V2V-Net: A Vehicle-to-Vehicle Overlay Structure for Service Deployment in Vehicular Ad Hoc Networks

Anandha Gopalan and Taieb Znati

Department of Computer Science

University of Pittsburgh

Pittsburgh, PA 15260, U.S.A

Email: {axgopala, znati}@cs.pitt.edu

Abstract

Ad-hoc networks are an emerging technology with immense potential. Recent advances in high-tech sensors, communication devices and microprocessors has given rise to a growing commercial interest in ad-hoc networks for inter-vehicular communications. Most of the research in inter-vehicular communication systems has focused on driver safety, accident reporting and multi-hop routing. There is an enormous potential for service deployment in such a network, when vehicles need access to services "anytime, anywhere". The lack of infrastructure, coupled with the time-varying characteristics of vehicular ad-hoc networks, brings about new challenges to the design and deployment of services. This paper addresses these challenges and presents V2V-Net, a vehicle-to-vehicle overlay structure for service deployment in vehicular ad hoc networks. We discuss the main functionalities of the architecture and describe the algorithms for service registration and discovery. The service-architecture also considers the mobility of the vehicles and

provides a mechanism by which this can be incorporated into the framework. Finally, the architecture was evaluated using simulations and the results show that the architecture performs well under different network conditions.

1 Introduction

Advances in wireless technology and portable computing along with demands for greater user mobility have provided a major impetus towards development of an emerging class of self-organizing, rapidly deployable network architectures referred to as ad-hoc networks. Ad-hoc networks, which have proven to be useful in military applications, are expected to play an important role in future commercial settings where mobile access to a wired network is either ineffective or impossible. In this regard, there has been a growing commercial and research interest in deploying ad-hoc networks for inter-vehicular communications [11, 21].

Most of the research in inter-vehicular communication systems has been in the area of driver safety, by avoiding collisions using data about traffic scenarios [27, 6] and multi-hop routing [4, 3, 2]. There is a lot of potential for service deployment in such a network. A typical example is a driver assistance service that provides information about traffic patterns, optimal routes, accidents and safe driving speeds.

Several challenges must be addressed in order to develop an effective service architecture to support the deployment of applications in a scalable manner. These challenges are related to the development of several capabilities necessary to support the deployment of a service architecture for vehicular ad-hoc networks. These capabilities include: Service registration, Service discovery, Vehicle location, and Traffic routing and forwarding[1].

Node[2] mobility, coupled with the limitation of computational and communication resources, brings about a new set of challenges that need to be addressed in order to

[1]The focus of this paper is primarily on the overlay design of this framework

[2]We will use the term node and vehicle interchangeably in this paper

enable an efficient, robust and scalable architecture for service deployment in vehicular ad-hoc networks. This node information, however, changes dynamically, as the node moves from one location to another. Efficient mechanisms must, therefore be in place to update this information as nodes move.

The main contribution of this paper is in providing a novel Vehicle-to-Vehicle overlay structure for vehicular ad hoc NETworks (*V2V-Net*) that allows for service deployment in vehicular ad-hoc networks. The proposed framework is scalable, robust and efficient. V2V-Net is efficient, by not requiring nodes to maintain routing information. Bootstrapping in V2V-Net does not require the knowledge of other V2V-Net nodes in the network. A node only needs to know the $hash$ function used in the network and the network area map that maps the zone ids to the zones to register and query for services. V2V-Net is also flexible by not imposing a routing algorithm on the service architecture. The routing algorithm chosen could depend on the properties of the underlying network.

The main components of this service-architecture revolve around the concepts of *zones*, *rendezvous area* and *mobility profile*. A zone is a physical area in the network that acts as an information database for available services in the network. The zones are organized as a virtual DHT-based structure that enables service location through distributed indexing. The novelty of this approach is that no specific table content is managed by the nodes. It uses a virtual structure that is tightly coupled to the physical structure of the network to locate nodes where service information is stored.

The *rendezvous area* of a node is the physical area where the node is most likely to be located. This is used as a *congregation* point by nodes to contact other nodes. In the case when a node is mobile, it leaves behind its mobility information with select $proxy$ nodes within its rendezvous area. This mobility information constitutes the *mobility profile* of the node and consists of the expected direction and speed of travel. The mobility profile of a mobile node enables other nodes to predict the location of this

node.

The rest of the paper is organized as follows: Section 2 details the related work in this area while Section 3 details the network characteristics used in V2V-Net, Section 4 details the different components of the system architecture and the algorithms used and Section 5 concludes the paper and identifies areas of future work.

2 Related Work

This section details some of the work related to this paper and clearly differentiates our work from them.

Service discovery provides the interface by which clients and servers in a network discover the services that are available in the network. It also provides a consistent view of the network by providing mechanisms by which servers (or clients) can join and leave the network. Service discovery for ad-hoc networks is still a very new area of research. There have been some protocols for service location and discovery that have been developed for LANs, namely: Service Location Protocol [8] and Simple Service Discovery Protocol [28]. SLP relies on agents to search for and locate services in the network; an *user agent* is used on behalf of users to search for services, while a *service agent* advertises services on behalf of a server and finally a *directory agent* collects the advertisements that are sent out by the *server agent*. SSDP uses a specific protocol and port number to search for and locate services in the network. HTTP UDP is used on the reserved local multicast address *239.255.255.250* along with the $SSDPport$ while searching for services. Both SLP and SSDP cannot be directly used for vehicular ad-hoc networks due to their reliance on an existing network structure.

CAN [22] (Content Addressable Network) provides a distributed, Internet-scale hash table. The network is divided into zones according to a virtual co-ordinate system, where each node is responsible for a zone in the network. Given (key,value) pairs, CAN maps the key to a point P in the co-ordinate system using a uniform hash function. The

corresponding (key,value) pair is stored at the node that owns the zone in which point P is located. To retrieve a value corresponding to a key, a node can apply the same hash function to obtain a point P_1 in the co-ordinate system and get the value from the node that owns the zone in which P_1 lies. Our approach is similar to that of CAN, in mapping the hash value of services to zones, but we differ by having more nodes in a zone. Also, when a new node arrives, the zone is not split into two and this overhead is avoided. Mobility is also incorporated into our architecture by using the *mobility profile management base*.

The Landmark routing hierarchy [19, 20] provides a set of algorithms for routing in large, dynamic networks. Nodes in this hierarchy have a permanent node ID and a $landmark$ address that is used for routing. The $landmark$ address consists of the list of the IDs of nodes along the path from this node to a well known $landmark$ node. Location service is provided in the landmark hierarchy by mapping the node IDs to addresses. A node X chooses its address server by hashing its node id. If a node Y exists with that ID, it is chosen as Xs address server. Otherwise, the node whose ID is the closest to the hash value is chosen to be Xs address server. Our scheme is similar to the landmark scheme in terms of using the address server for a service, but we do not rely on any one specific node to hold this information. We elect a group of nodes from within the $zone$ to maintain this information. The information is stored in a manner such that only k out of N fragments are necessary to *re-construct* it.

The Grid location service (GLS) [14] provides distributed location information service in mobile ad-hoc networks. GLS combined with geographic forwarding can be used to achieve routing in the network. A node X "recruits" nodes that are "closest" to its own ID in the ID space to act as its location server. Similar to GLS, our scheme also uses a distributed scheme to store location information, but instead of using one node in the network as a location server, a group of nodes are selected from within an area to act as the location server. The information is stored in a manner such that only

k out of N fragments are necessary to *re-construct* it.

[13, 12, 24] use the concept of home agents (or) home regions. Each node in the network is mapped to an area (using a hash function) in the network that is designated as its home agent (or) home region. The home region holds the location information about the mobile nodes which map to this location. A mobile node updates its location information by sending updates to its home region. In our scheme, a mobile node does not keep updating its *rendezvous area*, rather it leaves a trail behind that can be used by other nodes to locate it. Also, the *rendezvous area* is used as a "rendezvous point", where the node is most likely to be located.

Ekta [10] integrates DHTs (akin to peer-to-peer systems like [25, 29, 5, 18, 16]) into MANETs. This provides an efficient architecture for constructing distributed applications and services in MANETs. Our architecture differs from this scheme, since we do not use Pastry [23] for the purpose of our DHT. The DHT is constructed in a manner that allows it to take advantage of the location information provided in the network. V2V-Net is more lightweight when compared to Pastry and takes node mobility into consideration.

[15, 26] provide basis for resource discovery in MANETs. Our main contribution when compared to these works is the use of a virtual DHT-based system for providing resource discovery in vehicular ad-hoc networks.

3 V2V-Net: Network Characteristics

A vehicle-to-vehicle ad-hoc network is a special case of a mobile ad-hoc network where node movement and speed are restricted by the presence of obstacles (buildings, parks) and roads. We assume that each vehicle has a map of the network area that contains the zones and their respective *zone id*s.

3.1 Zones and Rendezvous Areas (RAs)

Consider an ad-hoc network covering a specific geographical area, denoted as Λ. We perceive this area to be divided into zones as shown in fig. 1, ($\Lambda = \bigcup_i Z_i$, where Z_i is a zone). A zone can, for example, represent neighborhoods, or an administrative domain where geographical proximity facilitates communication between nodes of the zone. A zone Z_i, is uniquely characterized by an *m-bit* identifier. A neighborhood is defined as a physical area within a zone in the network.

Each zone Z_i, contains one or more nodes in the network whose *rendezvous areas* map to a neighborhood in Z_i. The *rendezvous area* of a node is defined as the physical area within a zone where the node is most likely to be located.

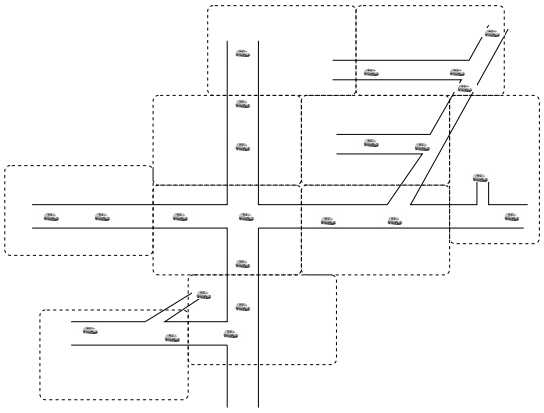

Figure 1: Partitioning the network into zones

The *zone*s in the network form a virtual DHT-based distributed database that hold service information. Any node wishing to register a service (or) query about a particular service (S_i), must first calculate the *hash value* (using a pre-defined hash function that is known to all the nodes in the network), based on its service id[3]. This *hash value* maps to a *zone id* of a zone in the network that is responsible for holding the information about S_i. The physical co-ordinates of the zone associated with the *zone id* is provided in the network area map that is available with the node.

[3]Each service is associated with a service id

These *zone*s always exist in the network, the nodes inhabiting these zones, however can be transient. The database is thus, a fixed size hash table with each entry corresponding to a $zone$ in the network. To ensure that there are no *hot spot*s in the network due to the hashing of service ids to zones, the hash function is chosen to be a $uniform$ hash function. This results in producing $balanced$ hash buckets, given a fixed size hash table [7].

The size of a zone (in terms of the number of nodes) plays an important part while designing this framework. A large zone results in a large number of nodes that are a part of this zone. This leads to a higher overhead in terms of managing the zone. Having a small zone has the advantage of having a reduced overhead in managing the zone, but the lack of nodes in the zone may have an impact on the database by not having enough nodes to maintain information about the services in the network. Also, not having enough nodes in a zone has a direct impact on the *mobility profile management base*. Given the map of the network area, we can choose to model the zone based on zip code or block and street names.

Vehicles in the network are characterized by a unique $identifier$ and their *rendezvous area*. A mobile node, A, upon departure from its RA leaves behind some information so that other nodes can use this information to interact with A. This information is left with the *mobility profile management base*, which consists of a set of $proxy$ nodes that are responsible for holding the mobile node's mobility information. This information is in the form of a vector that contains the expected direction and speed of travel of A and is called the *mobility profile* of A. Since a node in the network is a vehicle with an intended destination, this information is easily computed using the network map. Using the *mobility profile*, other nodes in the network can probabilistically determine the location of A to initiate interaction with A.

4 V2V-Net: System Architecture and Services

4.1 Service Registration

A node A, managing a service (or) a collection of services must register its service interface(s) with the network. A registers this information by hashing the *service id* to obtain a hash value. This hash value corresponds to a *zone id* of a zone (Z) in the network. Using the network area map, A knows the location of Z and registers its service interface, rendezvous area ($RA(A)$) and relevant attributes with the nodes in Z.

Algorithm 1 details the process by which node A, containing a service with id s_i, registers this information with the network, (H is the hash function).

Algorithm 1 Service Registration

Input: s_i, H
Output: Result
SERVICE-REGISTER(s_i, H)
(1) Calculate $Z_{id} = H(s_i)$
(2) Use network map to get the zone (Z) corresponding to Z_{id}
(3) Register [$RA(A)$, s_i, other relevant attributes of service, if necessary] with the nodes in Z
(4) **return**

4.2 Service Discovery

Service discovery in V2V-Net is performed in a manner, similar to service registration (Section 4.1). A node A, wishing to locate the service of interest (s_i) in the network, first calculates the hash value corresponding to the *service id*. This hash value maps to a *zone id* of a zone (Z) in the network that is responsible for holding information about s_i. This information consists of a list of peers (along with their RAs and additional information) that offer this service. Node A can then choose to select a particular node

offering this service. This decision can be made based upon factors like past history, distance to the RA of the node and the stability of the node.

Algorithm 2 details the process by which a node A, discovers information about a service with service id s_i, (H is the hash function).

Algorithm 2 Service Discovery

Input: s_i, H
Output: Result
SERVICE-DISCOVER(s_i, H)
(1) Calculate $Z_{id} = H(s_i)$
(2) Use network map to get the zone (Z) corresponding to Z_{id}
(3) Nodes in Z replies with a list of peers that own the service associated with s_i
(4) **return**

4.3 Mobile Node Location and Node Interaction

4.3.1 Mobile Node Location

Vehicular mobility must be incorporated into the service architecture. Nodes in the network must have the ability to predict the location of a mobile with high probability. The *mobility profile* is used by a mobile node to leave behind some information about its mobility pattern; using this information, other nodes in the network can *predict* the current location of the mobile node.

Consider the situation when a node A becomes mobile, or leaves its *rendezvous area*. Node A must leave behind some information using which other nodes in the network can locate A. Earlier methods to handle mobility were using mobile IP [9] which used the concept of a *home-address* and a *care-off* address. The problem with this scheme was that traffic intended for node A is held at the *home-agent* of A, until node A sends its home-agent its care-off address. This leads to a lot of overhead in the network (and also an increase in the latency). Our algorithm does not require a node to keep updating its position to its *rendezvous area*. Given the network map, node A

has the information about its intended destination and hence its initial direction and speed of travel. Node A leaves behind this information in the form of a *mobility profile* (as mentioned in algorithm 4) with select *proxy* nodes that act as the *mobility profile management base*. With a reasonable probability, nodes which wish to contact node A can predict the new location of node A based on its *mobility profile* and the `elapsed` time since this information was provided [1].

The mobile node A, needs to find the set of *proxy* nodes that form the *MPMB* in its *rendezvous area* to leave behind its *mobility profile* with. To recruit the $proxy$ nodes, node A sends out a broadcast message $once$ at its highest power (to ensure the highest broadcast radius) and waits for replies from the other nodes. The *mobility profile* is encoded in a manner such that k out of N (number of nodes that reply) fragments are enough to re-construct the original $profile$. This is to ensure that the mobility profile is still available even when a few proxy nodes decide to leave the *rendezvous area*. [17] defines a key encoding scheme that can be used for this purpose. The set of steps that are involved in building the *MPMB* are given in algorithm 3.

Algorithm 3 Building the MPMB

Input: k
Output: Result
BUILD-MPMB(k)
(1) Broadcast-Msg to recruit (Send broadcast at highest power)
(2) Receive-Replies
(3) Let number of replies received be N
(4) Encode *mobility profile* such that k out of N fragments are enough to re-construct the original $profile$
(5) **return**

The advantage with this scheme is that it saves power, since we send only one broadcast per node that needs to form the *MPMB* to hold its mobility profile.

A node also sends back $corrections$ with regards to its *mobility profile* to its rendezvous area. Consider a scenario, when a node A has left its rendezvous area and

hence left its *mobility profile* with select $proxy$ nodes in $RA(A)$. Along the path to its intended destination, node A discovers that its behavior is different when compared to its mobility profile. Node A, then sends back a $correction$ to the $MPMB$ in its rendezvous area. The $MPMB$ in $RA(A)$ update the mobility profile of A to reflect this change. Node A also leaves some $hints$ about the change in its mobility profile in its current location.

4.3.2 Node Interaction

Once a node discovers the service of interest in the network, it resolves the node (containing the required service) into its *rendezvous area*. A mechanism is required by which node interaction can be achieved. Algorithm 4 and figure 2 detail the procedure by which a node *C* obtains a resource from a node *S* that is mobile.

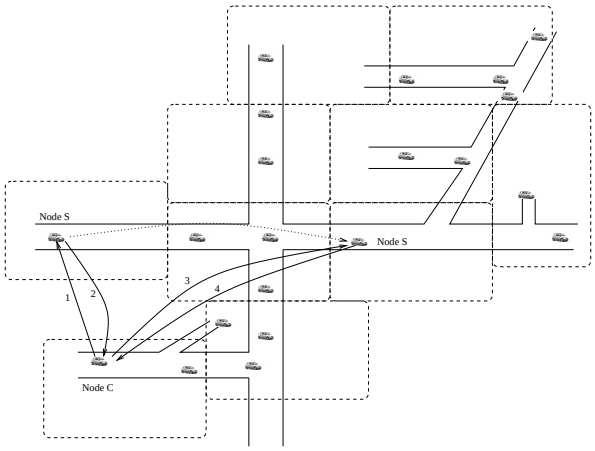

Figure 2: Mobile Node Interaction

The components of the metric $[t_0, V(t_0), D(t_0), P_V(t), P_D(t)]$ in algorithm 4 are:

- t_0: starting time
- $V(t_0)$: expected starting speed.
- $D(t_0)$: expected initial direction.

Algorithm 4 Handling node mobility

Input: Void
Output: Result
NODE-MOBILITY()
(1) C receives RA(S) from the zone holding the service information
(2) C sends messages towards RA(S) (msg 1 in the figure)
(3) **case** S is in RA(S)
(4) Connection is established between C and S
(5) **case** S is currently not in RA(S)
(6) The nodes in RA(S) reply with the *mobility profile* of S, a metric: $[t_0, V(t_0), D(t_0), P_V(t), P_D(t)]$ (msg 2 in the figure)
(7) C uses the *mobility profile* of S to determine with high probability the current position of S and sends messages to this location (msg 3 in the figure)
(8) S upon receiving the messages by C acknowledges it (msg 4 in the figure) and initiates a conversation with C
(9) **return**

- $P_v(t)$: Predictor for speed after t units of time since departure.
- $P_d(t)$: Predictor for direction after t units of time since departure.

In the case when node S has not yet reached its *predicted* position, the nodes in the vicinity of this position hold on to the message sent by C until they hear from node S or there is a $timeout$, after which the message is discarded (the $timeout$ is sufficiently large to incorporate in-accuracies in the *mobility profile*). This ensures that node S does indeed receive the message.

5 Conclusion and future work

The main contribution of this paper is in providing a novel Vehicle-to-Vehicle overlay structure for vehicular ad hoc NETworks (*V2V-Net*) that allows for service deployment in vehicular ad-hoc networks. The proposed framework is scalable, robust and efficient. V2V-Net is efficient since nodes do not maintain and update routing tables.

Bootstrapping does not require the knowledge of other V2V-Net nodes. A node needs to know the $hash$ function used in the network and network map that maps the zone ids to the zones to register and query for services. V2V-Net is also flexible by not imposing a routing algorithm on the service architecture. The routing algorithm chosen for the architecture could be chosen depending on the properties of the underlying ad-hoc network.

Service registration and discovery are achieved by hashing the *service id* associated with a service to obtain the *zone id* of the zone associated with it. V2V-Net is scalable, since the hash function ensures that the information stored in the network is spread across the various $zone$s.

The mobility of the nodes in the network is incorporated into V2V-Net by using the *mobility profile* of a node (which consists of the expected direction and speed of travel of the node), to predict the location of the node.

There is a lot of potential for future work in this area. Security needs to be incorporated into this service-architecture at various levels, be it service registration and discovery or handling the *mobility profile*s using the *MPMB*.

References

[1] A. R. Aljadhai and T. Znati. Predictive mobility support for QoS provisioning in mobile wireless environments. *IEEE Journal on Selected Areas in Communications*, 19(10):1915–1930, 2001.

[2] Boon-Chong Seet and Genping Liu and Bu-Sung Lee and Chuan Heng Foh and Kai Juan Wong and Keok-Kee Lee. A-STAR: A Mobile Ad Hoc Routing Strategy for Metropolis Vehicular Communications. In *NETWORKING*, pages 989–999, 2004.

[3] C. Lochert and H. Hartenstein and J. Tian and H. Fler and D. Herrmann and M. Mauve. A Routing Strategy for Vehicular Ad Hoc Networks in City Environments. In *Proc. IEEE Intelligent Vehicles Symposium (IV2003)*, Jun 2003.

[4] Christian Lochert and Martin Mauve and Holger Fuler and Hannes Hartenstein. Geographic routing in city scenarios. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(1):69–72, 2005.

[5] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46+, 2001.

[6] Dedicated Short Range Communications (DSRC) Home. `http://www.leearmstrong.com/DSRC/DSRCHomeset.htm`.

[7] Donald E. Knuth. *Art of Computer Programming, Volume 3*. Addison-Wesley Professional, 1998.

[8] E. Guttmann and C. Perkins and J. Veizades and M. Day. Service Location Protocol, Version 2. *Internet Draft, RFC 2608*, 1999.

[9] IP Routing for Wireless/Mobile Hosts (mobileip) Charter. `http://www.ietf.org/html.charters/mobileip-charter.html`.

[10] H. Pucha, S. M. Das and Y. Charlie Hu. Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks. In *In Proc. of the 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2004)*, December 2004.

[11] Hannes Hartenstein and Bernd Bochow and André Ebner and Mathhias Lott and Markus Radimirsch and Dieter Vollmer. Position-aware ad hoc wireless networks for inter-vehicle communications: the Fleetnet project. In *MobiHoc*, pages 259–262, 2001.

[12] J. P. Hubaux, J. Y. Le Boudec, and M. Vetterli Th. Gross. Towards self-organizing mobile ad-hoc networks: the terminodes project. *IEEE Comm Mag*, 39(1):118 – 124, January 2001.

[13] I. Stojmenovic. Home agent based location update and destination search schemes in ad hoc wireless networks. Technical Report TR-99-10, Computer Science, SITE, University of Ottawa, Sep. 1999.

[14] J. Li and J. Jannotti and D. De Couto and D. Karger and R. Morris. A Scalable Location Service for Geographic Ad-Hoc Routing. In *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, pages 120–130, August 2000.

[15] Jivodar B. Tchakarov and Nitin H. Vaidya. Efficient Content Location in Mobile Ad Hoc Networks. In *Proc. IEEE International Conference on Mobile Data Management (MDM 2004)*, January 2004.

[16] John Kubiatowicz, David Bindel, Yan Chen, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westly Weimer, Christopher Wells, and Ben Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of ACM ASPLOS*. ACM, November 2000.

[17] Gretchen Lynn. ROMR: Robust Multicast Routing in Mobile Ad-Hoc Networks. *PhD. Thesis, University of Pittsburgh*, December 2003.

[18] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nataraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. Peer-to-Peer computing. Technical Report HPL-2002-57, HP Laboratories Palo Alto, March 2002.

[19] Paul F. Tsuchiya. Landamrk Routing: Architecture, algorithms and issues, MTR-87W00174, MITRE, May 1988.

[20] Paul F. Tsuchiya. The Landmark Hierarchy: A New Hierarchy for Routing in very large Networks. In *Proc. ACM SIGCOMM*, pages 35–42, August 1988.

[21] R. Morris, J. Jannotti, F. Kaashoek, J. Li, D. De Couto. CarNet: A scalable ad hoc wireless network system. In *Proceedings of the 9th ACM SIGOPS European workshop: Beyond the PC: New Challenges for the Operating System, Kolding, Denmark, September.*, 2000.

[22] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, 2001.

[23] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–350, 2001.

[24] Seung-Chul M. Woo and Suresh Singh. Scalable Routing Protocol for Ad Hoc Networks. *Journal of Wireless Networks*, 7(5), Sep. 2001.

[25] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. *ACM Sigcomm*, 2001.

[26] U. Kozat and L. Tassiulas. Network Layer Support for Service Discovery in Mobile Ad Hoc Networks. In *Proceedings of IEEE INFOCOM*, 2003.

[27] Xue Yang and Jie Liu and Feng Zhao and Nitin H. Vaidya. A Vehicle-to-Vehicle Communication Protocol for Cooperative Collision Warning. In *MobiQuitous*, pages 114–123, 2004.

[28] Yaron Y. Goland, Ting Cai, Paul Leach, Ye Gu. Simple Service Discovery Protocol/1.0. *Internet Draft*, Oct. 1999.

[29] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.