for a certain amount of time and then restarts this sequence shown in Figure 4.6.

We assume that the frequency in which these messages are sent is set high enough (every 5-16 seconds for example) so that over a certain amount of time the number of messages sent will emulate a broadcast. The design rationale for sending a message according to a frequency, and not a set of messages as a broadcast, is so that channels will not be clogged with too many messages (which may be very large depending on the broadcast feature).

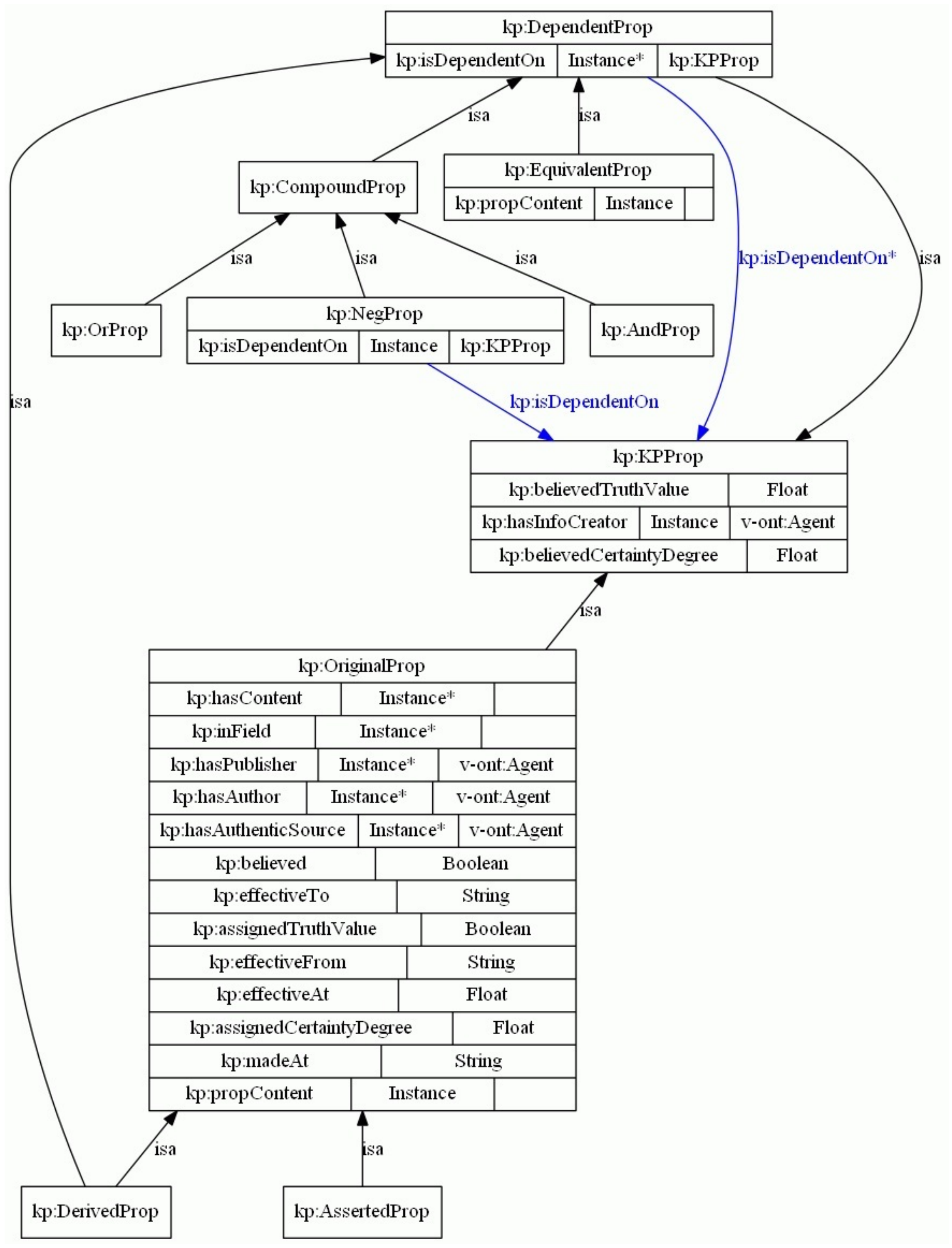
kp:DependentProp
kp:isDependentOn | Instance* | kp:KPProp
isa
kp:CompoundProp
kp:EquivalentProp
kp:propContent | Instance
kp:isDependentOn*
kp:OrProp
kp:NegProp
kp:isDependentOn | Instance | kp:KPProp
kp:AndProp
kp:isDependentOn
kp:KPProp
kp:believedTruthValue | Float
kp:hasInfoCreator | Instance | v-ont:Agent
kp:believedCertaintyDegree | Float
kp:OriginalProp
kp:hasContent | Instance*
kp:inField | Instance*
kp:hasPublisher | Instance* | v-ont:Agent
kp:hasAuthor | Instance* | v-ont:Agent
kp:hasAuthenticSource | Instance* | v-ont:Agent
kp:believed | Boolean
kp:effectiveTo | String
kp:assignedTruthValue | Boolean
kp:effectiveFrom | String
kp:effectiveAt | Float
kp:assignedCertaintyDegree | Float
kp:madeAt | String
kp:propContent | Instance
kp:DerivedProp
kp:AssertedProp

Figure 4.4: Knowledge Provenance Ontology showing all variables.

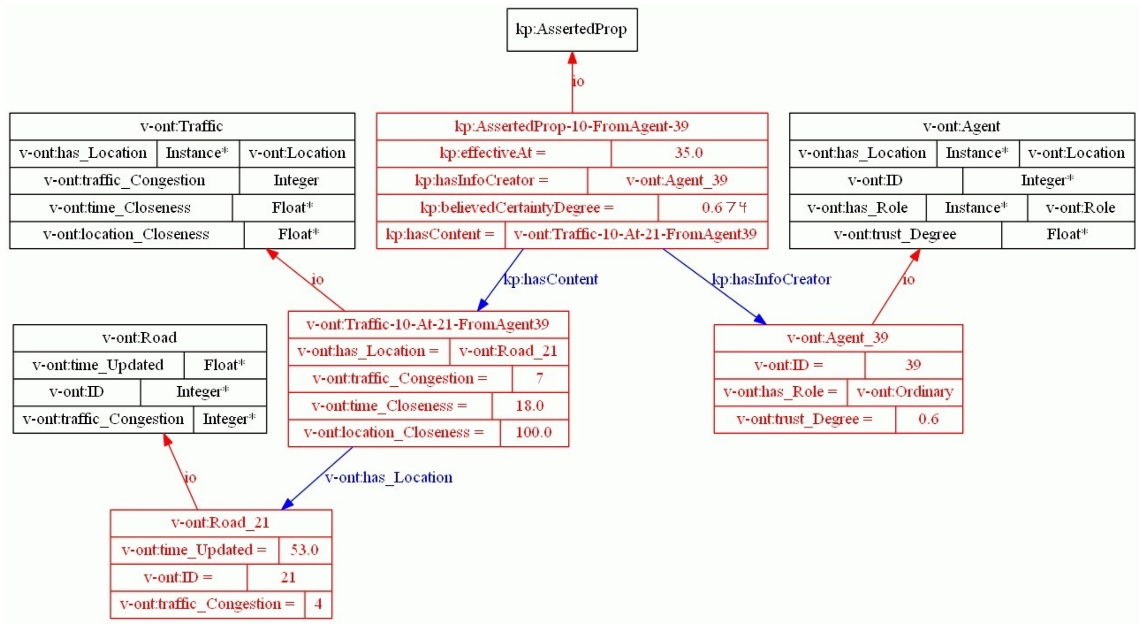
kp:AssertedProp
io
kp:AssertedProp-10-FromAgent-39
kp:effectiveAt = 35.0
kp:hasInfoCreator = v-ont:Agent_39
kp:believedCertaintyDegree = 0.674
kp:hasContent = v-ont:Traffic-10-At-21-FromAgent39
v-ont:Traffic
v-ont:has_Location Instance* v-ont:Location
v-ont:traffic_Congestion Integer
v-ont:time_Closeness Float*
v-ont:location_Closeness Float*
v-ont:Agent
v-ont:has_Location Instance* v-ont:Location
v-ont:ID Integer*
v-ont:has_Role Instance* v-ont:Role
v-ont:trust_Degree Float*
kp:hasContent
kp:hasInfoCreator
v-ont:Traffic-10-At-21-FromAgent39
v-ont:has_Location = v-ont:Road_21
v-ont:traffic_Congestion = 7
v-ont:time_Closeness = 18.0
v-ont:location_Closeness = 100.0
v-ont:Agent_39
v-ont:ID = 39
v-ont:has_Role = v-ont:Ordinary
v-ont:trust_Degree = 0.6
v-ont:Road
v-ont:time_Updated Float*
v-ont:ID Integer*
v-ont:traffic_Congestion Integer*
v-ont:has_Location
v-ont:Road_21
v-ont:time_Updated = 53.0
v-ont:ID = 21
v-ont:traffic_Congestion = 4

Figure 4.5: Example of KP and Vont instances describing a traffic report.

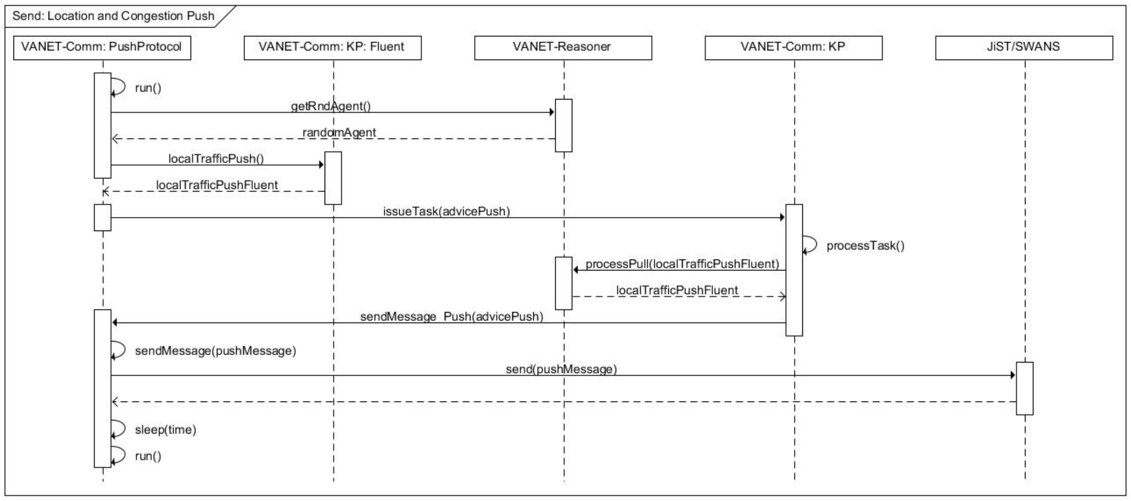

Figure 4.6: UML Sequence Diagram of an agent pushing their current location and congestion information to another agent.

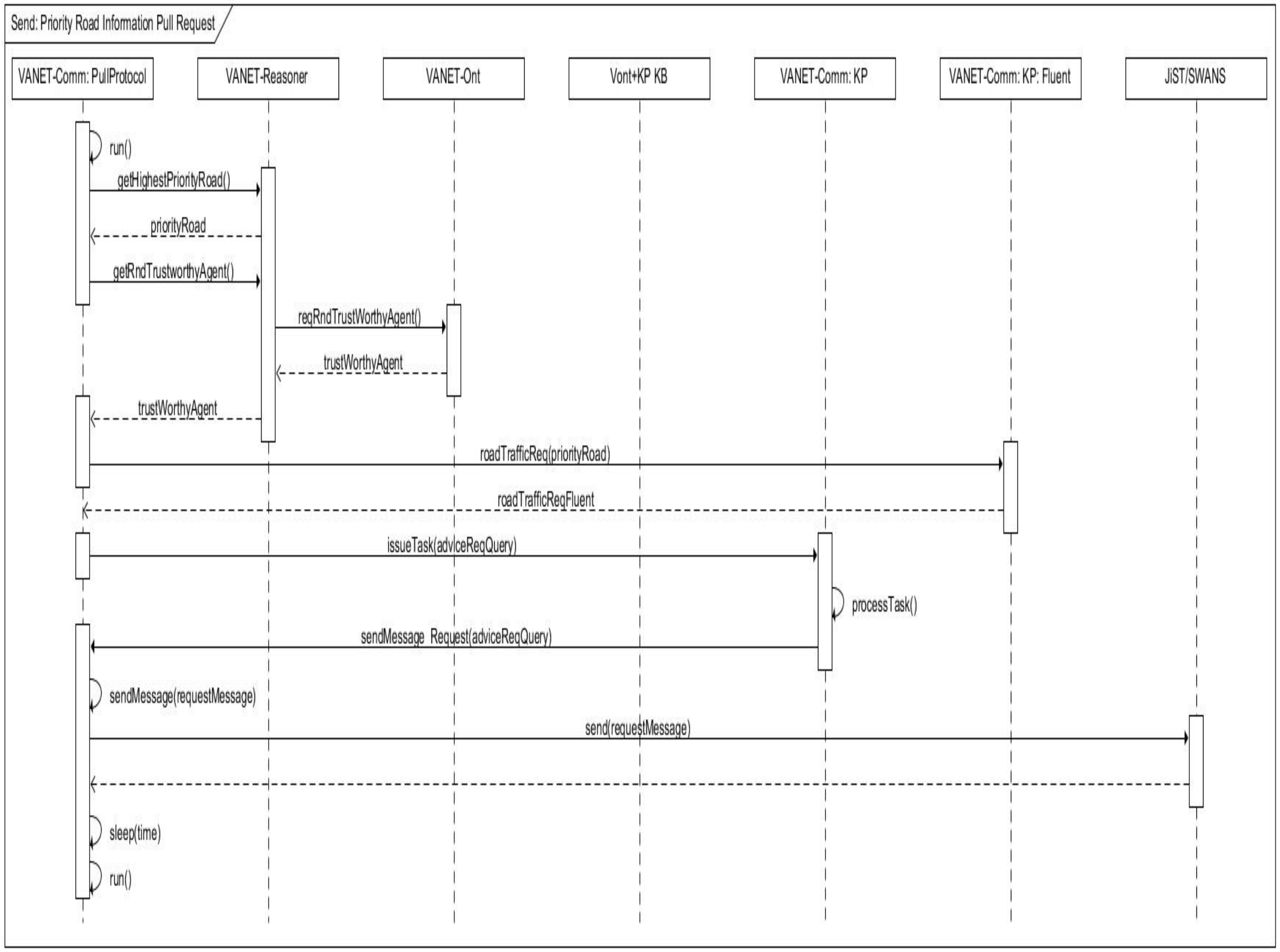

Figure 4.7: UML Sequence Diagram of an agent sending request for information about a priority road.

The VANET-Comm: PullProtocol, similar to PushProtocol, is a continually executing process that is responsible for periodically requesting advice from agents. A loop is formed which is indicated by the last two method calls. This loop only exits when advice no longer needs to be collected. The requesting process begins by retrieving an agent. A knowledge provenance fluent is then created and contained within a query to embody the information being requested. This query is then issued as a task to the part of VANET-Comm which contains the knowledge provenance distributed communication model, which is responsible for processing all communication tasks. After the task is processed by KP, the message is sent to the requestee agent. After the message is sent, PullProtocol waits for a certain amount of time, set by the communication frequency, and then starts this sequence shown in Figure 4.7 again.

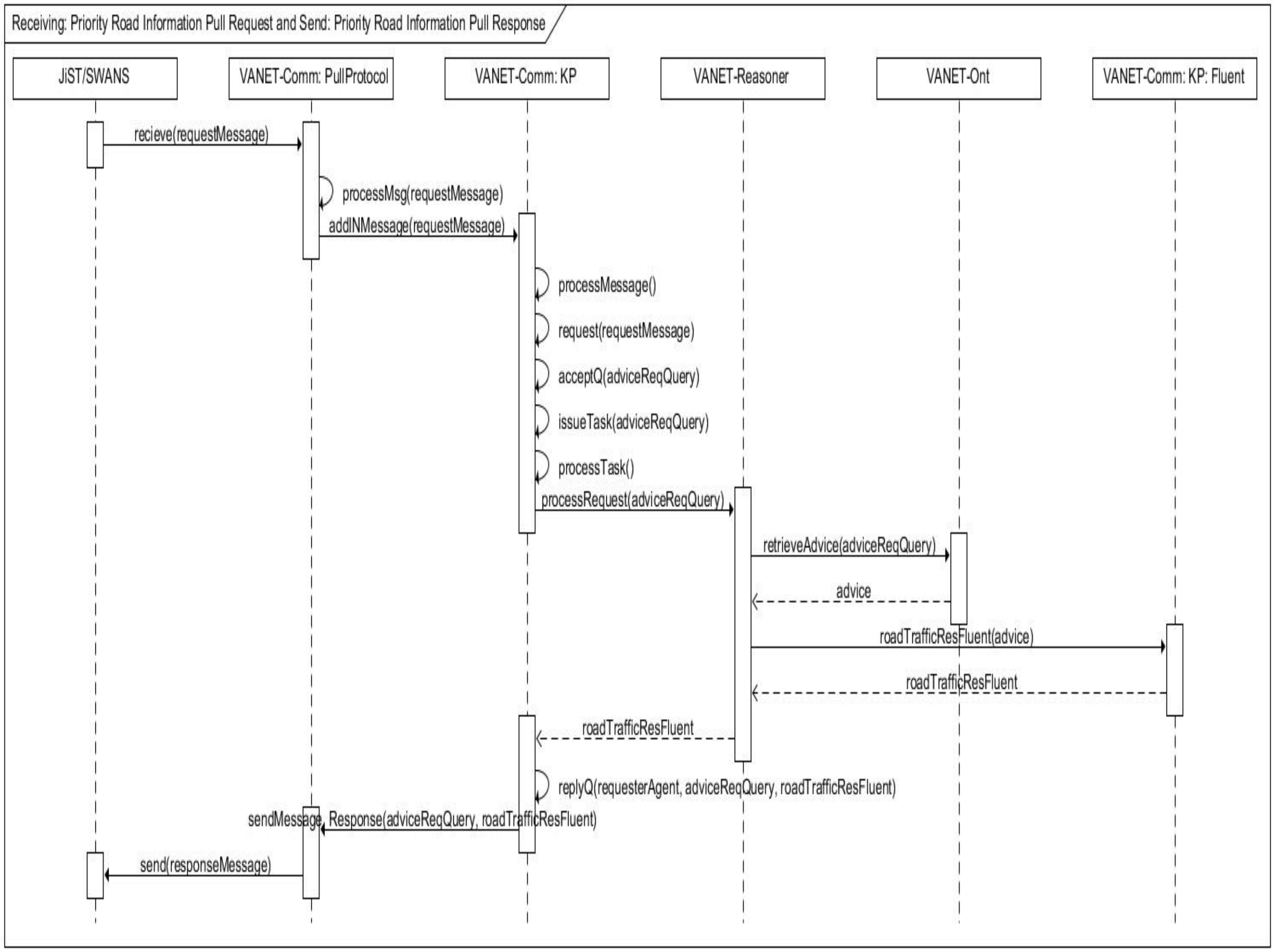
Receiving: Priority Road Information Pull Request and Send: Priority Road Information Pull Response
JiST/SWANS
VANET-Comm: PullProtocol
VANET-Comm: KP
VANET-Reasoner
VANET-Ont
VANET-Comm: KP: Fluent
recieve(requestMessage)
processMsg(requestMessage)
addINMessage(requestMessage)
processMessage()
request(requestMessage)
acceptQ(adviceReqQuery)
issueTask(adviceReqQuery)
processTask()
processRequest(adviceReqQuery)
retrieveAdvice(adviceReqQuery)
advice
roadTrafficResFluent(advice)
roadTrafficResFluent
roadTrafficResFluent
replyQ(requesterAgent, adviceReqQuery, roadTrafficResFluent)
sendMessage, Response(adviceReqQuery, roadTrafficResFluent)
send(responseMessage)

Figure 4.8: UML Sequence Diagram of an agent receiving a request for information about a specific road and sending a response.

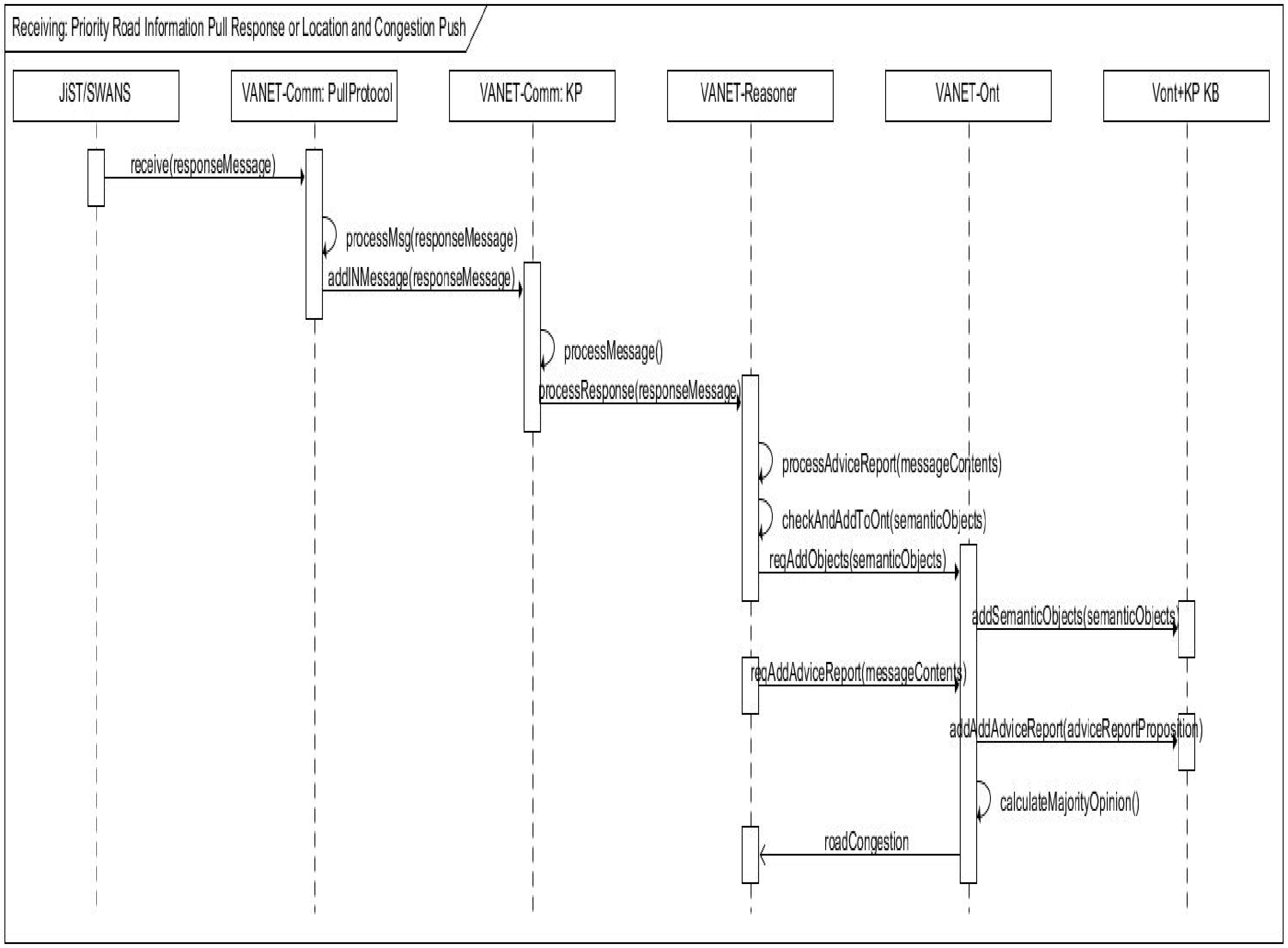
Receiving: Priority Road Information Pull Response or Location and Congestion Push
JiST/SWANS
VANET-Comm: PullProtocol
VANET-Comm: KP
VANET-Reasoner
VANET-Ont
Vont+KP KB
receive(responseMessage)
processMsg(responseMessage)
addINMessage(responseMessage)
processMessage()
processResponse(responseMessage)
processAdviceReport(messageContents)
checkAndAddToOnt(semanticObjects)
reqAddObjects(semanticObjects)
addSemanticObjects(semanticObjects)
reqAddAdviceReport(messageContents)
addAddAdviceReport(adviceReportProposition)
calculateMajorityOpinion()
roadCongestion

Figure 4.9: UML Sequence Diagram of an agent receiving information about a road.

The sequence diagram of Figure 4.8 is a direct result of the process from Figure 4.7. It describes an agent receiving the request from the previous sequence diagram, processing it, and sending a response accordingly with the requested advice. However, it is important to indicate that in the trust domain of VANET, if this receiving agent is not honest then the resulting advice's information may be false.

The process initially begins by the JiST/SWANS controller component receiving the message and informing PullProtocol. PullProtocol subsequently processes the message by supplying it to the KP communication model. The KP communication model begins its processing of the message by determining it as a request, which is then accepted, issued as a task, and then processed as a task. The task is processed by the VANET-Reasoner component which retrieves the advice information from the VANET-Ont component and contains it within a fluent. The fluent is returned to the KP communication model and made into a reply which is sent to the requesting agent.

The sequence diagram of Figure 4.9 is a direct result of either sequence diagrams 4.8 or 4.6. The process begins in a way that is similar to Figure 4.8, by having the JiST/SWANS controller component receiving the message and informing PullProtocol (or PushProtocol), which processes it by supplying it to the KP communication model. The KP communication model recognizes that the message is a response, and allows VANET-Reasoner to process it. The advice is processed by first checking whether the advice information's objects already exist semantically in the knowledge base. Examples of such objects in VANET would be the requestee agent and its current location. If they do not exist then the missing objects are added. This is done by VANET-Reasoner sending a request to the VANET-Ont component to add the objects. VANET-Ont receives the request and adds the requested objects to the agent's knowledge base. Once the advice information's objects are checked and possibly added to the knowledge base, another request is made to add the advice report, which subsequently returns a new congestion calculation for the road.

4.3 Trust

This section outlines and details the incorporation of trust into the components of Figure 3.1 to facilitate the profiling of agents within the framework. Important modifications and updates to the Muti-faceted Trust Management Framework of Section 2.4 are also introduced and explained. Trust modeling is important because it drives decision making about path planning for vehicles.

The VANET trust modeling research utilized various types of trust to theoretically optimize profiling effectiveness and efficiency (see Section 2.4). The key trust components are separated into three areas, 1) experience based trust, 2) role based trust, and 3) majority opinion [20, 18]. Experience based trust, as the name implies, is trust that has been built up with a particular agent over time, eventually more greatly valuing their reports. Role based trust is trust that is assigned by default to agents of a particular role. Majority opinion is a summarization of reports from a collection of agents, with an aggregated trustworthiness value that must be above a threshold to be used. The calculation models time and location closeness and distinguishes direct and indirect reports.

We begin with a clarification of how our messaging framework would support trust modeling in the context of Boolean traffic reports. We then progress to a description of messaging and trust modeling in support of numeric-valued reports.

4.3.1 Advice Gathering Update

In Section 4.2.1, we proposed a messaging framework that has agents receiving traffic reports with a certain frequency, leading to updates to the knowledge base. This constitutes information gathering outside the case where "advice is needed".

We now clarify how this proposed information gathering does not radically change the use of trust and traffic information in Algorithm 1. Algorithm 1 theoretically sends requests only to agents in a prioritized list, when advice was needed. Our proposed update to this algorithm, shown in Algorithm 5, would have each agent's knowledge base continuously updated with periodic messages, from the pull, push or both protocols. When advice is needed, the most relevant and trustworthy reports are chosen and used.

Algorithm 5: New Majority Computation Steps, with Advice Gathering Update

```
while on the road do
    Send requests and receive responses;
    if in need of advice then
        Choose $n$ reports $R$; //number of reports to use for advice
        Check Priority Road(Current Road);//to help update the Priority list
        Prioritize $n$ reports; //according to roles and experiences
        if response consensus > acceptable ratio then
            Follow advice in response;
        end
        else
            Follow advice of user with highest role and highest trust value;
        end
    end
    Verify reliability of advice;
    Update users' trust values;
end
```

The work by Minhas et al. mentioned in Section 2.5 presented a Multi-faceted Trust Management Framework that was described as operational for Boolean values of congestion (Heavy (True), Light (False)). In order to calculate a majority opinion, reports which featured the same Boolean value of congestion were aggregated together.

The percentage of reports with same congestion value would be compared against a threshold to determine whether the advice would be followed. The trust modeling itself respects the formulae outlined in Section 2.5. The use of a new advice gathering protocol (as per Algorithm 4) would not intrinsically alter the majority opinion calculation; it simply clarifies how traffic reports are retrieved. Note that calling *Check Priority Road(Current Road)* within this algorithm has the eventual effect of coping with stale or missing information on roads that are critical to current path planning. Algorithm 6 describes the conditions for when a road is added to the list of priority roads. This ultimately achieves the same objective desired by the VANET model of Minhas et al., namely to be well informed about roads that are currently important for decision making.

Algorithm 6: Check Priority Road(Current Road)

```
if NO Rj for Current Road OR Rj for Current Road IS old then
    Add Current Road to list of priority roads;
end
```

4.3.2 Trust in the Ontology

The knowledge base described in Section 4.1 was chosen to store all trust information. This information is primarily stored as a variable of Vont agent and KP proposition instances. Three types of interactions are performed on trust variables of these instances. The first is **creation**: all agents are given a default trust initially and all propositions are assigned a trust value depending on the agent they are associated with. The second type of interaction is **retrieval** where trust values are retrieved for evaluating confidence and/or for considering propositions into a list of N priority propositions. The last interaction is the most complicated and involves **modification** of the agent's trust variable. This is only performed on agent instances to update within one agent's knowledge base its trust in the agent in question.

The trust update action is only performed by a reasoner within the VANET-ont component which is responsible for profiling agents. Agents are profiled when the model encounters information that suggests that the trustworthiness of an agent needs to be uploaded in the knowledge base. These evaluations are important to the advice processing algorithm of Figure 7, which will be described later in this Section, and are used to profile the agents and update their trust accordingly.

The proposed use of a knowledge base for storing user models and world specific data does not alter the calculations or use of trust related information, such as the majority opinion. The proposed knowledge base simply clarifies how information is stored and accessed. Many classes and properties of the proposed Ontology are mentioned in the following subsections and chapters to reference how and where information is stored.

4.3.3 Trust Modeling with Numeric Traffic Reports

In this section we clarify how our framework could support the use of numeric traffic reports, leading to a "confidence metric" used for trust modeling, in contrast to the Boolean evaluation of traffic in Section 2.4. Section 4.3.1 described how our proposed framework can support Boolean values of traffic and trust. Our new proposed confidence metric and use of numeric congestion and trust values serve to allow a more accurate description of traffic and agent information, which will be explained in this subsection.

The original theory in Section 2.5 assumed that congestion would be communicated as a simple $true$ (Heavy) or $false$ (Light), stating either that the road was congested or not. However, direct application may result in an unfair and biased calculation of the majority opinion. This is because determining whether a road is congested or not is a subjective opinion and is prone to inaccuracies. Also, by representing the congestion as a Boolean, this severely limits the system's ability to compare roads, evaluate agents, and make the best decisions. Our proposed model seeks to alleviate this problem by representing congestion as a number , which will bring a more suitable level of accuracy to the system [2].

Formula (2.3) shows the calculation for the aggregated effect of a majority opinion. The new way of representing congestion as a numeric value requires a careful recasting of formula (2.3). (2.3) aggregates the effect of all agents that sent the same report (i.e. cong = true). This simple aggregation of similar reports is impossible with the new congestion representation because there are no longer only two types of reports (Cong=true or Cong=false). In the new framework, each report must be evaluated for addition into the majority opinion system. This is done by giving the report a confidence and then evaluating it for inclusion into the majority opinion (similar to the aggregated effect calculation). The following sections will detail how the factors of experience and role based trust, time and location closeness, and whether the advice is direct or indirect are incorporated into our proposed confidence metric and utilized in calculating a majority opinion.

4.3.3.1 Confidence Calculation

Confidence functions as a metric similar to trust, and is calculated by combining many different report and agent factors, which were introduced in Formula 2.3 and will be described in detail later in this section. These factors include experience and role based trust, time and location closeness, and whether the advice is direct or indirect.

Our proposed Equation for calculating confidence must effectively replace Formula

[2]Note that a reported congestion value for instance of 23 would ideally be representing the actual number of cars on the road; in reality a car would be more likely to provide a value on a fixed scale to represent whether the traffic it observed were very heavy, moderate or light, for example. It may also be reasonable for cars to report their speed and for this to be used as a reflection of the road's congestion. In so doing, this deflects issues of number of cars relative to length of the road, for example.

2.3, while representing a trust-like metric. Modifications to confidence should then be reflected in a manner similar to how trust is increased and decreased in Equations 2.1 and 2.2. α and β function in these Equations as a standard for increasing and decreasing trust, respectively. For our proposed confidence calculation it did not make sense to atomically increase or decrease the value according to the influencing factor (role, time closeness, etc.). The increase or decrease should reflect the significance of the factor. As a result, our proposed confidence metric replaces Formula 2.3 with Equation 4.1, where Equations 2.1 and 2.2 are used as the basis for calculating the confidence of report R_j, through a modified summation of a geometric series [3].

The factors of role based trust, time and location closeness, and whether the advice is direct or indirect in formula (2.3), are reflected through Variable (G). Experience based trust of an agent automatically forms the default value of the confidence metric ($CurrConf(R_j)$). Variable (G) represents the number of times[4] to increase or decrease confidence. G's calculation is specific to each factor. If G is calculated to a negative value, this indicates that β should be used instead of α. Examples are shown in Section 4.3.9. The following sections briefly detail how each factor influences G; however the exact calculations are dependent on how parameter values are chosen, within an implementation.

$$Conf(R_j) = (CurrConf(R_j) - 1)(1 - (\alpha \text{ } or \text{ } \beta))^{|G|} + 1 \tag{4.1}$$

As an example, Figure 4.5 presents the use of Vont and KP classes and instances to describe a message received about the congestion of a single road. Factors of experience and role based trust, time and location closeness, and whether the report is direct or indirect are shown in the knowledge base instances through the parameters described in the previous section. These parameters would be used to calculate the confidence metric presented in parameter *kp:believedCertaintyDegree*. Section 4.3.9 demonstrates a mathematical calculation of the confidence according the default implementation parameters.

4.3.4 Majority

Majority based trust is incorporated into our framework as a core algorithm for determining the trustworthiness of an agent, to then dictate whether to believe the congestion value reported about a road, which influences path planning. Section 2.4 describes majority based trust as a consensus, with a value which has been agreed upon by many agents. For our proposed non-Boolean extension to trust modeling, majority based trust is described as an opinion, where a similar value has been agreed

[3]A Geometric series is necessary because the calculations are capturing atomic increases in trust values but we are reasoning about non-Boolean factors that are therefore not atomic. See Appendix A for a fuller depiction of the geometric series in question.

[4]Note that we use the absolute value of G as the exponent in order to ensure that the number of times is a positive number.

upon by many agents. The rationale for the change from a Boolean based congestion value to a numerical congestion value was described in Section 4.3.3.

The advice is used by choosing and prioritizing information from various reports and calculating a majority opinion, which is followed if its confidence is above a threshold, similar to the threshold of Equation 2.5. The primary advice presented in the VANET project would be road congestion reports, which would be used to help an agent decide what roads to take and which to avoid by considering all the facets of the multidimensional trust model. This continues to hold in our framework. In our calculation, if the confidence is below a threshold, then the advice is used from the report with the highest confidence.

The majority opinion is calculated using Algorithm 7. All relevant advice report propositions referencing an object, for example a location in the VANET domain, are retrieved and prioritized into a list of size n. The majority opinion is then calculated, stored, and reported back to VANET-Reasoner. During the processing of potential propositions, if a report contains information that is suspicious, such as an extremely high congestion report, the sender is reported as a suspicious agent. Labeling agents as suspicious is helpful in order to remove them from consideration, regardless of their current trustworthiness value. The reasoner component of VANET-Ont, which is responsible for profiling agents and updating trust values, would process the suspicious agent, profiling it and updating its trust value in the knowledge base.

4.3.4.1 Majority Calculation

Algorithm 7 is a modified algorithm from Algorithm 1, which shows the calculation of a majority opinion in the framework. The algorithm uses suspicious agent detection in helping to avoid the inclusion of congestion advice which is outside a standard deviation from the current majority congestion. The majority opinion is used if there are at least n agents to use advice from and the majority confidence is above the majority threshold.

4.3.4.2 Suspicion Calculation

Suspicion detection is important to include to help avoid congestion advice that greatly deviated from the current majority. Only using advice that has similar congestion reports forms our majority opinion, rather than conceiving of majority opinion as just an average congestion of the highest trusted agents (n).

If an agent is deemed suspicious, then they are reported and the agent's advice is not used in the majority opinion calculation. However, the reverse is possible where if an agent's advice has higher confidence than the majority and confidence greatly deviates from the majority. If this happens then the majority confidence is decreased proportionally and the agent's advice is potentially used as the *report with highest confidence.*

Algorithm 7: New Majority Computation Steps, with Numerical Congestion Metric

```
while on the road do
    Send requests and receive responses;
    if in need of advice then
        Choose $n$ reports $R$; //number of reports to use for advice
        Check Priority Road(Current Road);//to help update the Priority list
        Prioritize $n$ reports; //according to Confidence (roles, experiences, time,
        location, and if report is indirect or direct)
        foreach n reports do
            if $R_j$ suspicious then
                Report suspicious agent $R_j$;
            end
            else
                Include report $R_j$ in Majority;
            end
            if Majority suspicious then
                Decrease Majority confidence;
            end
        end
        if Majority confidence > acceptable threshold && Number of reports >
        n threshold then
            Follow advice in response;
        end
        else
            Follow advice of report with highest confidence;
        end
    end
    Verify reliability of reports;
    Update users' trust values;
end
```

4.3.5 Experience

Experience based trust is the most basic type of trust and is applied to every agent in our model framework. As detailed in Section 2.4, it is trust as a result of direct experiences with the individual agent. This is emulated in our framework model through attributing a trust value (*v-ont:trust_degree*) to each *v-ont:Agent* instance in the resident agent's knowledge base, shown in Figure 4.3 and 4.5, and updating when the model encounters information that it can use in a judgmental nature. An example of such information would be from detecting suspicious information being reported by an agent, encountering definitive information that can be used as a comparison factor against information previously reported by an agent, or processing the opinion of a more trusted agent about the agent in question. Since experience based trust is the most basic type of trust, this forms the basis of the confidence calculation.

This facet of trust management is very simple but powerful. The thesis' implementation in Chapter 5 demonstrates this through *basic* simulations which only use experience and majority based trust.

4.3.6 Role

Experience based trust is a powerful tool for profiling agents; however, it is often challenged in scenarios with data sparsity. Data sparsity is an absence of agents with which the resident agent has had previous experience. This is often the case in the real world where it is rare to encounter a car which you have previously profiled.

Role based trust helps alleviate the issue of data sparsity by assigning roles to agents in our model framework. As detailed in Section 2.4, predefined roles (e.g. police patrols, traffic reporters or taxi drivers) are assigned to all agents in the system. Different roles may be associated with different levels of trust. The Model uses the four different types of roles, motivated by the classification of Minhas et al: *Ordinary*, *Seniority* (e.g. commuter pool), *Authority* (e.g. news station car), *Expert* (e.g. police). These roles are stored as default instances of the *v-ont:Role* class and referenced in the *v-ont:has_Role* variable of *v-ont:Agent* instances, shown in Figure 4.3 and 4.5.

Role based trust is incorporated into a proposition's confidence calculation by increasing it by a magnitude proportional to the particular role's rank. Role rank of each role is shown in Figure 4.3 through the *v-ont:role_Rank* variable. Equation 4.2 shows how G is calculated for Equation 4.1. *RPenal* is a standard value for weighting roles, and *RoleRank* is the rank of the roles. G is inversely proportional to *RoleRank* so that higher roles (*Authority* has *RoleRank* of 2) warrant greater increases in confidence.

$$G = RPenal/RoleRank \tag{4.2}$$

4.3.7 Time/Location

It can often be the case that an agent receives a great deal of reports about a road, with some being more accurate than others. A combination of time and location closeness is used in confidence calculations to determine how accurate reports are. Time closeness is a measure of how old the report is with respect to when the advice is needed. Location closeness is a measure of how how far the agent providing the report is to the road in question.

Time and location closeness helps alleviate the issue of old and inaccurate reports by assigning these metrics to traffic report propositions and using them in confidence calculations in our model framework. As detailed in Section 2.4, metrics of time and location closeness are used in calculating a majority consensus. Our proposed model similarly uses these metrics in calculating a majority opinion, through modifying the confidence of propositions by a magnitude inversely proportional to these metrics.[5] These metrics are stored in the *v-ont:time_Closeness* and *v-ont:location_Closeness* variables of *v-ont:Traffic* instances, shown in Figure 4.3 and 4.5.

Equations 4.3 and 4.4 show how G is calculated for Equation 4.1. *TPenal* and *LPenal* are standard values for weighting time and location respectfully. *TimeDifference* and *LocDifference* are time difference and location difference respectively. *MultiplicativeFactor* is a standard multiplicative factor for the calculation (max confidence increase will be *MultiplicativeFactor*, and not 1, if *TimeDifference* or *LocDifference* is 0.). The calculation finds the difference between *TimeDifference/LocDifference* and *TPenal/LPenal* and then divides the difference by *TPenal/LPenal*. This achieves the purpose of scaling the values to be within their unit metrics.[6]

$$G = (TPenal - TimeDifference)/TPenal * MultiplicativeFactor \tag{4.3}$$

$$G = (LPenal - LocDifference)/LPenal * MultiplicativeFactor \tag{4.4}$$

4.3.8 Direct/Indirect

The model framework of this thesis also incorporates the distinction of direct and indirect reports. Direct reports are reports which have been directly observed and reported by an agent. Indirect reports are direct reports of a third agent which are stored in the knowledge base of the agent the resident agent is communicating with.

For example, when one agent (Ar) communicates with another agent (A2) through a pull request concerning a priority road (R1), A2's highest confidence traffic report concerning R1 may have been reported by another agent (A3) and not A2. A2 would send Ar the report and indicate that it is an indirect report (A2 did not create

[5]This is consistent with the placement of these factors in the denominator of Equation 2.3. The specific magnitude used in the implementation of Chapter 5 is explained in Section 5.3.

[6]This required scaling was not considered in sufficient detail in the model of Minhas et al. and Equation 2.3.

the report), which would include A2's confidence of the report. A2 calculates the confidence using the report's experience and role based trust, and time closeness [7].

The inclusion of indirect reports, as opposed to only allowing direct reports, is important because it greatly increases the response rate of a pull request concerning a priority road. Indirect reports, however, may be more inaccurate than direct reports. This is taken into consideration through the use of the corresponding agent's confidence of the report (A2's confidence of the report) and by modifying the confidence value of a report by a predetermined factor.

As detailed in Section 4.1.3, the KP ontology is used to distinguish direct and indirect propositions. [8]

Equation 4.5 shows how G is calculated for Equation 4.1. $InPenal$ is a standard value for penalizing indirect reports, and *IfIndirect* is 1 if the report is indirect and 0 otherwise.

$$G = InPenal * \mathit{IfIndirect} \tag{4.5}$$

4.3.9 Confidence Calculation Examples

This subsection presents two examples which describe how the confidence metric for a report is calculated according to the multidimensional trust factors of experience and role based trust, location and time closeness, and whether the report is indirect or not. The following examples will show iterative modifications to the confidence value of a report according to the various factors.

The following calculation demonstrates how the confidence value for the report in Figure 4.5 was calculated. Note that all the parameter values used in these examples are the ones used in our implementation [9](described in Chapter 5).

[7]Location closeness is not incorporated because it is dependent on the agent who is using the report.

[8]Chapter 5 describes the implementation's specific use of the KP classes to distinguish direct and indirect propositions.

[9]However, we use InPenal=-2 in the example here instead for a more effective illustration.

Example 1: (illustrating α)

$$
\begin{aligned}
\text{Confidence} &= \textit{Agent\_39:trust\_degree}\ (0.6) \\[1em]
G_{time} &= (\text{TPenal}(90)\text{-TimeDiff}(18))/\text{TPenal}(90) \\
&\quad *\text{MultiplicativeFactor}(1.5) \\
G_{time} &= 1.2 \\[1em]
\text{Confidence}(0.6) &= (\text{Confidence}(0.6)\text{-}1)(1-\alpha)^{|G_{time}|}{+}1 \\
\text{Confidence} &= 0.6475 \\[1em]
G_{loc} &= (\text{LPenal}(200)\text{-LocDiff}(100))/\text{LPenal}(200) \\
&\quad *\text{MultiplicativeFactor}(1.5) \\
G_{loc} &= 0.75 \\[1em]
\text{Confidence}(0.6475) &= (\text{Confidence}(0.6475)\text{-}1)(1-\alpha)^{|G_{loc}|}{+}1 \\
\text{Confidence} &= 0.674
\end{aligned}
$$

Example 2: (illustrating β)

$$
\begin{aligned}
\text{Confidence} &= \textit{Agent\_41:trust\_degree}\ (0.7) \\[1em]
G_{role} &= \text{RPenal}(8)/\text{RoleRank}(2) \\
G_{role} &= 4 \\[1em]
\text{Confidence}(0.7) &= (\text{Confidence}(0.7)\text{-}1)(1-\alpha)^{|G_{role}|}{+}1 \\
\text{Confidence} &= 0.8032 \\[1em]
G_{time} &= (\text{TPenal}(90)\text{-TimeDiff}(180))/\text{TPenal}(90) \\
&\quad *\text{MultiplicativeFactor}(1.5) \\
G_{time} &= \text{-}1.5 \\[1em]
\text{Confidence}(0.7813) &= (\text{Confidence}(0.7813)\text{-}1)(1-\beta)^{|G_{time}|}{+}1 \\
\text{Confidence} &= 0.7413 \\[1em]
G_{loc} &= (\text{LPenal}(200)\text{-LocDiff}(500))/\text{LPenal}(200) \\
&\quad *\text{MultiplicativeFactor}(1.5) \\
G_{loc} &= \text{-}2.25 \\[1em]
\text{Confidence}(0.7604) &= (\text{Confidence}(0.7604)\text{-}1)(1-\beta)^{|G_{loc}|}{+}1 \\
\text{Confidence} &= 0.6100
\end{aligned}
$$

$$\begin{aligned}
G_{indirect} &= \text{InPenal(-2)*IfIndirect(1)} \\
G_{indirect} &= \text{-2} \\
\\
\text{Confidence(0.6991)} &= \text{(Confidence(0.6991)-1)}(1-\beta)^{|G_{indirect}|}\text{+1} \\
\text{Confidence} &= 0.4385
\end{aligned}$$

4.3.10 Travel Decisions when using Numeric Trust Modeling

Algorithm 7 clarifies whether an agent will choose to take a certain road or not based on consensus about the congestion on the road. If the agent wants to reason about which road to choose (from a set of possible roads), it can run Algorithm 7 for each road [10].

4.4 Model Summary

In this chapter we have presented our model and described each of its proposed components. Figure 4.10 describes the high level processing of our model framework and how each of the components from Figure 3.1 work together.

Figure 4.10 refers to many of the processes and algorithms we have described in this chapter. The activity diagram elements within *VANET-Ont: Mediator* shows the mediation of requests from *VANET-Reasoner* and the use of the knowledge base (*Process Request (Uses KB)*). *VANET-Ont: Reasoner* outlines how the component waits and processes requests that profile agents and update their experience-based trust. *JiST/SWANS* briefly defines where the controlling of the vehicle is done (driving/pathing/simulating) and displays its interactions with the other components through information discovery (*Encounter new Information*) and requests for congestion information (*Need Advice*). Where decision making about travel occurs is in *Run Simulation/ Drive to Destination*(driven by its path planning algorithm). *VANET-Reasoner* outlines the component's role in congestion advice retrieval, reasoning about *New Information* from *JiST/SWANS* and *VANET-Comm*, and sending requests to *VANET-Mediator*. *VANET-Reasoner* clarifies the push and pull protocols within the component and their interaction with *VANET-Reasoner*.

[10]Note that this is in fact what we do in our implementation in Chapter 5.

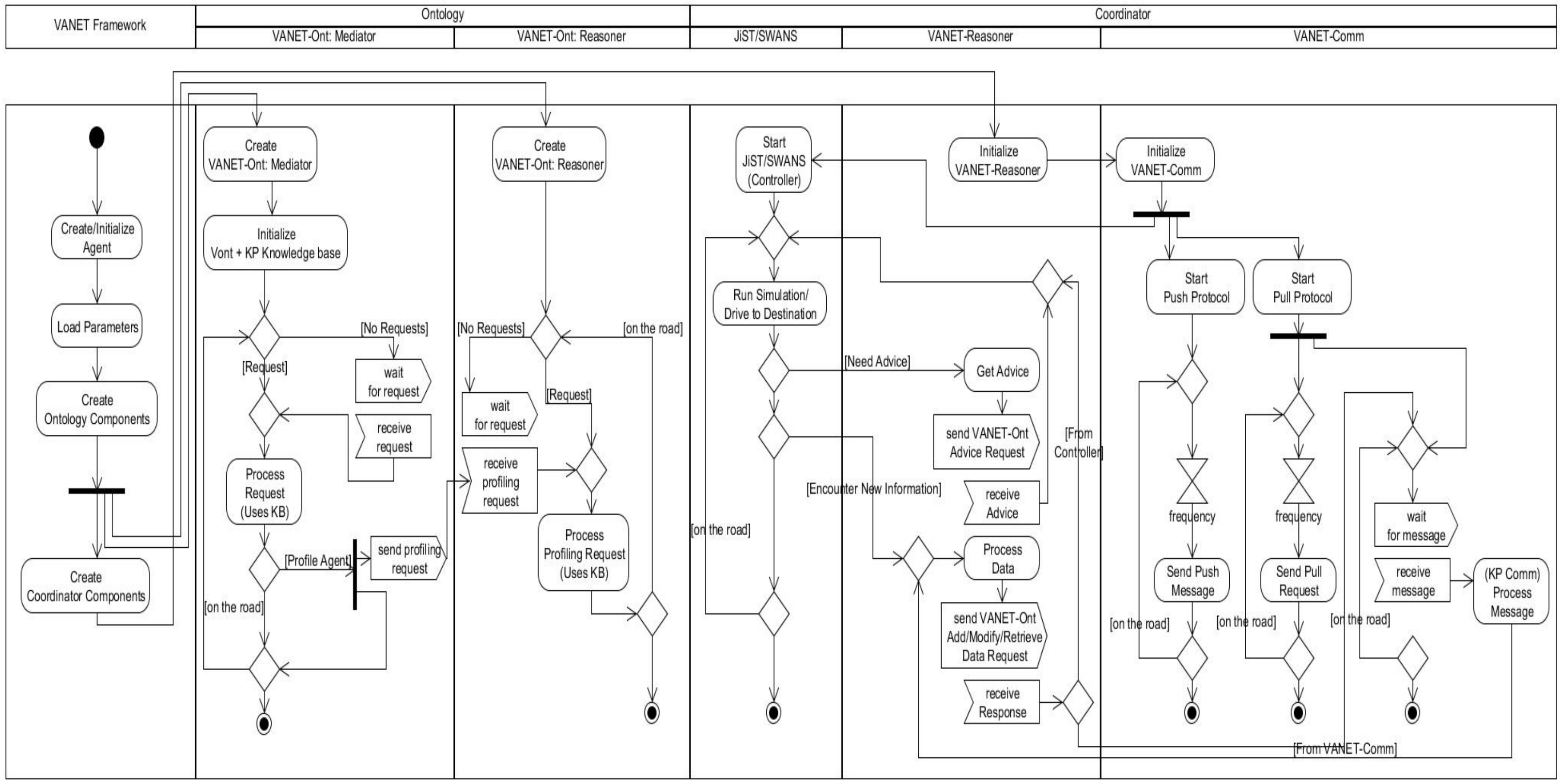

Figure 4.10: High Level Activity Diagram of the Model Framework.

Chapter 5

Validation

This chapter describes the implementation of the framework model proposed in this research and presents simulation results which demonstrate its effectiveness in the VANET domain. The implementation was designed to simulate a number of VANET-enabled cars that can efficiently route themselves to randomly generated destinations, utilizing congestion information gathered by communicating with other cars, while modeling the trust of other agents and their congestion advice. The implementation includes all aspects of the proposed framework. Experience, Role, and Majority based trust are implemented and tested, including the use of time and location closeness and distinguishing direct from indirect reports. The following sections describe the implementation, including third party software, simulation details, design rationale, and testing results.

5.1 Motivating Scenarios Revisited

This section revisits the motivating scenarios from Section 1.1 and describes how the use of our framework model could solve their issues. The motivating scenarios included four situations which were grouped into congestion issues and honesty issues.

The first congestion scenario described a typical and simple case where an inexperienced agent is traveling through a city, trying to reach a destination, but unnecessarily is delayed by heavy traffic. The second congestion scenario described an experienced agent who attempts to avoid congestion, by using a side route, but is delayed by unforseen circumstances. Both of these congestion issues could have easily been avoided through an implementation of our framework model into their GPS systems. Their GPS systems, through the utilization of ad-hoc information from other cars, would inform the driver of the path that would take the least time.

The scenarios with honesty issues implement a system similar to ours into their GPS, but it does not take into consideration agent honesty. The first honesty scenario described how a resident agent is fully trusting and is misled by a dishonest agent. The second honesty scenario described how the implemented congestion communication system fails due to a large amount of dishonest agents. Through the implementation of our system, with profiling (i.e. trust modeling) enabled, the resident agent of the first scenario could avoid the dishonest agent's advice if they previously profiled them or if other more trustworthy advice existed which contradicts the dishonest advice. The second scenario would most likely be avoided, because the dishonest agent from the previous scenario would most likely not succeed due to the incorporation of multiple reports. If it were the case that there were a high number of dishonest agents, this would not have a substantial impact on the resident agent. This is because our proposed framework model would ignore bad advice from dishonest agents, because of previous experience, and the few honest agents would quickly become trusted, and be used for gathering traffic information. Our simulations in the following sections will demonstrate these claims.

5.2 Implementation

The implementation is designed as a VANET extension to an existing real time traffic simulator. Each car within the simulation is representative of an individual agent that implements the proposed VANET framework of Figure 3.1. This means in a 100 car simulation there are 100 instances of the framework, including 100 knowledge bases and 100 VANET-Reasoners.

This section discusses various implementation specifics associated with parts of the framework model from Chapter 3 and 4 and how they are significant to the simulation's execution. The most important details described in this section is use of 3rd party software, the separation of the ontology from the simulator, the dynamic growth of knowledge bases, communication protocols used, range of communication messages, the operation of the trust modeling component, and where majority opinion is used in the simulator.

5.2.1 3rd Party Software

The implementation makes use of the following third party software, JiST/SWANS, vans, DUCKS, and Protege [1]. JiST stands for Java in Simulation Time; it is a high-performance discrete event simulation engine that runs over a standard Java Virtual Machine (JVM). SWANS stands for Scalable Ad-hoc Network Simulator; it is built on top of the JiST platform and serves as a host of network simulation tools. Vans is a project comprising the geographic routing and the integrated Street Random Waypoint model (STRAW). STRAW utilizes an A* search algorithm to calculate shortest path to a destination. DUCKS is a simulation execution framework, which

[1]Minhas et al. also used JiST/SWANS, vans, and DUCKS in a modest simulation that tracked average speed of vehicles using trust modeling in order to validate its worth. However, no traffic reports were actually exchanged so that only parts of their trust modeling were explored and their representation of vehicle path choices was a general estimate.

allows for a Simulation Parameters file to be provided to define the simulation. Protege is a free, open source ontology editor and knowledge base framework.[2] Note that the total lines of code (JiST/SWANS software plus our own code) was 85,000, where approximately 35,000 was implementation code.

5.2.2 Ontology Details

The Protege API allows for easy editing and visualization of Ontologies. Due to this and its base language, Protege was chosen as the primary means of ontology interaction within the VANET-Ont component, as indicated in Figure 3.1. The Protege API is based in Java, which was desired and convenient due to the JiST/SWANS simulator being Java based.

The VANET-Ont knowledge base component of the implementation runs on a separate process from the simulation component. This is indicated in Figure 3.1 by the component diagram separating everything into either the Ontology or Coordinator node. The component diagram also shows that communication and data transfer between the nodes is done using sockets. The need for separating these two components is the result of a solution to an observed fundamental issue in the JiST software, where it is impossible to execute Protege API commands from within the simulation. The necessitated separation also has the benefits of enforced modularity and utilization of multiple threads.

The VANET-Ont component has three main elements, a **server** element which receives, interprets, and processes messages from the simulation's VANET-Reasoner component, a **mediator** element which calls the actual Protege API methods, and a **reasoner** element which runs on its own thread and is responsible for profiling and executing trust modifications. During the execution of requests, the simulation component simply waits for a return message, which indicates the completion of the request. This waiting insures a coherent parallel nature to the two components. The only exception to this is the trust reasoner which operates outside of this request and wait procedure. This was intended to allow the simulator to not be delayed by trust reasoning which could be done in parallel.

The cars/agents created in the simulation each create their own Knowledge Base, which is initially empty but all follow the Vont + KP ontology described in Chapter 4.1. During the simulation, each car autonomously and dynamically populate their database as they encounter road sections, other cars, messages, etc. This dynamic behavior serves to be both space efficient and to accurately simulate a real world scenario.

The implementation's use of the KP ontology is limited to the *kp:assertedProp* and *kp:derivedProp* classes. Asserted Propositions *kp:AssertedProp* describe information that was directly observed and Derived Propositions *kp:DerivedProp* describe information that was indirectly observed or derived from other propositions. The

[2]We choose to employ the OWL-Lite dialect here.

simplistic use of these two classes is effective enough for distinguishing direct and indirect reports. Future work in Section 6.2 describes the possibility of use a larger range of KP propositions to describe more complicated scenarios.

5.2.3 Communication Details

The implementation includes both the pull and push protocol. Differences in performance between the two protocols will be shown in Section 5.6. The current implementation includes all messages shown in Table 4.1. Congestion is represented as the number of cars on the road[3]; it is also possible to use average speed to represent congestion. The sending of all messages in the implementation uses the built in communication system of the JiST/SWANS simulator.

5.2.3.1 Priority Roads

As explained in Section 4.2.1.3, this thesis encountered an issue with the pull based protocol from the original VANET research in terms of requesting specific information. The pull based protocol and its associated message requests congestion information about a specific road. The challenge is specifying what road to request information about. This thesis addresses this issue by introducing the concept of priority roads, which are roads that have been deemed important for an agent during the execution of the simulation.

The concept of a priority road facilitates messaging (and hence the updating of user models). Priority roads are implemented by storing a list of important roads in a priority list, which are retrieved by the pull based protocol when the messaging frequency determines it is time to send an information request for a priority road. Roads are added to this list when the pathing algorithm looks for congestion information about a road and either none exists or the data is very old. Roads are removed from the list when a set number of retrievals has been done. The priority list is constantly maintained and stores a road and the number of times advice and information has been retrieved for the road since it was added to the list.

5.2.4 Trust Modeling Details

5.2.4.1 VANET-Ont Reasoner

The VANET-Ont reasoner operates autonomously on a separate thread from all other implementation components. The only interaction with other components comes from other components issuing tasks to the reasoner's queues. These tasks are either agents of interest or recently updated local road segments. These are subsequently processed and result in an update to one or more agents' trust variable or no action at all. Agents of interest are agents that have demonstrated either a highly accurate or inaccurate

[3]This is determined directly through the simulation software.

report during a congestion evaluation. Recently updated local road segments are road segments, and their congestion value, which have been reported directly from the resident agent and should be regarded as absolute truth. Due to the congestion being absolute truth, the reasoner can systematically inspect the knowledge base and evaluate any propositions that were reported within a specified time of this local report.

5.2.4.2 Pathing

Agents within the JiST/SWANS simulation software utilize an A* search algorithm that determines the most effective path for a car to take to its destination.

The A* search algorithm is the driving force behind when an agent is *in need of advice*. The algorithm is called either when a new destination is set for an agent, and the agent has to find out how to most effectively reach the destination, or if an agent's path is reassessed during their journey, so that the algorithm can incorporate more recently received traffic information.

The A* algorithm used within our framework operates as follows:

1. It is provided with the agent's current location and destination.
2. It incrementally assesses potential roads, from the current location to the destination, according to a cost.
    (a) The potential road's cost is calculated as its length plus congestion (triggers *in need of advice*).
3. It returns a list of roads which forms a path to the destination that has the least cost (which theoretically takes the shortest amount of time, according to current traffic information).

The algorithm attributes a cost to every road segment. The JiST/SWANS initially calculated this cost as the length of the road segment. In our implementation, cost is calculated as the length of the road segment and its congestion. Equation 5.1 shows the specific calculation of a road's cost. $RoadCong$ is the congestion of the road, which is multiplied by a simulation specific weight $CongWeight$. $CongWeight$ is specific to the implementation and the value we use is shown in Table 5.3. The retrieval of a road's congestion signifies an agent being *in need of advice* from Algorithm 7. Congestion data is stored both in the agent's knowledge base and in a hash table, within the VANET-Reasoner component, for quick reference.

$$RoadCost = RoadLength + CongWeight * RoadCong; \tag{5.1}$$

To facilitate efficient use of congestion information, and to increase the speed of the A* search algorithm, the implementation post-processes traffic information to form majority opinions so that the information can be immediately retrieved during algorithm execution. This means that majority opinions are calculated every time new information is retrieved, which is then stored in a local hash table for constant time (O(1)) retrieval by the A* algorithm.

5.2.4.3 Information Sparsity

The simulation software initially creates a set number of agents in the environment, which continuously seek new destinations. During the simulation, we decided to have agents request and push messages every 6-15 seconds; with 100 cars in total, experience with every other car would be gained quickly. Due to the reuse of agents/cars within the simulation, it was not possible to truly simulate information sparsity. Information sparsity is the case where an agent cannot effectively use experience based trust due to the lack of agents which they have had previous experience with. In order to simulate environments with low experience-based trust, we introduce a variable called sparsity.

Information sparsity is an important testing variable which reflects the scarcity of agents in the real world that you may have had previous contact with (For example, 80% sparsity resembles having a lack of previous experience with 80% of the agents). This in effect greatly influences the effectiveness of experience based trust. Role based trust is utilized to cope with this sparsity. Information sparsity is implemented in the simulation framework as the percent to which all trust updates are ignored, therefore hindering experience based trust.

5.3 Experimental Setup

This section describes the experimental setup and the simulation tests performed to compare and contrast the effectiveness of the framework's implementation against a variety of scenarios, border cases, and variations of core parameters (like agent honesty and percent sparsity). The different simulations and groups of simulations serve to either demonstrate the effectiveness or robustness of the model, or the value of separate parts of the model. The different simulation types are shown in Table 5.1.

Our first set of experiments incorporated experience-based trust and majority-based trust, alone. These were the central elements of the VANET framework (and we anticipated exploring the value of other trust modeling facets in later simulations). We call this type of simulation *Basic*. This was also implemented using the push protocol for communication, for simplification.[4] A second simulation type, with all trust modeling facets integrated, we refer to as *Full*. This implements the combined push-pull communication protocol outlined in Algorithm 4.[5]

The scenario descriptions are fairly straightforward except for the last case where parameters are also listed in the scenario's name. *Basic* and *Full* are typically paired

[4]Note that this means that priority roads were not modeled or used in travel decision making.

[5]Pull messages were required in order to support the direct/indirect trust facet: agents need to be able to comment on whether information on a specific road is only known indirectly.

Table 5.1: Simulation Types

Name	**Description**	**Type**
No Traffic	Simulation without our framework or any incorporation of traffic data.	Worst case scenario
Omni	Simulation without our framework but incorporations traffic data by querying the road through the JiST/SWANS simulator.	Best case scenario
Basic	Simulation with just Majority and Experience based trust.	Basic scenario
Full	Simulation with all multidimensional trust components.	Full utilization scenario
Full/Basic + (Parameter(s))	Full or Basic simulation with a modification on one or more parameters.	Special case scenario.

with an honesty degree and whether trust modeling is enabled. The simulation framework begins each simulation by processing these parameters and variables from a configuration file. Table 5.2 and 5.3 show the full list of parameters which can be set in the system. Many parameters are shown as a decimal number that represents a percent (Hon 0.5 is 50% honesty which means that 50% of the agents are honest). Not all parameters are indicated in the simulation's name. If a parameter is not indicated then it is either set to its default value for the scenario type or indicated in the figure.

Testing includes three different metrics and three different groups of simulations. The three metrics are average speed, average number of paths, and average path time. The three different groups of simulations are core, model and framework.

The three distinct metrics are used to quantify the performance of the simulation. The first metric of average speed is the most basic, representing average speed of cars over the duration of the simulation. The second metric is average number of paths. A path is a collection of roads that form a route to an agent's destination. Due to the simulation using a pathing system, opposed to random movement, at the end of every agent's path it needs to recalculate and start a new path. The number of paths an agent completes during a simulation is calculated as this metric and more effectively measures the performance of the system than the first metric.[6] The last metric is average path time, which is simply the average time taken for a path to complete during a simulation. The average path time is of particular interest because the overall goal of this research is to allow an agent to reach its destination in the shortest amount of time. Due to average path time's importance, it is used to quantify

[6]A vehicle which completes fewer paths in total is one which is making poorer decisions about travel.

Table 5.2: Simulation Framework Variables

Parameter Name	**Description**	**Representation**	**Default Value**
Honest agents	Percent of honest agents.	Hon # (0.5 is 50% honesty)	0.5
Number of agents	Number of agents and cars simulated in the tests.	Agent # (100 is 100 agents)	100
Message interval	Interval between congestion request messages sent by the agents.	MsgI #-# (6-15 is 6-15 second message intervals)	6-15
Profiling	Use of profiling.	No P indicates no use of profiling (False)	True (Basic, Full)
Role	Use of role based trust.	Role # (0.2 is 20% agents are given a role above *Ordinary*)	0(Basic) 0.2(Full)
Time closeness	Use of time closeness factor.	Time	False(Basic) True(Full)
Location closeness	Use of location closeness factor.	Loc	False(Basic) True(Full)
Indirect messages	Use of indirect messages.	Indirect	False(Basic) True(Full)
Information sparsity	Percent of agent trust updates ignored to simulate data sparsity.	MThresh # (0.6 means 60% of trust updates are ignored)	0
Dishonest Lie Percent	Percent of the time a dishonest agent lies.	Lie # (0.8 is 80% of the time dishonest agents lie)	1

Table 5.3: Simulation Algorithm Variables

Parameter Name	**Description**	**Representation**	**Default Value**
Majority N	Number of agents used in a majority opinion.	MajN # (10 is 10 agents used)	10
Honest trust increase α	Standard increment to an agent's trust resulting from an honesty evaluation, with a maximum value of 1.0.	α # (0.1 is 10% trust increase)	0.1
Dishonest trust decrease β	Standard decrement to an agent's trust resulting from an honesty evaluation, with a minimum value of 0.0.	β # (0.2 is 20% trust decrease)	0.2
Advice trust threshold	Threshold where only agents with a trust value above this percent may be considered for advice.	AThresh # (0.41 is 41% trust threshold)	0.41
Majority confidence threshold	Threshold which the majority opinion must be above in order to be considered.	MThresh # (0.51 is 51% majority threshold)	0.51
Role penalization	Standard factor for increasing confidence depending on agent role.	RPenal #	8
Time penalization	Standard comparison factor for time closeness.	TPenal #	90
Location penalization	Standard comparison factor for location closeness.	LPenal #	200
Indirect penalization	Standard factor for modifying confidence if the advice is indirect.	InPenal #	1
Congestion Weight	Standard factor for weighting the congestion value when calculating a road's A* cost.	CongWeight #	20

most of the simulations.

The three groups of simulations organize simulations into similar topics and importance. Core simulations in Section 5.4 demonstrate the effectiveness and robustness of our simulation framework against core scenarios (No Traffic (Worst case scenario), Omni (Best case scenario)). Model simulations in Section 5.5 demonstrate the effectiveness and robustness of our simulation framework against scenarios that include the more complicated model parameters (*Role #, Time, Loc, Indirect*). A more detailed comparison of varying honesty values is also included. Framework simulations in Section 5.6 demonstrate the effectiveness and robustness of our simulation framework against scenarios that vary implementation parameters of the framework (Number of agents, Message Interval). All simulation tests results are averaged over 5 runs.

5.4 Core Simulations

This section presents results from simulations that contrast the effectiveness of our simulation framework against core scenarios, using all metrics. Core scenarios are the *No Traffic*, *Omni*, and *Basic* scenarios, as well as scenarios that do not use profiling (*No P*)(i.e. agents do not model the trustworthiness of the traffic reports received and assume that they are truthful). These simulations validate the fundamental effectiveness of using traffic data in path planning and the incorporation of profiling presented in framework model of this thesis.

For Figure 5.1, all of the simulations that used profiling or the omni setup averaged close to the same speed at the end of the 10000 second simulation. The other simulations (No P or No Traffic) produced a predictably declining performance as the honesty percentage approached the worst case scenario[7]. The curves in the scenarios are representative of the simulations approaching a steady state.

For Figure 5.2, we use a metric of average number of paths. Recall that higher average number of paths indicates better success with travel decisions and thus higher performance. All of the simulations that used profiling or the omni setup averaged close to the same number of paths at the end of the 10000 second simulation. The other simulations produced a predictably declining performance as the honesty percentage approached the worst case scenario.

Figure 5.3 examines a third metric, average path time (appropriate due to the ultimate goal of reducing the travel time of users). This figure compares the worst case scenario against the best case scenario and various simulations which use our VANET system with the *Basic* simulation settings, at different degrees of honesty. Greater average path time in the figure indicates lower performance. As seen in the figure, the simulations that used our trust modeling framework (except *Basic, Hon 0.1*) or the omnipresent setup averaged close to the same path time at the end of the 10000 second simulation. The other simulations produced a predictably declining

[7] *Omni* appears to be outperformed at times. This occurs when agents take paths with little traffic that are in fact long detours (better average speed, but poorer choices due to time of travel).

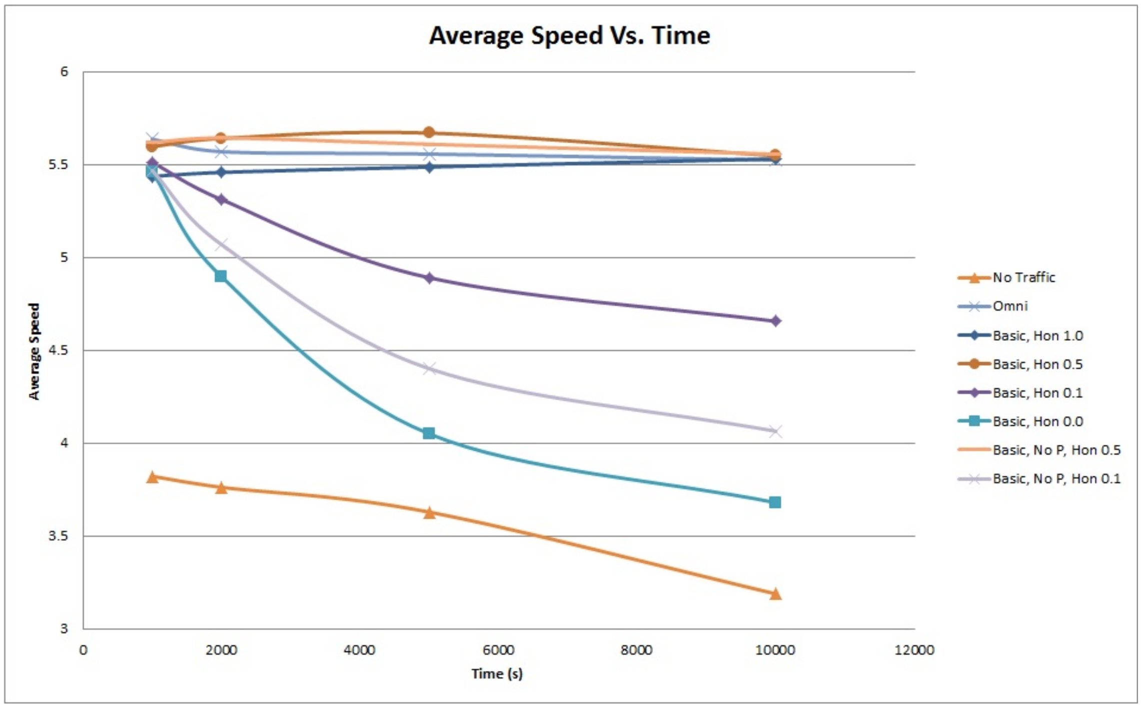

Figure 5.1: Average Speed.

performance as the honesty percentage approached the worst case scenario. The *Basic, Hon 0.1* simulation did much worse than the other *Basic* simulation most likely due to the extreme lack of trustworthy agents, but it still performed significantly better than the *Basic, No P, Hon 0.1* simulation. The VANET trust modeling simulations show approximately a 35% decrease in average path time over the worst case scenario. The curves in the scenarios are representative of the simulations approaching a steady state. Another observed trend is the tendency for the profiling-enabled simulations to reach a steady state faster than the other simulations.

Figure 5.4, 5.5, and 5.6 are screen shots taken directly from the simulator. These demonstrate the differences in congestion the simulations have at 10,000 seconds. Each of the numbered squares represents an agent and its respective car. The lines represent roads. A map of a real city is provided to the simulator. The road grid shown is a small section of the city which is created using coordinates provided in the Simulation Parameters file. The colors are representative of speed. White means no speed. Green represents increasing speed. Yellow represents no change in speed. Red represents decreasing speed.

Figure 5.4 demonstrates a simulation that uses the key elements of our framework implementation with trust modeling (Basic, Hon 0.5). Cars in the screen shot have a relatively equal distribution throughout the city. Figure 5.5 demonstrates the same simulation but with no trust modeling (Basic, No P, Hon 0.5). Cars shown are less

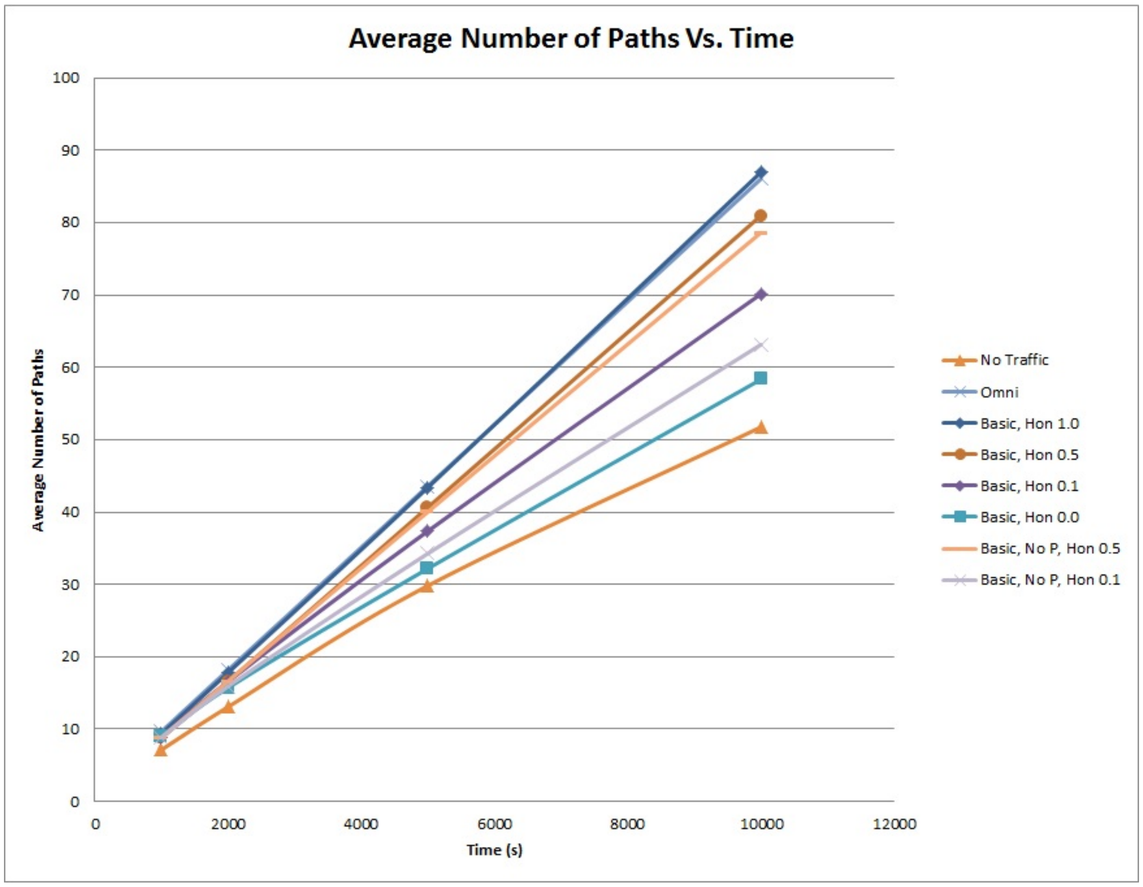

Figure 5.2: Average Paths.

distributed throughout the city and congestion can be seen in the middle roads and intersections. This is likely due to these intersections and roads being most often considered in shortest path calculations. The last screen shot, Figure 5.6, demonstrates a simulation that does not use our implementation or any traffic metrics in path calculations (No Traffic). The congestion in the middle roads and intersections is substantially greater, logically resulting in slower car speeds and longer travel times. The presented figures visually demonstrates the effectiveness of our proposed model framework on the traffic congestion of a city.

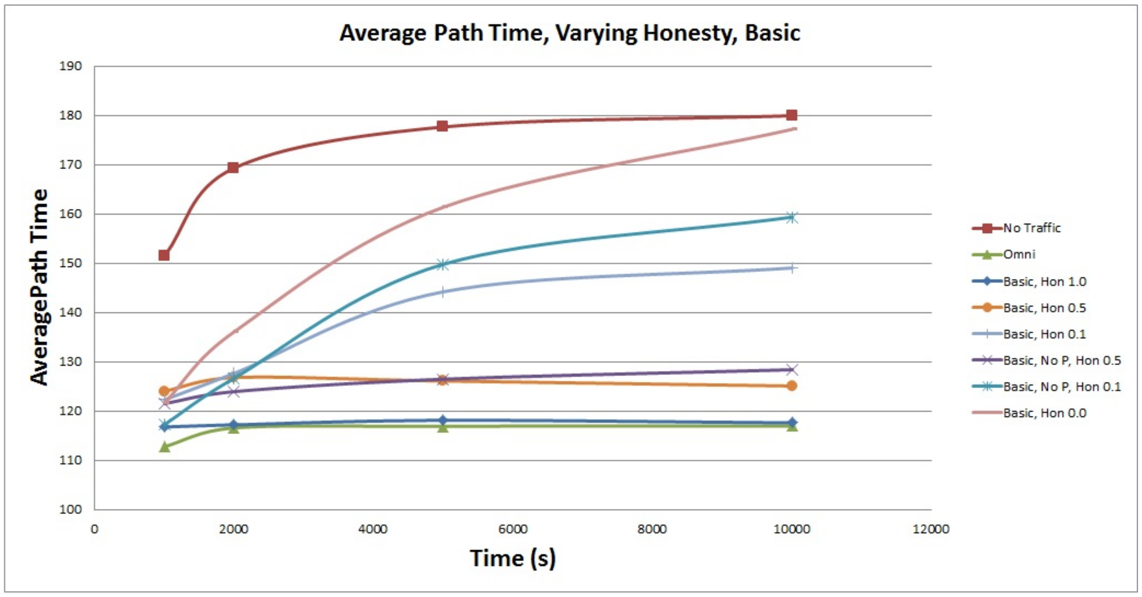

Figure 5.3: Avg Path Time comparison of our *Basic* model vs. best case, worst case, and *No P* scenarios

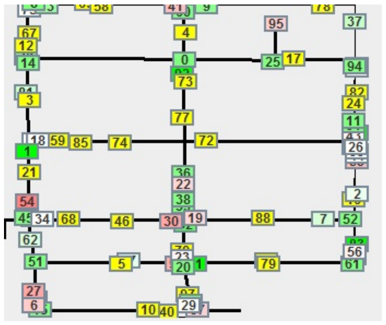

Figure 5.4: Profiling scenario with 50% honesty (*Basic, Hon 0.5*).

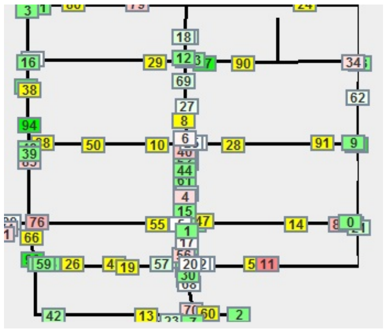

Figure 5.5: Non-Profiling scenario with 50% honesty (*Basic, No P, Hon 0.5*).

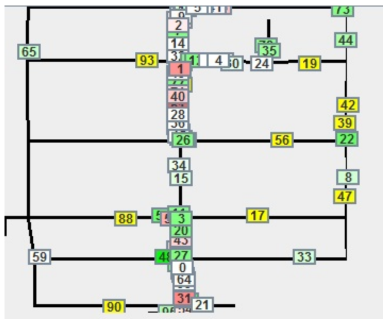
Figure 5.6: Worst case scenario that does not use traffic information (*No Traffic*).

5.5 Model Simulations

This section presents results from simulations that contrast the effectiveness of our simulation framework against scenarios that include various dimensions of the presented trust model with various parameter settings. The average path time metric is primarily used within the simulations of this section. Included scenarios compare the *Full* simulation type against *No Traffic*, *Omni*, *Basic*, and *No P* scenarios. The incremental inclusion of the model's multidimensional trust components are also simulated and compared. These simulations validate the fundamental effectiveness of the framework model's more complicated components, such as Role, Time and Location closeness, and the inclusion of indirect messages.

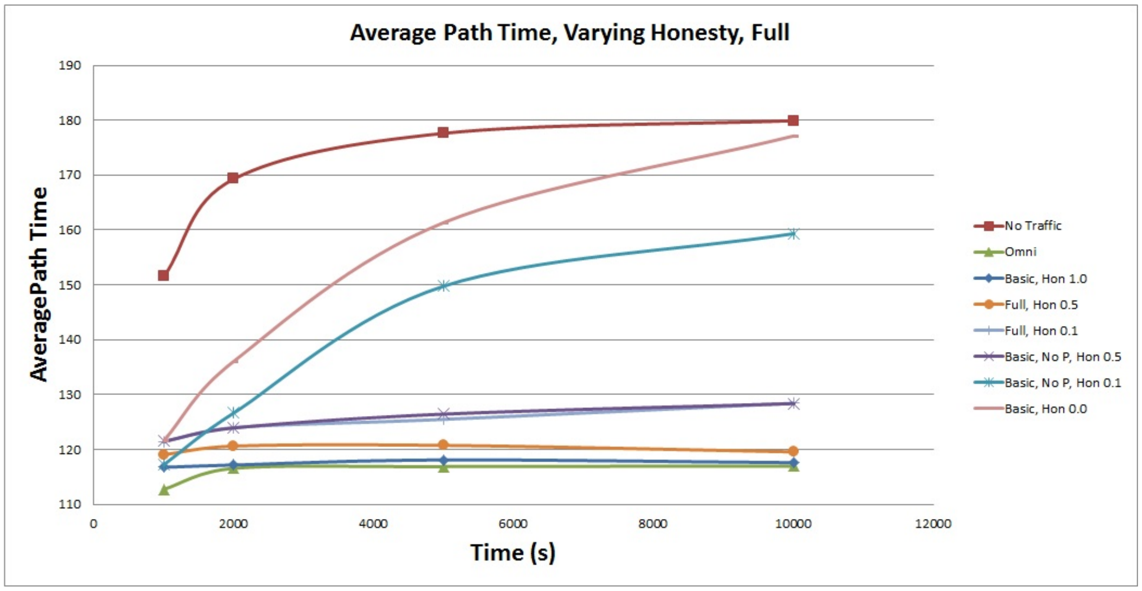

Figure 5.7: Avg Path Time comparison of our *Full* model vs. best case, worst case, and *No P* scenarios

Figure 5.7 compares the worst case scenario against the best case scenario and various simulations which use our VANET system with the *Full* (all trust multi-dimensional trust components activated) simulation settings, at different degrees of honesty. As seen in the figure, all of the simulations that used our trust modeling framework (Full) or the omnipresent setup averaged close to the same path time at the end of the 10000 second simulation. The other simulations produced a predictably declining performance as the honesty percentage approached the worst case scenario. In contrast with Figure 5.3, *Full* simulations performed significantly better compared to the *Basic* simulations of similar honesty.

Figure 5.8 compares the average path time, at 10,000 seconds, of the *No Traffic*, *Omni*, *Basic*, *Basic, No P*, and *Full* scenarios, across a range of honesty values. *No Traffic* and *Omni* are shown as straight lines because they do not use honesty

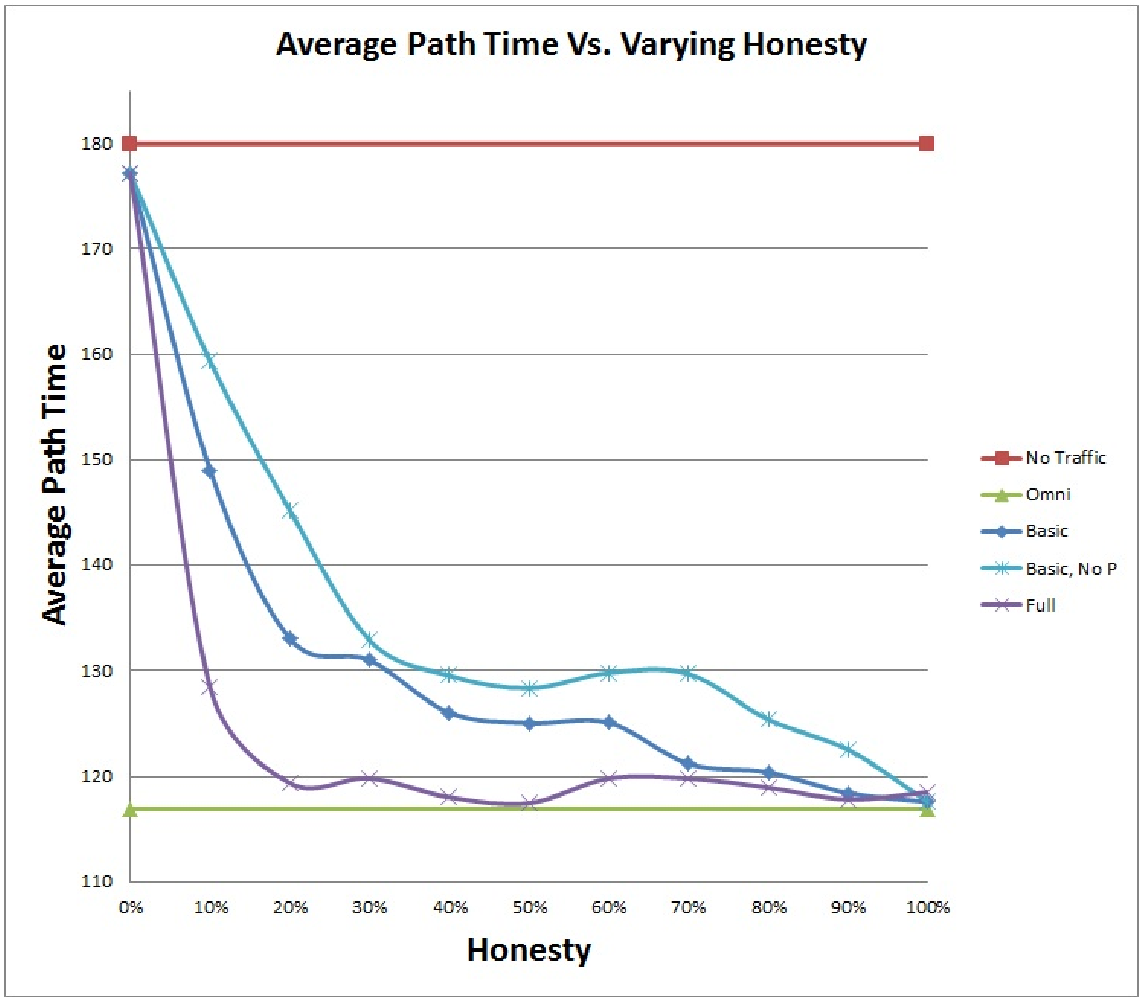

Figure 5.8: Avg Path Time comparison of simulation types over varying degrees of honesty at 10,000 seconds

values, but are useful as comparisons. The figure clearly shows the effectiveness of our framework across the range of honesty values. The *Basic* scenario consistently performs better than the *Basic, No P* scenario. The *Full* scenario also consistently performs better than the *Basic* scenario. All of the framework enabled simulations have a similar average path time at 0% honesty because they have no useful traffic data (and at 100% honesty because there are no untrustworthy agents to deflect through profiling). Figure 5.8 clearly demonstrates the impact dishonest agents can have on simulations (*Basic, No P*) and the effectiveness our proposed model framework scenarios (*Basic* and *Full*) can have on countering the influence of dishonest agents.

Figure 5.9 demonstrates the increased effectiveness of each of the multidimensional trust components described in Sections 4.3. The incremental components demonstrated are the base system (experience and majority based trust), then role based

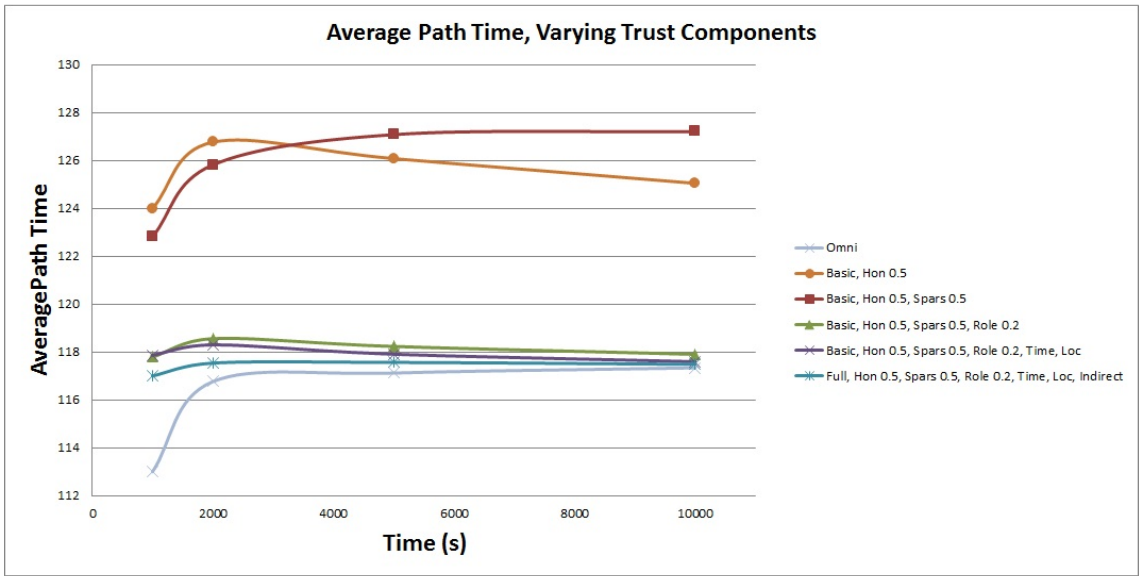

Figure 5.9: Avg Path Time comparison, multidimensional trust component variations

trust (Role 0.2), time and location closeness (Time, Loc), and indirect advice (Indirect). These simulations also simulate honesty at 50%, data sparsity at 50%, and additionally compare them to the best case scenario.[8] As seen in the figure, the incremental addition of trust components demonstrated predictable and substantial increases in performance. The simulation with sparsity enabled showed a predicably worse performance than its counterpart. This reflects the fact that when one has little experience-based trust, one makes poorer decisions. The simulation with role-based trust enabled shows a dramatic increase in performance, which demonstrates the impact roles have in situations with data sparsity. The best case scenario and the simulations with the higher number of trust components averaged close to the same path time at the end of the 10000 second simulation. The curves in the scenarios are representative of the simulations approaching a steady state. Another observed trend is the tendency for the component-enabled simulations to have a steadier state than the other simulations.

Finally, Figure 5.10 explores variations in parameter values to demonstrate the robustness of our proposed framework. We note that, even if there are very few roles assumed or if dishonest agents lie inconsistently, our framework is able to adapt and yield excellent performance. When using all dimensions (Full), being more challenged with experienced-based trust (higher sparsity) degrades performance slightly as does having less role-based trust to rely on.

[8]The worst case (i.e. No Traffic) is not present so that a finer granularity of the presented simulations can be shown.

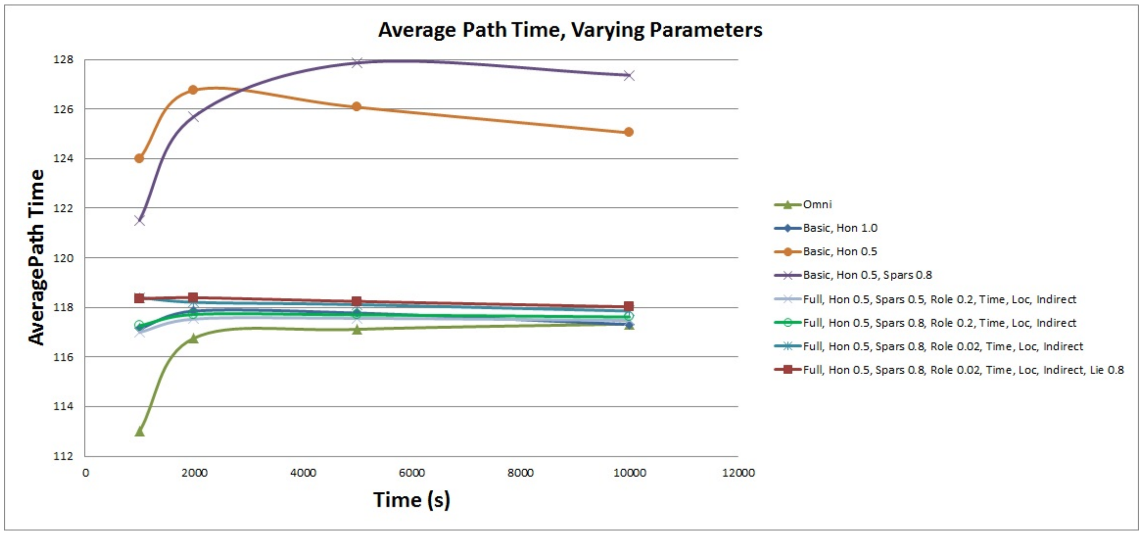

Figure 5.10: Avg Path Time comparison, multidimensional parameter variations

5.6 Framework Simulations

This section presents results from simulations that evaluate the robustness of our simulation framework through simulations that modify simulator-specific variables, such as the number of agents and messaging frequency. The average path time metric is the one primarily used within the simulations of this section. Included scenarios compare the *Full* and *Basic* simulation types against *No Traffic*, *Omni*, and *No P* scenarios that feature the modified parameter where appropriate.

Figure 5.11 compares the average path time, at 10,000 seconds, of the *No Traffic*, *Omni*, *Basic*, *Basic, No P*, and *Full* scenarios, across a range of values for the number of agents in the environment. The figure clearly shows the robustness of our framework across the span of agent values. The simulations around 50 agents have approximately the same path time because with such a small number of cars there is no real need for using traffic information in path planning. When increasing the number of agents, the *Basic* scenario consistently performs better than the *Basic, No P* scenario. The *Full* scenario also consistently performs better than the *Basic* scenario, when there are more than 50 agents. Figure 5.11 clearly demonstrates the robustness and scalability of our proposed model framework and implementation across a range of values for the number of agents in the environment.

Figure 5.12 compares the average path time, at 10,000 seconds, of the *No Traffic*, *Omni*, *Basic*, *Basic, No P*, and *Full* scenarios, across various messaging intervals (where x-y means that messages are sent every x to y seconds)[9]. The purpose of the figure is to demonstrate the robustness of the simulations when there are more or fewer messages. *No Traffic* and *Omni* are shown as straight lines because they do

[9]Messages are sent according to intervals to avoid all agents sending messages at the same time.

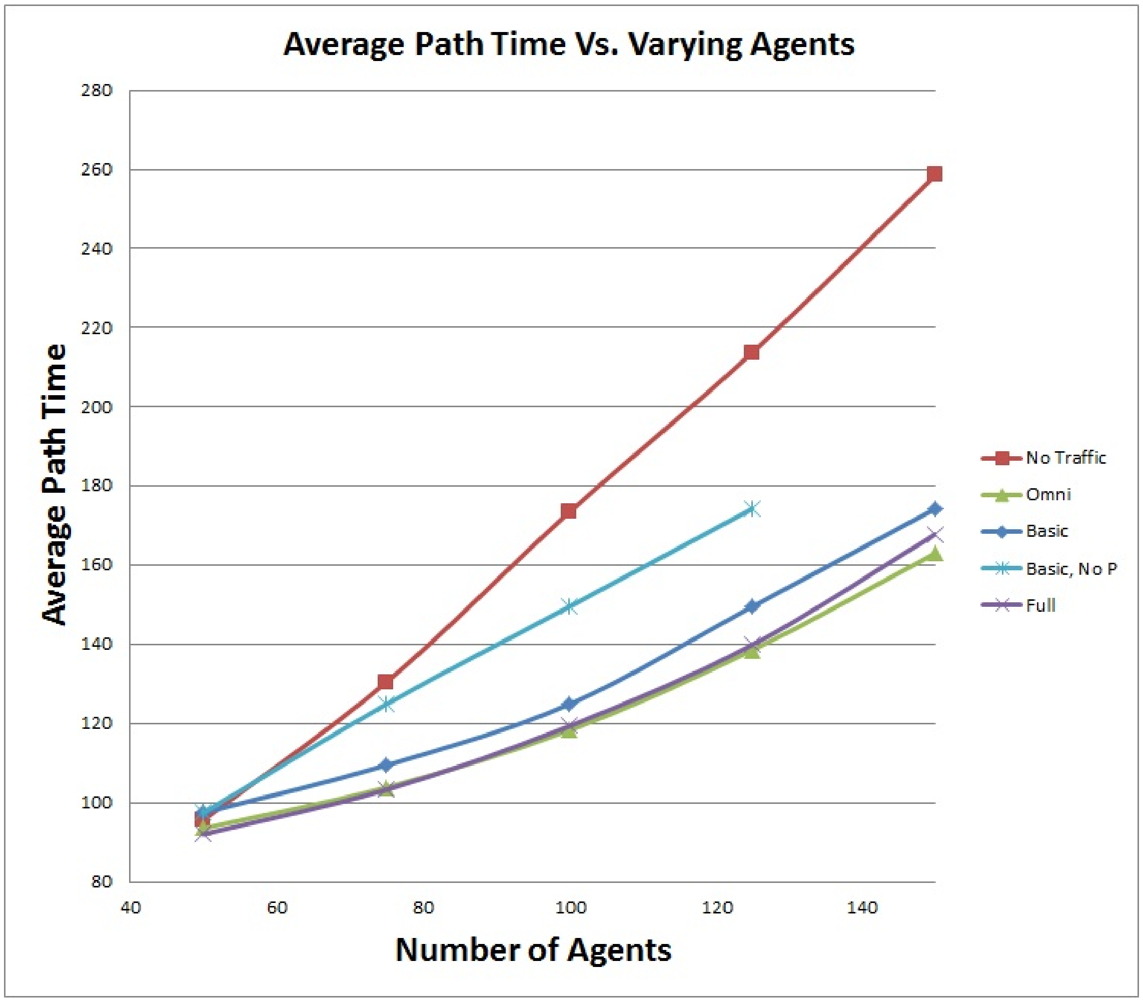

Figure 5.11: Avg Path Time comparison, varying number of agents

not use communication protocols, but are useful as comparisons. The figure clearly shows the robustness of our framework, especially the *Full* scenario, across various messaging intervals. The *Basic* scenario consistently performs better than the *Basic, No P* scenario until the message interval increases to (12-30 seconds) at which point the two lines are comparable. (This is because *Basic* is no longer receiving information at a sufficient frequency.). The *Full* scenario consistently performs better than the *Basic* scenario, with a more gradual decrease in performance as the message interval increases.[10]

Figure 5.13 compares the average path time, at 10,000 seconds, of the *No Traffic*, *Omni*, *Basic*, *Basic, No P*, and *Full* scenarios, with various communication protocols

[10]This more gradual decrease is likely due in part to the pull protocol requesting information on roads with more immediate priority and use, generating information on roads that will be used in decision making.

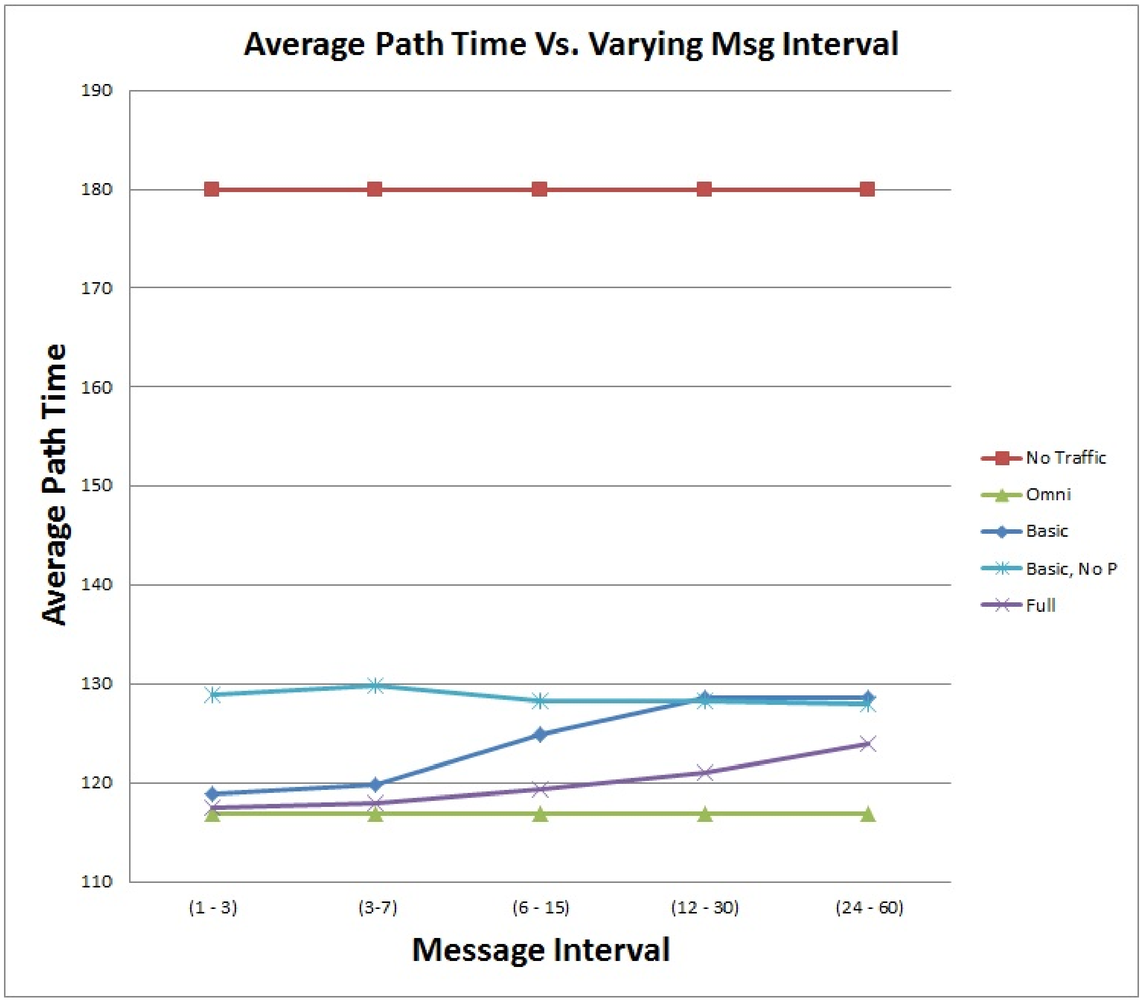

Figure 5.12: Avg Path Time comparison, varying message interval

enabled. *No Traffic* and *Omni* are listed under *No Msgs* because they do not use communication protocols, but are useful as a comparison. This figure is important for backing up our claim in Section 4.2.1 that replacing the pull protocol, for requesting agent location and congestion data, with the push protocol, which more simply sends out the resident agent's location and congestion data, does not impact performance. Our design rationale for this was to reduce the number of messages sent between agents.

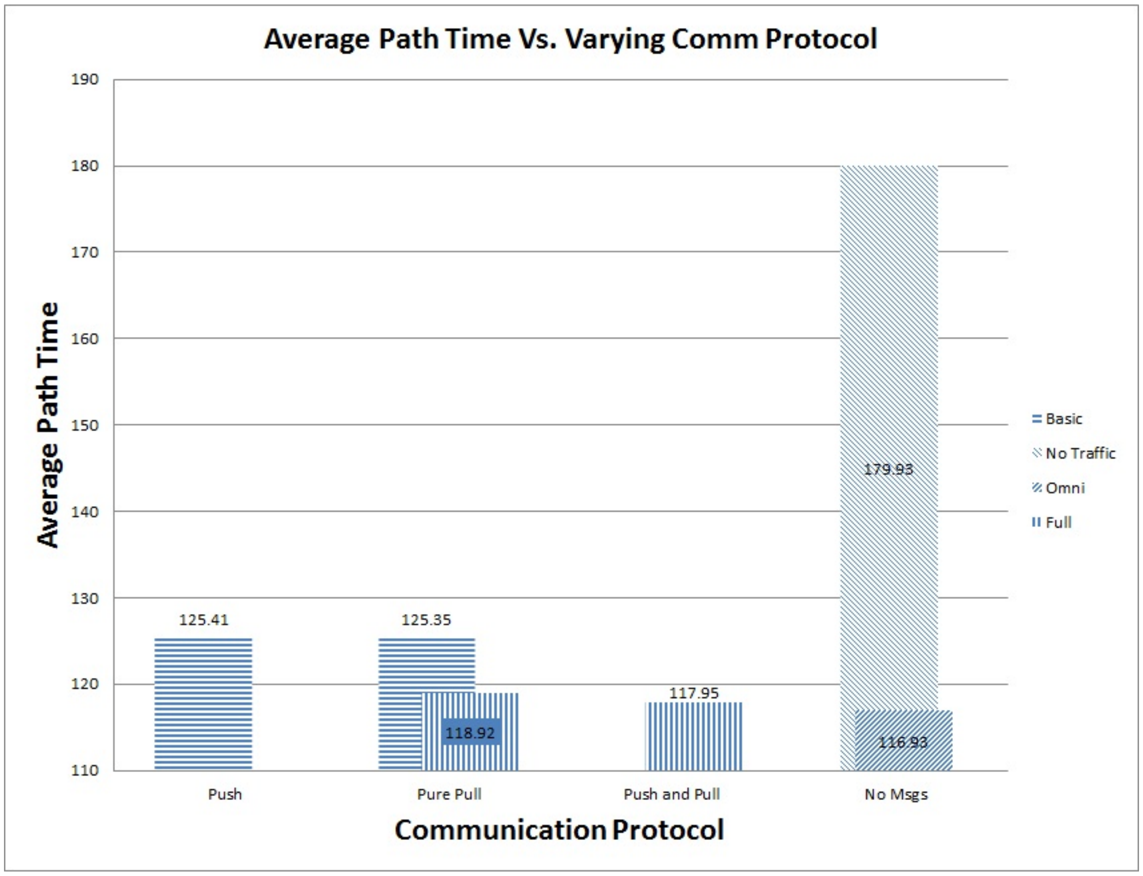

Figure 5.13: Avg Path Time comparison, varying communication protocols

Chapter 6

Discussion and Conclusion

6.1 Our Journey to the Framework and Results

One of the questions we set out to answer was the following: how can a proposed trust model used by intelligent agents representing drivers, be properly validated as valuable for producing effective vehicular travel decisions.

The VANET work of Minhas et al. had included preliminary validations in a simulated traffic environment. But the model of the intelligent reasoner for these simulations was very modest: allowing yes/no decisions at what time and finessing the proper simulation of exchanging traffic reports.

This challenge led us to the formulation of an extensive simulation testbed. This in turn required us to specify with far greater clarity the communication that was intended to take place between agents that would form the backdrop for the trust modeling and the subsequent travel decision making. Addressing this problem then obligated us to specify not only the desired content of the messaging but some appropriate underlying knowledge representation. Because trust modeling was an important element of that representation, we ultimately decided to introduce a representation framework that would not only store the information needed for decision making but would also facilitate reasoning about trustworthiness. For this we ended up repurposing the concept of knowledge provenance, as described in detail in Chapter 3 and 4.

Resolving the appropriate content for messaging was only one central design decision. The other was to specify with sufficient clarity the messaging that would occur, the conditions under which information would be exchanged. To resolve this problem, we developed a design that could potentially support a variety of possible messaging protocols, both push-based and pull-based. In addition, we carefully clarified as well the relationship between when messaging occurs, when reasoning about path planning occurs and when updates to user models occur (the central elements of the overall automated reasoning of the intelligent agents in our vehicular environment). All of this detail had been previously left unspecified in the models of Minhas et al.

With some decisions about message content and message delivery in hand, we turned to the underlying multi-faceted trust model of Minhas et al. to see how it might be validated, in full. It seemed clear that the binary traffic reports originally conceived by the authors could be expanded to include more useful communication, as well. To this end, we developed an overall design that supported a richer form of information exchange. In particular, when validating the model we sketched traffic reports being forwarded between agents which offered a congestion value, as more specific detail concerning the road in question. This then required us to develop more detailed algorithms for how to reason about the trustworthiness of agents, with this form of information exchange. Those algorithms are defined in Chapter 4. As we developed these algorithms, we also developed a useful extension to the trust modeling, affecting the algorithms for both experience-based and majority-based trust calculations, the concept of a suspicious agent.

Ultimately, it became clear that a very rich and detailed simulation testbed would be required in order to enable complex testing of any trust model for vehicular travel decisions. Towards this end we developed a complete testbed, sketched in Chapter 5. With this testbed we were then able to quantitatively compare the inherent value of different facets in the multi-dimensional trust model and to confirm more definitively the importance of including all the facets of trust modeling that this framework proposed. The way the testbed was designed, it was also possible to simulate a myriad of different parameter values and environmental conditions, in order to confirm the robustness of Minhas et al.'s model. A final element that we introduced in our construction of the simulation testbed was a novel modeling technique, the inclusion of what we refer to as priority roads. We are able to demonstrate that integrating this concept faciltiates the proper validation of trust models and serves to clarify in greater detail the ideal conditions under which messaging between vehicles should occur. This in turn then specifies the conditions under which user modeling updates should be occurring, as well. The concept of priority roads became an element of our overall proposal for reasoning about path planning, incorporating trust modeling, as summarized in the algorithms of Chapter 4.

6.1.1 Reflection on Related Work

6.1.1.1 Knowledge Bases

Our knowledge management system is managed in Protege, which is based on description logics [2], (where the metadata is terminology/ontology and the data is a set of assertions/a collection of facts). As such, we are proposing a combination of ontological and propositional elements which taken together enable all the central concepts of the environment to be represented (agents, roads, locations, etc.) and a representation of the current state of the environment to be modeled as well (in our case, a set of propositions which differs from agent to agent, reflecting what it knows about the current traffic conditions and what it has modeled about the other agents in the environment).

Our query language supports asking facts (and obtaining appropriate trustworthiness values of agents, calculated using the Vanet formulae) and the revision language supports updates and deletes to the propositional component. As such, our particular approach to knowledge base design is to gain the benefit of integrating both V-ont elements and KP elements, assuring a detailed representation of the world for each agent, but one where all agents are modeling objects with the same general specifications. Our solution therefore relies on the concept of knowledge provenance.

Ding et al. [7] also explores how to represent knowledge provenance. This is for the context of the Semantic Web, focused on evaluating the trustworthiness of discovered semantic associations. There are proposals for resolving inconsistency through the use of heuristics. Social reputation is assumed to be providing the required provenance information. The promotion of a Knowledge Provenance Infrastructure is also offered in [26]. This is once more described in terms of supporting answers for web applications and services.

For our application, the representation of knowledge provenance offered in [14] served us well, because it provided a straightforward method for connecting to our V-ont ontology. In addition, Huang uses the metaphor of agents to refer to information sources, so we were able to leverage this in order to adjust his proposals aimed at web sites for our context where actual intelligent agents are directing the cars. A detailed messaging framework that allows access to the required information was also an attractive feature.

6.1.1.2 Trust Modeling

While the Vanet reasoner is the central component in our system which enables intelligent agents in vehicles to make effective route planning decisions, it relies critically on effective management of the knowledge that is shared within the mobile environment. Included here is knowledge about the current road conditions and knowledge about the agents providing that information, effectively managed through appropriate messaging constructs. The design of the messaging that we employ in our framework, drawn from the model of [14], also offers a concise way to support all the processing needed to be incorporated into our simulation.

The multi-faceted trust model that forms the centrepiece of our proposed framework for enabling effective transportation decisions in a mobile vehicular network builds on well-established research in trust modeling by Minhas et al., originally explored for the application of electronic commerce.

Experience-based trust influences the learning performed in the trust model of [29] which confirms the value of representing trust as something that is difficult to build but easy to tear down, proposing that advice from only the most trustworthy peers be followed, but the model of Minhas et al. also makes valuable distinctions between direct and indirect reporting of trust. We are now able to show the contribution derived from modeling this additional facet through our simulation results.

Combining experience-based trust with some kind of majority opinion is suggested in the personalized trust model developed by Zhang [34] for the design of effective e-marketplaces, which empirically demonstrates the value of integrating both private and public trustworthiness of agents into its solution. Other researchers have proposed trust modeling frameworks that integrate majority opinion [15, 35]. We integrate here important consideration of time and location as well, in order to value more highly the reports from users closer to the destination. In so doing, we are able to weight the combination of majority and experience based considerations more appropriately. Others have employed a social network for trust modeling (e.g. [32] consider trust propagation in a network but this is less relevant in our sparsely populated environment) and others propose the use of stereotypical trust [4] (but in our domain a small set of roles can be used to reflect levels of trust.) [30] also describe trust as multi-faceted; this work is more focused on having trust calculated differently in distinct contexts. In addition, their selection of peer advice is based on similar preferences; for our domain, location of the user and the time of its report are more critical determinants.

Various researchers have also explored trust modeling and multiagent systems for traffic decision making. Lin et al.[16] have investigated the benefits achieved by self-interested agents in multiagent-based vehicular networks through simulations. They consider a scenario where agents can achieve road congestion information from other agents through gossiping. These self-interested agents pose a challenge either because they deliberately want to cause disorder or because they seek to maximize their own utility. The authors identify the need to establish trust in VANETS through distributed reputation mechanisms. One limitation about the work of [16] is that their simulations do not rely on any existing network, and thus ignore the complication of communication range, real-world traffic and driving rules such as speed limits, traffic signals, stop signs, etc. Our simulation testbed integrates the Scalable Ad hoc Network Simulator (SWANS) and the Street Random Waypoint model (STRAW) to simulate real-world scenarios. Another limitation about their work and the trust modeling work that will be introduced below is that they often evaluate the performance of their systems using only one single metric (e.g. average speed of cars or normalized journey length). In contrast, our testbed incorporates a variety of metrics to evaluate the effectiveness of trust models.

Various trust models have been proposed for traffic decision making. Here, we provide a brief summary of these models and contrast with our work.[1] Also focusing on the modeling of the trustworthiness of vehicular entities, the sociological trust model proposed in [10] shares some similarities with the multi-faceted trust management framework of Minhas et al.[20, 18]. It is based on the principle of trust and confidence tagging. Gerlach has identified various forms of trust including situational, dispositional and system. Additionally, he presents an architecture for securing vehicular communication. However, Gerlach does not provide a formalization of the

[1] A more complete discussion of trust management for VANETs can be found in the recent survey paper [33].

architecture for combining the different types of trust together.

In contrast to the traditional view of entity-level trust, [23] propose that data-oriented trust may be more appropriate in the domain of VANETs. Data-centric trust establishment deals with evaluating the trustworthiness of the data reported by other entities rather than trust of the entities themselves. Their decision logic ultimately outputs the level of trust that can be placed in evaluated evidences indicating whether the event related with the data has taken place or not. One of the shortcomings of their work is that trust relationships in entities can never be formed, only ephemeral trust in data is established, and because this is based on a per event basis, it needs to be established again and again for every event. This is acceptable as long as there is enough evidence either in support of or against a specific event, but in the case of data sparsity their model is significantly challenged. [11] also present a technique that aims to address the problem of detecting and correcting malicious data in VANETs. Their approach maintains a model of VANET at every entity which contains all the knowledge that a particular entity has about the VANET. Incoming information can then be evaluated against the entity's model of VANET. If all the data received agrees with the model with a high probability, then the entity accepts the validity of the data. However, this approach assumes that each vehicle has the global knowledge of the network and solely evaluates the validity of data, which may not be feasible in practice.

Several trust models have been proposed to not only model trustworthiness of entities but also use the modeling results to evaluate the reliability of data. For example, [8] have suggested building a distributed reputation model that exploits a notion called opinion piggybacking where each forwarding entity (of the message regarding an event) appends its own opinion about the trustworthiness of the data. They provide an algorithm that allows an entity to generate an opinion about the data based on aggregated opinions appended to the message. However, this approach repeatedly makes use of the opinions from different nodes. The nodes that provide opinions about a message earlier will have larger influence than the nodes which generated opinions later. [22] propose an approach in which the reputation of an entity is determined by data validation. In this approach, a few entities, which are named as anchor nodes here, are assumed to be pre-authenticated, and thus the data they provide are regarded as trustworthy. Data can be validated by either agreement among peers or direct communication with an anchor node. Malicious nodes can be identified if the data they present is invalidated by the validation algorithm. One problem about this scheme is that it does not make use of reputation of entities when determining the majority consensus. Overcoming some problems of the above two models, [5] propose a trust-based message propagation and evaluation framework in vehicular ad-hoc networks where entities share information regarding road condition or safety and others provide opinions about whether the information can be trusted. More specifically, the trust-based message propagation model collects and propagates entities' opinions in an efficient, secure and scalable way by dynamically controlling information dissemination. This model is demonstrated to promote network scalability and system effectiveness in information evaluation under the pervasive presence of false information.

Different from the above mentioned trust modeling work, our work focuses on clarifying the format of the messages that will be communicated among vehicular agents regarding travel decision marking. For this, we propose the detailed ontological representation of knowledge sharing among agents and trust information about agents. We also provide the specification of an algorithm for information exchange which clarifies when, where and with whom knowledge is shared towards effective decision making for the agents in the VANET environment.

6.2 Future Work

The future work for this project plans to expand both the proposed model and implementation to further describe and simulate various aspects of the VANET model. The following subsections present a variety of possibilities for future work with the presented framework model and simulation implementation.

6.2.1 Additional Simulation

Further troubleshooting of our chosen software for the simulation (or possible exploration of other valuable simulation frameworks) is one avenue for future research. In our current simulations, the choice of map and simulation parameters are important factors in the execution of simulation tests. The map used for the simulations in Section 5 was chosen due to its high stability and the significant improvement from using the best case setup over the worst case setup. Some maps produce a much more significant improvement but are prone to becoming jammed while others are stable but don't produce the traffic congestion desired. Future work might include sensitivity tests to these map variations. Simulation parameters also have a significant impact on the quality of the results.

6.2.2 Extended KP Propositions Use

The implementation presented in this thesis utilizes only the *kp:AssertedProp* and *kp:DerivedProp* classes of Figure 2.2. Future work could utilize a larger range of KP classes to describe more intricate and complicated scenarios. Fr example, the subclasses of *kp:CompoundProp* may be used to describe relations between propositions and their collective use in describing another proposition, such as a *kp:DerivedProp*.

6.2.3 Greedy Agents

The model presented in this thesis was shown to effectively counteract the influence of dishonest agents within the ad-hoc network. Future work could introduce and explore greedy agents, which may or may not be dishonest, who ignore the advice of framework model and decide to take the shortest route. The rationale the greedy agents would have in doing this would be that they expect that our presented framework model is working effectively and that the shortest route will not be as busy as it would in the case where the framework was not being used by all vehicles.

6.2.4 Disconnected Agents

The implementation presented in this thesis equips every car/agent with a GPS that uses our framework model. Future work could introduce disconnected cars/agents, which would make up various portions of the overall number of cars, that do not use our framework model and hence always take the shortest route. This would be different from the greedy agents because they would have no communication with framework-enabled cars/agents.

6.2.5 Destination Awareness

The framework model and implementation of this thesis primarily deals with communicating either current or older location and congestion information between agents. Future work could introduce messages or additional message information between agents where an agent will communicate its current destination or path. This would be useful information for predicting future congestion of roads. It also would allow an additional dynamic of misinformation, where the agent could communicate a false path or destination for whatever reason. This would have to be dealt with appropriately, possibly by evaluating the location information of later messages.

6.2.6 Honesty Queries

The framework model and implementation of this thesis primarily deals with communicating location and congestion information between agents. Future work could introduce messages that query an agent about their trust of another agent. These messages could help increase the rate at which agents are profiled and lead to fewer situations where dishonest agent information is trusted and used.

6.2.7 Recognizing Patterns of Agent Behavior

It would be useful to make use of historical data of agent behaviour. For instance, we may detect patterns of dishonesty, such as agents consistently being untruthful when congestion is high but truthful when congestion is low. We may also detect groups of agents that are dishonest when traveling together (perhaps reflective of collusion). This might suggest a need to a expand the modeling of agent roles to include stereotypes (i.e. class membership). We may also observe consistent patterns of travel and congestion within a city at certain times of day, which may be valuable to consider as part of path planning advice.

6.2.8 Data Expiration

For future work we will explore when to refresh the knowledge base, removing stale reports. This would increase search efficiency and decrease memory usage.

6.2.9 Real World Implementation

The framework model presented in this thesis was designed with a real world implementation in mind. This process would most likely take a good deal of effort and time, but an implementation as a phone GPS add-on could be possible. Implementing the framework in this manner would allow for easy integration into a city's driving population. The Android operating system and platform is a viable candidate for implementation due to its use of Java as a primary language and the capability to allow applications access to a wide range of phone systems (such as the GPS). Android phones also allow multi-threading. The phones could communicate with each other through minimal internet access.

6.3 Conclusion

This thesis has presented a framework consisting of a model and a simulation testbed to support effective mobile knowledge management for the application area of intelligent transportation. In particular, we enable effective path planning by intelligent agents in mobile vehicular networks through the integration of trust modeling: we enable the exchange of traffic reports between agents and then adjust decision making based on the modeling of the trustworthiness of the senders of the reports. All of this is achieved through a specific proposal for representing and employing the knowledge that forms the basis for the agents' travel decisions.

The VANET Ontology (V-Ont) was created to serve as a semantic repository for the VANET terms and relations. It also provides a semantically meaningful context which can be understood and used by external VANET software. At present, V-Ont is designed to be powerful and general enough to describe any feasible situation from the VANET environment, as well as to be used by third party software such as JiST/SWANS, vans, DUCKS, and Protege. In addition, it is important to note that the design of V-Ont would allow the ontology to be extended, if necessary, to handle any possible situation which arises from a more generalized multi-agent mobile vehicular ad-hoc network, and to work with various external VANET-related software applications other than the above-listed ones.

Our approach makes use of semantic knowledge to store complex information, including all trust information, via the design of an agent's knowledge base (KB). The current design facilitates three types of actions. The first action is **creation**, which assigns an initial, default trust value to all agents. The second action is **retrieval**, which reads the trust value of an agent, to evaluate the trustworthiness of the agent. The third action, also the most complicated one, is **modification**, which updates the trust value of an agent according to the rules and formulas provided by the trust model. Clearly, more actions could be added to the knowledge base if desired. As such, an important contribution of this approach is that the knowledge representation and management system being used here will allow for the storage, retrieval, evaluation, and modification of trustworthiness values not only of agents in a VANET environment, but also of agents in any multi-agent system where trustworthiness needs to be represented.

Our work also provides a detailed design and implementation of the messaging between agents in the VANET environment (Figures 4.6, 4.7, 4.8, and 4.9 in Section 4.3), which clearly specifies how an agent sends a request for location and congestion information; how an agent receives a request, processes it, and sends a response accordingly with the requested information; and how an agent receives a response to its request and processes the requested information, including subsequent evaluation and updating of the information to be used in decision making. Indeed, the proposed messaging details form a communication protocol that enables agents to exchange and understand messages, and an interaction protocol that enables agents to have conversations with one another. This serves as a concrete sample example of how to design and implement communication and interaction protocols for agents to operate effectively and interact with each other productively in a society of agents.

The VANET model originally presented in Minhas et al.[20, 18] is simplified somewhat and then implemented, making use of external software, namely JiST/SWANS, vans, DUCKS, and Protege, and offering a careful experimentation incorporating a variety of metrics to validate the effectiveness of the model. This provides a simulation testbed that can be used as a framework for the simulation of actual traffic flow with large number of cars in a general mobile vehicular ad-hoc network. In fact, due to the significant attention to detail in the provided simulation, we are well positioned to offer a framework that may be deployed within actual vehicles.

The experimental evidence presented in Chapter 5 also serves to provide impressive validation of the multi-faceted trust modeling algorithm that is central to the proposed decision making of the vehicles. With our particular trust modeling in place, even in scenarios where there is considerable deception in the environment, our vehicles are able to perform their path planning extremely well, maintaining an effective speed and travel time, without significant compromise from poor path selection.

In summary, we address an important challenge for mobile transportation environments: facilitating the use of intelligent systems to represent and reason with knowledge about the environment in order to enable effective traffic flow.

Included in our contributions are a proposal for reasoning with numeric information provided by agents, set in a framework for modeling trustworthiness according to confidence values. How majority consensus can be computed for non-Boolean trust modeling is clarified in detail. This research may be of value to trust modeling researchers considering a variety of possible applications.

Another contribution offered is a proposal for reasoning with information that has been obtained through frequent broadcasts and polling. This is distinct from simply requesting information just prior to a critical decision, which may be challenging for environments such as ours with dynamic change and real-time decision requirements.

A final contribution is an explanation for how the concept of knowledge provenance can be integrated into a trust modeling framework, emphasizing the value of knowledge bases for agent decision making. With our design in place, the VANET trust model of Minhas et al. is now specified much more clearly and its overall value is confirmed more definitively. In addition, our messaging description and our extensive simulation testbed now lays the groundwork for deployment of our framework in realworld vehicular networks.

Bibliography

[1] Oxford english dictionary a, http://dictionary.oed.com/cgi/entry/50000000/50000000se148?single=1\&query_type=word\&queryword=ADHD\&first=1\&max_to_show=10\&hilite=50000000se148

[2] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)

[3] Berners-Lee, T., Hall, W., Hendler, J.A., O'Hara, K., Shadbolt, N., Weitzner, D.J.: A framework for web science. Foundations and Trends in Web Science 1(1), 1–130 (2006)

[4] Burnett, C. Norman, T., Sycara, K.: Sources of stereotypical trust in multi-agent systems. In: Proceedings. AAMAS Trust Workshop (Trust-2011) (2011)

[5] Chen, C., Zhang, J., Cohen, R., Ho, P.H.: A trust-based message propagation and evaluation framework in vanets. In: Proceedings of the International Conference on Information Technology Convergence and Services (ITCS) (2010)

[6] Cheney, J., Chiticariu, L., Tan, W.C.: Provenance in databases: Why, how, and where. Found. Trends databases 1(4), 379–474 (Apr 2009), http://dx.doi.org/10.1561/1900000006

[7] Ding, L., Kolari, P., Finin, T., Joshi, A., Peng, Y., Yesha, Y.: On Homeland Security and the Semantic Web: A Provenance and Trust Aware Inference Framework. In: Proceedings of the AAAI Spring Symposium on AI Technologies for Homeland Security. AAAI Press (March 2005), (poster paper)

[8] Dotzer, F., Fischer, L., Magiera, P.: VARS: A vehicle ad-hoc network reputation system. In: Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks. pp. 453–456 (2005)

[9] Fox, M.S., Huang, J.: Knowledge provenance. In: Proceedings of Canadian Conference on AI. pp. 517–523 (2004)

[10] Gerlach, M.: Trust for vehicular applications. In: Proceedings of the International Symposium on Autonomous Decentralized Systems (2007)

[11] Golle, P., Greene, D., Staddon, J.: Detecting and correcting malicious data in vanets. In: Proceedings of VANET (2004)

[12] Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquis. 5, 199–220 (June 1993), `http://dl.acm.org/citation.cfm?id=173743.173747`

[13] Happel, H.J., Seedorf, S.: Applications of ontologies in software engineering. In: International Workshop on Semantic Web Enabled Software Engineering (SWESE'06). Athens, USA (November 2006), `http://fparreiras/papers/AppOntoSE.pdf`

[14] Huang, J.: Knowledge Provenance: An Approach to Modeling and Maintaining The Evolution and Validity of Knowledge. Ph.D. thesis, University of Toronto (2007)

[15] Josang, A., Ismail, R.: The beta reputation system. In: Proceedings of the 15th Bled Electronic Commerce Conference (2002)

[16] Lin, R., Kraus, S., Shavitt, Y.: On the benefit of cheating by self-interested agents in vehicular networks. In: Proceedings of International Autonomous Agents and Multi Agent Systems (AAMAS) (2007)

[17] McGuinness, D.L., van Harmelen, F.: OWL web ontology language overview. W3C recommendation, W3C (Feb 2004)

[18] Minhas, U.F., Zhang, J., Tran, T., Cohen, R.: Promoting effective exchanges between vehicular agents in traffic through transportation-oriented trust modeling. In: Proceedings of International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS) Workshop on Agents in Traffic and Transportation (ATT) (2010)

[19] Minhas, U.F., Zhang, J., Tran, T.T., Cohen, R.: Intelligent agents in mobile vehicular ad-hoc networks: Leveraging trust modeling based on direct experience with incentives for honesty. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT) (2010)

[20] Minhas, U.F., Zhang, J., Tran, T.T., Cohen, R.: A multifaceted approach to modeling agent trust for effective communication in the application of mobile ad hoc vehicular networks. IEEE Transactions on Systems, Man, and Cybernetics, Part C 41(3), 407–420 (2011)

[21] Motik, B., Patel-Schneider, P.F., Grau, B.C.: Owl 2 web ontology language direct semantics. Director (October), 1–17 (2009), `http://www.w3.org/TR/2009/REC-owl2-direct-semantics-20091027/`

[22] Patwardhan, A., Joshi, A., Finin, T., Yesha, Y.: A data intensive reputation management scheme for vehicular ad hoc networks. In: Proceedings of the Second International Workshop on Vehicle-to-Vehicle Communications (2006)

[23] Raya, M., Papadimitratos, P., Gligor, V., Hubaux, J.P.: On data-centric trust establishment in ephemeral ad hoc networks. In: Proceedings of the 27th Annual IEEE International Conference on Computer Communications (IEEE INFOCOM). pp. 1238–1246 (2008)

[24] Rector, A., Rogers, J.: Ontological issues in using a description logic to represent medical concepts: Experience from galen. In: IMIA WORKING GROUP 6 WORKSHOP (1999)

[25] Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson Education (2003), `http://portal.acm.org/citation.cfm?id=773294`

[26] da Silva, P.P., McGuinness, D.L., McCool, R.: Knowledge provenance infrastructure. IEEE Data Engineering Bulletin 26(4), 26–32 (2003)

[27] Spackman, K.A., D, P., Campbell, K.E., D, P., Cote, R.A., hon, D.S.: Snomed rt: A reference terminology for health care. In: J. of the American Medical Informatics Association. pp. 640–644 (1997)

[28] Sure, Y., Bloehdorn, S., Haase, P., Hartmann, J., Oberle, D.: The swrc ontology – semantic web for research communities. In: Bento, C., Cardoso, A., Dias, G. (eds.) Progress in Artificial Intelligence, Lecture Notes in Computer Science, vol. 3808, pp. 218–231. Springer Berlin / Heidelberg

[29] Tran, T.T., Cohen, R.: Improving user satisfaction in agent-based electronic marketplaces by reputation modelling and adjustable product quality. In: AAMAS. pp. 828–835 (2004)

[30] Wang, Y., Vassileva, J.: Bayesian network-based trust model. In: Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on. pp. 372 – 378 (oct 2003)

[31] Weiss, G. (ed.): Multiagent systems: a modern approach to distributed artificial intelligence. MIT Press, Cambridge, MA, USA (1999)

[32] Yu, B., Singh, M.P.: Detecting deception in reputation management. In: Proceedings of the second international joint conference on Autonomous agents and multiagent systems. pp. 73–80. AAMAS '03, ACM, New York, NY, USA (2003), `http://doi.acm.org/10.1145/860575.860588`

[33] Zhang, J.: A survey on trust management for vanets. In: Proceedings of the 25th International Conference on Advanced Information Networking and Applications (AINA) (2011)

[34] Zhang, J., Cohen, R.: Design of a mechanism for promoting honesty in e-marketplaces. In: AAAI. pp. 1495–1500 (2007)

[35] Zhang, J., Cohen, R.: A framework for trust modeling in multiagent electronic marketplaces with buying advisors to consider varying seller behavior and the limiting of seller bids. ACM Transactions on Intelligent Systems and Technology (ACM TIST), to appear (2011)

Appendix A

Confidence Geometric Series

This appendix seeks to further clarify and detail the geometric series equation and design rationale for calculating confidence in Section 4.3.3.1 and to provide examples.

In Section 4.3.3.1 we proposed Equation 4.1(A.1) for calculating the confidence of a report. Equations 2.1(A.2) and 2.2(A.3) are used as the basis for calculating the confidence of report R_j in Equation 4.1(A.1), through a modified summation of a geometric series.

$$Conf(R_j) = (CurrConf(R_j) - 1)(1 - (\alpha\ or\ \beta))^{|G|} + 1 \tag{A.1}$$

The following will describe why a geometric series was necessary.

Equations A.2 and A.3 shown below are used to modify the trust of an agent. In the framework it is necessary to attribute a trust value to each report from an agent, which we define as confidence, due to each report having possibly different attributes, such as age and if the report was observed indirectly.

$$T_A(B) \leftarrow \begin{cases} T_A(B) + \alpha(1 - T_A(B)) & \text{if } T_A(B) \geq 0, \\ T_A(B) + \alpha(1 + T_A(B)) & \text{if } T_A(B) < 0, \end{cases} \tag{A.2}$$

$$T_A(B) \leftarrow \begin{cases} T_A(B) + \beta(1 - T_A(B)) & \text{if } T_A(B) \geq 0, \\ T_A(B) + \beta(1 + T_A(B)) & \text{if } T_A(B) < 0, \end{cases} \tag{A.3}$$

A report's confidence is initially set to the experience-based trust of the agent that provided the report. If Equations A.2 and A.3 were used to atomically increase a report's confidence according to various attributes (Time, Loc, Indirect, etc.), then their influence on confidence would be disproportionate to their value and importance. A simple solution to this issue would be to weight or multiply α and β according to the attribute (Time, Loc, Indirect, etc.). However, this can result in the confidence value being above 100% or below 0%. In addition, to solve this by simply placing a bound on the confidence value (So that max is 100% and minimum is 0%) would not be faithful to the founding research.

Equations A.2 and A.3 implicitly bound $T_A(B)$, and have an effect of decreasing the magnitude by which trust is increased or decreased as the trust value becomes greater or smaller, respectively. Equation A.1 is intended to reflect the culmination of several increases or decreases, according to A.2 and A.3. If you were to graph the trust value over all atomic iterations, the graph would form a Sigmoid function ("S" curve).

Equation A.4 for a geometric series is shown below. Equation A.5 shows the calculation at n terms in the series. This is the type of calculation we need because we need to calculate confidence after Equation A.2 or A.3 has been applied n times (Equivalent to G in Equation A.1). Equation A.5 can not be used because it does not take into consideration the result of the previous calculation, which we need to. Equation A.6 describes our calculation, after Equation A.2 or A.3 has been applied n times, and the series which we need to represent for our calculation. Equation A.6 describes the need for each term of n terms to sum the result of all previous terms. This is due to Equation A.2 and A.3 multiplying α and β by $T_A(B)$ (the previous trust value). The simplification of Equation A.6 is equivalent to Equation 4.1(A.1).

$$a + ar + ar^2 + ... + ar^{n-1} = a\frac{1 - r^n}{1 - r} \tag{A.4}$$

$$a_n = ar^n \tag{A.5}$$

$$\begin{aligned} a_n &= a_{n0} + r(1 + / - a_{n0}) + r(1 + / - a_{n1}) \\ &+r(1 + / - a_{n2}) + ... + r(1 + / - a_{n_{n-1}}) \\ &= (a - 1)(1 - r)^{|n|} + 1 \end{aligned} \tag{A.6}$$

Defining our confidence calculation using Equation A.1(4.1, the simplification of Equation A.6) allows us to utilize Equations A.2 and A.3, their Sigmoid nature and implicit bounding, use of decimal numbers for G (n) (providing a granularity that atomic changes do not allow), and a representation of the calculation in a simple format.

The following example demonstrates the modification of confidence according the time difference attribute.

Example 1: (Modification of Confidence according to Time)

$$
\begin{aligned}
Confidence_0 &= \textit{Agent\_39:trust\_degree}\ (0.6) \\
\\
\alpha &= 0.1 \\
\\
G_{time} &= \text{(TPenal(90)-TimeDiff(45))/TPenal(90)} \\
&\quad \text{*MultiplicativeFactor(4)} \\
G_{time} &= 2\ (\text{Increase } Confidence_0 \text{ twice}) \\
\\
Confidence_0 &= 0.6 \\
Confidence_1 &= (0.6) + \alpha(1 - (0.6)) \\
&= 0.64 \\
Confidence_{2(G_{time})} &= 0.64 + \alpha(1 - (0.64)) \\
&= 0.676
\end{aligned}
$$

(Again using Equation 4.1)

$$
\begin{aligned}
Confidence_2 &= (Confidence_0 - 1)(1 - \alpha)^{|G_{time}|} + 1 \\
&= ((0.6) - 1)(1 - \alpha)^2 + 1 \\
&= 0.676
\end{aligned}
$$