QuickSilver: Application-driven Inter- and Intra-cluster Communication in VANETs

Riccardo Crepaldi
University of Illinois
at Urbana Champaign
rcrepal2@illinois.edu

Mehedi Bakht
University of Illinois
at Urbana Champaign
mbakht2@illinois.edu

Robin Kravets
University of Illinois
at Urbana Champaign
rhk@illinois.edu

ABSTRACT

Support for efficient vehicle-to-vehicle communication is increasingly more important with the emergence of newer vehicles equipped with one or more wireless interfaces. While the applications aimed at such networks range from car-to-car chats to sharing dynamic map data, current approaches to inter-vehicular networking have been designed based on either node-centric or content-centric communication, but not both. The challenge for a comprehensive solution arises from the high mobility of nodes as well as the heterogeneity of contact patterns. In this paper, we propose QuickSilver, a system architecture that meets this challenge by leveraging an intrinsic characteristic of vehicular networks - clustering. By being aware of such clustering, Quicksilver enables a seamless integration of two networking paradigms, one for node-centric communication between members of the same cluster, the other for effective content dissemination and exchange during short cluster-to-cluster contacts. To achieve this goal efficiently, Quicksilver employs a novel combination of channel management and light-weight clustering that enables detection and management of headless, multi-hop clusters at very low cost. Evaluation results show that Quicksilver enables close-to-optimal performance when nodes across different clusters communicate.

1. INTRODUCTION

Vehicular networks (VANETs) turn the tables on mobile social networks, where the devices are carrying the people instead of the people carrying the devices. The promise of vehicular networks has led many researchers to design systems and protocols for enabling wireless communication between vehicles. Since it is still under discussion what type of applications need to be supported in vehicular networks, many general solutions have been proposed based on ad hoc [1, 2] or DTN-based [3, 4] routing. However, these solutions focus on point-to-point communication between vehicles, which may not always be the best communication paradigm for vehicular networks. Additionally, they overlook the dynamic characteristics of the networks, missing out on important solutions and optimizations. On the other hand, many solutions have been proposed for more specific scenarios that focus on a vehicle's ability to find an open or accessible access point [5, 6]. These solutions only support applications that need access to the Internet and also ignore the available networking and data resources found locally around a given vehicle.

In reality, vehicular networks are a special form of an opportunistic social network with highly constrained mobility rules that typically result in heavy clustering in the network [7]. The concept of clustering in vehicular networks is in no way new and it has been used as a means for providing optimized communication in both DTN protocols to improve forwarding decisions and in last-hop style vehicular networks to support extended connectivity to two or more hops around an access point. However, if one looks at clustering not as an optimization possibility, but as an intrinsic characteristic of vehicular networks, clustering not only shapes the network topology, but also results in a set of applications with requirements and communication patterns that are so different that clearly they cannot be supported efficiently by a single communication paradigm.

The distinct mobility patterns and sheer scale of VANETs result in geographically wide-spread networks with unique contact opportunities between the diverse nodes. Essentially, related and non-related clusters of vehicles may travel together for long periods of time, briefly encountering other random nodes and clusters that are traveling in different directions. This clustered mobility pattern results in interesting data exchange patterns in VANETs. Although the people in the vehicles inside a cluster may not actually know each other, they may still want to share music, videos, or even just chat with each other because they are going in the same direction like old CB users. Since clusters are somewhat stable, *intra-cluster* communication should support these *node-centric* applications with relatively stable, good quality links to specific vehicles. However, such support is no longer necessary, nor even possible, beyond the border of a cluster. Therefore, the focus of *inter-cluster* communication should not be on the endpoints of the communication but instead on supporting *content-centric* applications that aim to distribute shared data, such as dynamic maps and participatory sensor data, with many unknown nodes across the network.

To cover the whole scope of VANETs, many researchers have classified VANETs as Delay Tolerant Networks (DTNs) with dynamic mobility models, network partitioning and the natural formation of clusters. Since energy is not a severe constraint in moving vehicles, DTN-based VANET routing solutions have focused on supporting node-to-node communication that is robust to network partitioning and intermittent connectivity. On a large scale, it is not clear that such node-centric support is needed when nodes are far apart. Instead, support for geographically distributed nodes may be better off relying on 3G- or hot-spot-based solutions. However, content-centric applications, which rely on probabilistic dissemina-

tion of data between clusters, match well with existing DTN-based communication. While probabilistic replication mechanism can be used, the main challenge comes from establishing efficient inter-cluster communication due to the short connection times and contention for the channel caused by the clustering of vehicles. On the local scale, clustered nodes within a smaller geographic area may want to communicate with each other. However, in this case, simpler ad hoc-based routing solutions may be sufficient. While some existing solutions have leveraged the natural clustering in VANETs to optimize message delivery, they have not considered the important impact of such clustering on the application space.

While intra- and inter-cluster communication have very diverse requirements, a complete VANET solution must support both. To this end, we propose QuickSilver, a system architecture built around a novel, passive cluster management mechanism that does not require the use of cluster ids or other explicit clustering mechanisms. This clustering mechanism enables both intra-cluster networking support for node-centric applications, as well as inter-cluster networking support for content-based applications. For the latter, nodes within a cluster implicitly cooperate to establish as many contention-free inter-cluster communication channels, without requiring explicit channel allocation mechanisms. The goal of QuickSilver is the efficient the use of the available resources to guarantee that no harmful competition over the limited channel bandwidth is engaged. Our evaluation shows that QuickSilver's clustering mechanism can maintain good consistency for cluster memberships, despite high mobility and its distributed nature. Additionally, we show that QuickSilver is able to establish multi-link inter-cluster connections that perform significantly better than the existing hierarchical approaches, and in some scenarios reach 95% of the performance of an optimal, centralized solution, without incurring any control overhead. In the rest of this paper, we discuss the challenges of such a comprehensive solution and present the design and evaluation of QuickSilver.

2. VANET ECO-SYSTEMS

The formation of clusters in vehicular networks has been observed in different scenarios. For example, the analysis of mobility traces from 600 taxis over an 8 hour period on a working day in San Francisco [7], showed that while 40% of the cabs were sparsely connected, the remaining 60% were part of clusters of more than 20 vehicles, whose routes periodically intersect generating inter-cluster contacts. Using short to mid-range transceivers, vehicles can communicate with each other to distances up to 200m [8]. This distance, which is certainly larger than the average distance between vehicles in a cluster, is also likely to be shorter than the length of the cluster itself. This implies that from a networking point of view, clusters can be seen as partitions of the network with a number of stable, multi-path links among their nodes.

2.1 Cluster-Aware Communication

Given this clustering and the diversity of applications in VANETs, many different kinds of applications may coexist, each characterized by different requirements and goals. However, these applications can be divided into two categories: node-centric and content centric. Node-centric applications focus on providing a distinct communication channel between two or more end hosts. For example, nodes A and B in Figure 1(a) are participating in a voice conversation, while nodes X and Y are using a real-time instant messaging application. To support these node-centric applications, each node must be able to learn about about each other's existence to be able to establish a good quality and long lived link for bounded-delay communication, both of which limit node-centric communication to within a cluster. Essentially, except for scenarios where nodes follow predetermined routes (i.e., DieselNet [9]), it is extremely unlikely to meet a specific user during an inter-cluster contact, and even when this happens, the contacts are too rare and short-lived to support node-centric networking.

On the other hand, content-centric applications focus on data that lives in the network [10, 11, 12], where nodes are the relays that spread the data through the network. This type of data is typically spread through the network based on nodes' interests in a particular type or class of data. For example, two nodes in one cluster in Figure 2(b)) are expressing interest in some type of music, while one node in the other cluster is sharing songs. If a path exists between the content provider and the consumers, the data can be shared among the nodes. In this case, the identity of the nodes is completely irrelevant, only the interests they express matter. This kind of communication can cross the boundaries of a cluster. Even with short-lived inter-cluster connections, applications can benefit from receiving as much data as possible from the other cluster.

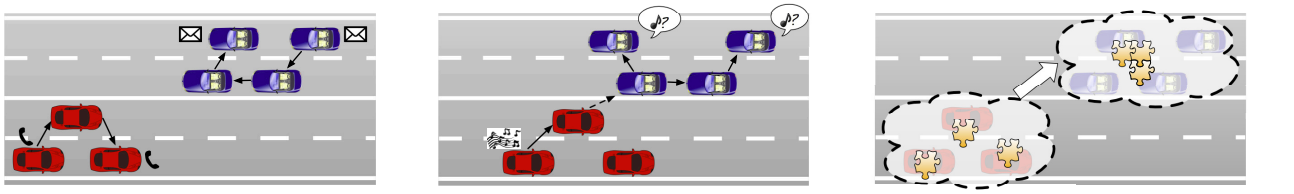
Content-centric applications can also be used to collect and disseminate information about the environment [10]. The vehicles in Figure 1(c) are equipped with a large number of sensors to collect environmental information. This information can be used to augment a map with live data, useful for example for route planning. In this type of application, the data is shared among vehicles that have been traveling together, and each of them is part of a distributed storage. When a cluster traveling in the opposite direction is encountered, it is irrelevant which nodes share the information, as long as it is transferred from one cluster to the other, where it can be distributed to all nodes even when the cluster contact has expired.

2.2 Cluster Contacts

Contact opportunities between nodes that belong to different clusters are generally short. Given two vehicles traveling on an highway in opposite directions and approaching each other at speeds v_a and v_b, contact is possible only while their relative distance is smaller than the wireless coverage range R. The contact duration in this case is $T_{cont} = 2R/(v_a + v_b)$. Assuming a coverage range of $200m$ and a speed of $15m/s$ for both cars, a contact between two vehicles can last approximately $15s$, a very short time for any significant data exchange. However, cluster contact opportunities last from when the first contact between two vehicles is possible, until the distance between the last pair of cars becomes larger than R. For two clusters traveling in opposite directions, with velocities v_a and v_b, and cluster lengths L_a and L_b, both larger than R (i.e., multi-hop clusters), the overall duration of the cluster contact is $T_{c\_cont} = \lceil L_a/2R \rceil \cdot T_{cont}$, which could allow a significant amount of data exchange among the two clusters if the communication channels are managed wisely.

2.3 Cluster Management

Understanding the network topology and its organization in clusters is necessary to define the boundaries of intra-cluster connectivity and determine which links belong to an inter-cluster contact. Traditional cluster management approaches are the basis of communications systems such as Bluetooth or ZigBee, where clustering enables coordinated, hierarchical communication to save energy and increase security. A similar hierarchical structure, based on the election of a cluster head (CH), has been considered in previous research work for VANETs [13, 14, 15]. For example, clustering can be used to coordinate medium access to avoid collisions [16] by allowing the CH to acquire information about the cluster members and then coordinates their accesses to the share



(a) (b) (c)

Figure 1: The types of communication that characterize a VANET: *intra-cluster* (a), *inter-cluster* (b), *cluster-to-cluster* (c)

medium using a TDMA approach. All of these types of approaches view of clusters in VANETs as single-hop subnetworks concentrated around the cluster head. Since membership typically changes rapidly due to the nodes' independent mobility patterns in mobile networks like VANETs, multi-hop clusters are typically dismissed due to the amount of overhead required for electing a cluster head and maintaining cluster membership information. However, the use of multi-hop links has been proposed in VANETs without explicit clustering. For example, CASCADE [6] is an access point-based approach where vehicles traveling in the same direction forward data through each other to an access point. However, approaches like CASCADE are not able to leverage inter-cluster contacts since they do not have any concept of a cluster.

2.4 Inter- and Intra-Cluster Routing

Given the diversity of application requirements and the dynamic mobility in the network, routing solutions for VANETs must be able to integrate both intra-cluster, node-based routing with bounded delays and inter-cluster content-based routing with quick and efficient data dissemination. Traditional ad hoc routing protocols (i.e., DSR [17], AODV[18]) can only need slight modifications to provide the bounded delays for intra-cluster communication, but fail dramatically to to support the opportunistic inter-cluster networking. On the other hand, store-carry-and-forward DTN protocols (i.e., Prophet [19], Spray and Focus [4]) as well as protocols specifically designed for data-centric VANET applications (i.e., Locus [10]) are resilient to partitioning, but cannot effectively support node-centric applications. Essentially, a one-size-fit-all protocol cannot satisfy the requirements of both intra- and inter-cluster networking. Although protocols have been designed with one or the other pattern in mind, it is unclear how they would or should interact in the same system. If the two protocols compete in an uncoordinated manner to access the shared resources (i.e., available channels and bandwidth) on a node, their combined performance might be severely degraded. Only with a careful system design that is aware of the network topology and cluster structure and coordinates the use of resources can the full potential of VANETs be achieved.

3. QUICKSILVER

While node-centric and content-centric communication in VANETs have very different network requirements, the key feature of clusters enables a symbiotic solution. In this section, we present QuickSilver, a system architecture that leverages the clustered topology for supporting the different types of communications in VANETs. Cluster detection is pivotal to tracking the boundaries inside which node-centric communications can happen. When nodes know what other nodes are in the same cluster, they can cooperate to route intra-cluster traffic, and at the same time concentrate their effort on detecting inter-cluster contacts and maximize the amount of data that is exchanged between two clusters.

To enable effective and efficient communication in VANETs, the QuickSilver design is inspired by the following principles. First, QuickSilver is *completely distributed*. No infrastructure is needed for communication. Second, QuickSilver employs *lightweight clustering*, where clusters form and behave in an uncoordinated manner without requiring a cluster ID. There are no cluster coordinators in QuickSilver. Therefore, each node builds its own view of the cluster, which might be partial and differ from node to node. Finally, QuickSilver supports *cluster-aware communication*, where intra-cluster networking leverages the fast delivery guaranteed by stable paths inside a cluster and inter-cluster networking manges the cluster-to-cluster channels to opportunistically support content-centric communication.

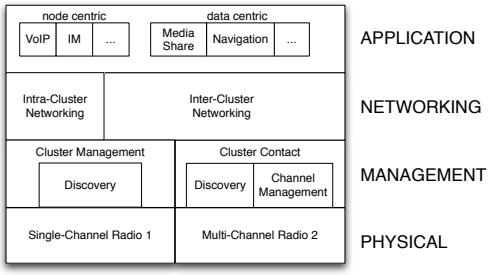


Figure 2: QuickSilver system architecture

3.1 System Architecture

The general architecture of QuickSilver is shown in Figure 2. Similar to [3], QuickSilver utilizes two radio interfaces to support concurrent communication using orthogonal channels. QuickSilver is not tied to a specific wireless standard. It can use generic 802.11a/g/n, or the 802.11p standard designed for Digital Short Range Communications (DSRC). This dual radio setup allows the vehicles to maintain their intra-cluster connectivity, and at the same time look for inter-cluster contact opportunities. However, the resource allocation policy in QuickSilver is different from that of other clustering based approaches that prioritize intra-cluster communications by assigning more channels for it and reserving only one channel for data exchange between clusters. Since intra-cluster communication is typically for longer-lived connections, QuickSilver uses one radio and a fixed channel for all intra-cluster communication. Therefore intra-cluster communication is easier to manage and requires no coordination, enabling better support for node-centric communication. Additionally, it leaves more channels available for the short-lived inter-cluster communication, increasing the bandwidth achievable during an inter-cluster contact.

The *management* layer, which includes *Cluster Management* and *Cluster Contact* components, is the heart of QuickSilver's clustering. The goal of the cluster management component of QuickSilver is to discover neighbors with stable links, and build Cluster Membership Lists (CML) in each node. Cluster formation is based on the rationale that nodes traveling together in the same direction

and at similar speeds experience links that are long-lived enough to support multi-hop path-based intra-cluster routing. Therefore, only nodes connected through such *stable links* are considered members of the same cluster.

Cluster membership is very dynamic, with continuous joins and leaves. Keeping a consistent view of the cluster requires a mechanism to maintain an up-to-date CML in each node, and of course this property can be satisfied only with a certain amount of overhead. The key property of efficient cluster management in such a dynamic environment is to be light-weight. To minimize cluster-wide negotiation and overhead, QuickSilver does not try to enforce a globally consistent view on the cluster. Rather, each node forms its own view of its cluster and maintains a CML of those it believes to be members of its cluster.

Using the inter-cluster radio interface, the *cluster contact* looks for the short-lived contacts with other clusters traveling in different directions. The ultimate goal of QuickSilver in this respect is to build links across the two clusters that support to the formation of a *fat pipe* between the two clusters. This pipe is opportunistically established to maximize the data exchanged between the two clusters. Independently from which nodes are transmitting, the goal is to establish the largest pipe between the clusters and maximize its utilization, using non-interfering ad hoc links between pairs of nodes belonging to the two clusters.

Finally, the *networking* layer implements the multi-hop delivery for both intra-cluster and the inter-cluster communication. The former only uses links established through the single-channel interface, while the latter uses both interfaces in the data path between content providers and consumers.

3.2 Cluster Management

The construction and maintenance of CMLs in QuickSilver is a two-step procedure: first, a node identifies its stable neighbors, then this local topology is distributed so that each node can update its CML with cluster members that are more than one hop apart. A downside of this approach is that nodes in the same cluster might temporarily have different views of the cluster memberships (i.e., incomplete lists or stale information, caused by recent changes in the cluster topology), and the system must be robust to such inconsistencies. However, given the opportunistic nature of communication in VANETs, such inconsistencies should have little or no effect.

Identifying stable neighbors

Neighbor discovery in opportunistic networks is generally supported by beacons or simple keep-alive message. QuickSilver takes advantage of these existing neighbor discovery mechanisms for evaluating the stability of a connection to a given neighbor. Each node broadcasts `hello` messages with a fixed period T_h. These messages contain the source node ID, and a sequence number, incremented for each new message. If a node receives th_{join} sequential `hello` messages from a neighbor, it marks that neighbor as stable. If th_{leave} periods expire without receiving a `hello` message from a neighbor, the neighbor is removed from the neighborhood table. The identification of these one-hop stable links is central to the notion of clusters in QuickSilver.

Building the Cluster Membership List (CML)

Any node that becomes a stable neighbor of a node is immediately inserted into the node's CML. To identify cluster members that are multiple hops away, each node periodically broadcasts a `c_hello` message to all of its stable neighbors. The period T_c is larger than T_h. Relaying nodes attach their IDs to the message and forward it over all stable links. Any node receiving a `c_hello` message immediately puts the original sender of the message and all the nodes that relayed the message in their CML.

Since intra-cluster routing requires the use of broadcast messages, which will be describe in Section 3.4, QuickSilver leverages these broadcast messages to enable better and more up-to-date cluster management. When a node initiates one of these broadcasts, the message is also used as a `c_hello`, and the relative timer is reset. The relaying nodes piggyback their IDs onto the control message but their timers are unaffected. It is important to note that the use of adding node IDs to every broadcast message can be expensive in large networks. However, these techniques are only used within clusters, which are expected to be relatively small in size, and so will not incur as much overhead as if used network-wide.

While stable neighbors are always part of the CML, destabilization of a stable neighbor does not immediately lead to its removal from the list. The reason is that the node does not know if the single-hop link, which has now become unstable, was the only stable path to that node. A node is removed from the CML only if more than th_{c_leave} cluster advertisement periods T_c pass after its latest insertion in the set (using either `c_hello` or any cluster-wide broadcast message).

A simple analysis gives us an upper bound for the time it takes nodes in a cluster to reach a consistent view on cluster membership (i.e., the CML on each node is identical). When a node joins a cluster, that node must have at least one stable link with a neighbor in the cluster, resulting in the insertion of the new node in the CML of its one-hop neighbors. When the proper timer on the node expires, a `c_hello` message containing the new node ID is propagated throughout the cluster and each cluster member updates their CML accordingly. Assuming that MAC retransmission policies and the redundancy introduced by multiple routes cancels the impact of packet loss and that the time lapse from when a node creates a message and one of its neighbors receives it is T_{tx}, the largest time required for a new node to be present in all cluster members' CMLs is:

$$T^{max}_{cons\_join} = (th_{join} \cdot T_h) + T_c + T_{tx} \cdot \max(n_{hops}), \quad (1)$$

where the first term captures the time necessary to stabilize the local link, the second is the maximum time necessary for the `c_hello` timer to expire after the link is stabilized (in the worst case, the times could have fired right before the new link is stabilized), and the third one is the propagation time necessary for the `c_hello` message to travel to the farthest node in the cluster. Similarly, the maximum time necessary to reach a consistent view after a node leaves the cluster is simply the timeout of a CML entry, which is:

$$T^{max}_{cons\_leave} = T_c \cdot th_{leave}. \quad (2)$$

Given these limits, to achieve a consistent view of the cluster, T_c, T_h, th_{join} and th_leave must be chosen accordingly to the cluster dynamics. The right tradeoff between system responsiveness and overhead must be found. Additionally, it should be noted that a too short T_h or a too low th_{join} would cause short-lived links to be erroneously considered stable. This is a worst case analysis, and in most cases a consistent view might be reached with less strict requirements. However, although QuickSilver's performance would benefit from a consistent view, the system design is robust to inconsistencies, and the timer values and thresholds should be chosen trying to limit the control overhead they generate.

3.3 Cluster contacts

Although there are mechanisms that can be used to establish contention free communication over shared channels, such as TDMA,

they incur significant overhead due to synchronization and coordination. The mobility of clusters poses an additional obstacle to centralized, coordinated solutions: the connection between clusters can be maintained for the whole T_{c_cont} period, but the single links that keep it alive change constantly as nodes move and change neighbors. For this reason, QuickSilver adopts a technique that is completely distributed and opportunistic, an alternative that does not require control overhead.

The fat pipe between two clusters is built on N_c links, where N_c is the number of available orthogonal channels, excluding the one used for intra-cluster communication. Each link is locally used by a single pair of nodes, one per contention area in a cluster, so that no contention is necessary and the channel utilization is maximized, as long as the nodes' queues are not empty. These nodes are *gateways* through which the traffic from their cluster members can be forwarded to the other cluster. Once they reserve the channel they are the only nodes authorized to transmit on that channel in their coverage range. As shown in Figure 3(a), if $N_c = 1$ (i.e., only one channel is assigned to inter-cluster communication), when two clusters are traveling in opposite directions like on an highway, if they overlap for a length d, a number of links $N_L^S \in [\lceil d/R \rceil, \lceil d/2R \rceil]$ links can be established. When two clusters meet at a crossing, as shown in Figure 3(b), $d < R$, thus only one contention free link can be established. This is equivalent to having single hop clusters where only the cluster head is authorized to perform inter-cluster communication [3, 14]. If $N_c > 1$, the number of possible links becomes $N_L = N_c \cdot N_L^s$ for the highway scenario and $N_L = N_c$ in the crossing scenario. Building the single links involves two challenges: first nodes must detect a contact opportunity, then they must select an available channel and become its owners.

In all mobile scenarios, where contacts can be short lived, delay in discovery reduces the contact utilization enough to degrade the performance. In general, two factors influence the delay in discovering a contact. The first one is energy savings mechanisms that force the radio to sleep for long periods of time. In this case, even if two nodes are in communication range, they must wait until both radios are on at the same time to initiate discovery. Fortunately, QuickSilver does not require an aggressive power-saving mode, given that the energy consumed by the radio is negligible compared to the rest of the system when the vehicle is running. However, QuickSilver suffers from another source of delay in contact discovery: the presence of multiple channels. For a successful discovery, two nodes must be tuned on the same channel at the same time. One approach to meet this requirement is the agreement on a specific inter-cluster control channel as in [3], where all nodes not currently part of an active link are advertising their presence. When two nodes meet on the control channel, they exchange control packets to agree on a different channel on which they should both switch to and communicate. This approach causes the waste of one channel for control packets. Additionally, unless an omniscient cluster head has full knowledge of which nodes are using which channels, as well as their interference range, there can be no guarantee that the negotiation phase brings the two nodes to a free channel. This challenge is amplified by the multi-hop nature of clusters in QuickSilver.

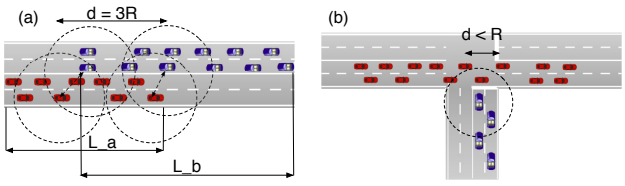


Figure 3: Single-channel connection for inter-cluster contacts.

QuickSilver takes a completely distributed approach that avoids wasting one channel for control traffic. The principle on which QuickSilver is based is very simple: each node channel hops following a randomized permutation of the N_c channels, sending a `disc` message if the channel is free. The message contains the node's CML. If a node from a different cluster is on the same channel, it responds to the message and the two nodes form a inter-cluster link. Since consistent cluster IDs are not possible in QuickSilver's lightweight clustering, nodes determine if they belong to different clusters by comparing their CMLs and verifying if they differ for a fraction larger than $th_{CML} \in [0; 1]$.

The randomized channel hopping sequence must be carefully chosen to guarantee that two nodes will meet on the same channel at least once within a bounded time, and that they will meet on a free channel, if any is available. QuickSilver's sequence generation algorithm is inspired by SSCH [20], where each node selects a seed $s \in [1, N_c]$ and a starting channel x_0. When N_c is a prime, the i-th element of the hopping sequence is: $x_i = (x_{i-1} + s) \bmod N_c$. Two nodes with a different s are guaranteed to meet on one of the channels in at most N_c hops. However, if the channel is busy the nodes stay silent, thus they do not discover each other despite being on the same channel. QuickSilver introduces a novel sequence algorithm:

$$x_i = ((x_{j-1} + s) \bmod N_c) + k \bmod N_c, \qquad (3)$$

where $j = i \bmod N_c$ and $k = \left\lfloor \frac{i}{N_c} \right\rfloor$. This algorithm guarantees that two nodes with a different s will meet at least once on *each channel* after N_c^2 hops, as shown in Figure 4, so that they can establish a link unless all channels are busy. If N_c is not prime, the same property can be achieved by choosing a prime number $M > N_c$ and modifying the hopping sequence using a modulus reduction:

$$x_i = ((x_{j-1} + s) \bmod M) \bmod N_c + k \bmod N_c, \qquad (4)$$

where $j = i \bmod M$, $k = \lfloor \frac{i}{M} \rfloor$, and the number of hops required for a pair of nodes to meet on each channel becomes $M \cdot N_c$.

If two nodes have the same seed, and they start from a different channel, they will follow each other without ever meeting. To prevent this from happening, SSCH introduces parity slots, but unfortunately this requires synchronization among nodes to guarantee that the parity slots happen at the same time, which would be impractical in a large distributed vehicular network. Since the goal during an inter-cluster contact is just to build a link with any node from the other cluster, missing a contact with a specific node would not harm QuickSilver's performance as long as there are other nodes available in the same coverage range. Nonetheless, if after $2M \cdot N_c$ ($2N_c^2$ when N_c is prime) hops no contact is detected although at least one channel was found free, the node randomly selects a new s, and continues the discovery procedure.

3.4 Networking

The Networking layer of QuickSilver includes two components. The first one manages intra-cluster unicast or broadcast communication with peers in the same cluster (see Figure 2). The second component manages the inter-cluster communication and relies on both Cluster management and Cluster Contact.

Intra-cluster networking

Since a node's knowledge of the cluster is locally defined by its CML, intra-cluster unicast packets can only be sent to nodes that

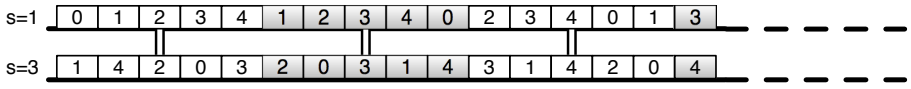


Figure 4: Channel hopping sequence for two nodes, as long as s is different the nodes will meet on every channel after N_c^2 hps.

are currently in the CML of the sender node. The delivery happens using a traditional source routing approach, such as DSR. When a node generates a new message, it initiates an attempt to discover a route to the destination by flooding a RREQ message within its cluster. Relaying nodes attach their IDs to the message and propagate the broadcast. When the destination receives the message and unicasts a RREP back to the source following the reverse path through which the first RREQ reached it. The source node can then unicast the data message to the destination also using source routing. If the source node does not receive a reply in a sufficient amount of time, it assumes that the destination is not part of its cluster. To reduce the overhead of issuing a RREQ message after the creation of every new message, QuickSilver aggressively uses a route cache. To facilitate building up this cache, all messages, both control and data, are expected to contain the list of nodes traversed. Therefore, whenever a node receives any message, it can use the path contained in that message to build up routes to all of the nodes that the message has traversed. To ensure freshness, routes are timed out from the cache every after T_{route} seconds.

The scope of intra-cluster broadcast messages is also limited to cluster members. The broadcast is best-effort and follows a simple flooding algorithm, with the exception that a message is accepted and rebroadcast only if it comes from a node that has been marked as a stable neighbor. To avoid congestion, each node retransmits a broadcast packet exactly once, even if the same message is received from multiple links. A log of the broadcast messages that the node most recently relayed insures this property.

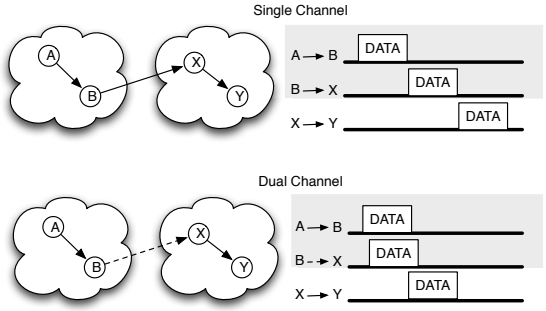


Figure 5: Different channels for intra and inter-cluster communication mitigate the delay for multi-hop inter-cluster delivery.

Inter-cluster networking

When any two nodes from different clusters meet, inter-cluster communication can be initiated by setting up the fat pipe described in Section 3.2. All inter-cluster communication is multicast based on a publish-subscribe mechanism. Nodes that are interested in receiving a certain type of data (e.g., map updates, music) advertise in the `c_hello` messages. A node stores information about its cluster member's interests in a list similar to the CML. When two nodes establish an inter-cluster link, they exchange their interest lists and flood them in their cluster.

In this way, each node is informed of which cluster members are gateways, and what kind of data they are trying to relay to the other cluster. When the inter-cluster link stops working, the gateway informs the cluster, so that the multicast stream is stopped or rerouted to a different gateway. It could be argued that if the links are between nodes that are not storing information but simply act as gateways for other nodes, the inter-cluster links might be underutilized while the gateways receive the data for the other cluster. The choice of using a separate channel for intra-cluster communication, as shown in Figure 5, mitigates this problem. Consider a single link between two clusters. As shown in the single channel scenario, in order to send a packet from node A to node Y, the packet has to traverse three links, and only one of them can be active at the same time to avoid collision. If instead we differentiate the channels, as in the dual channel scenario, node B can initiate transmitting the data as soon as it receives the first packet. Of course, node X might still have to wait to deliver the inter-cluster traffic to the destination, but since they are in the same cluster, this can be done at a later time. The only link that is short lived in this scenario is that between nodes B and X and QuickSilver concentrates on ensuring that that link is never underutilized.

4. EVALUATION

The two main components of QuickSilver are its distributed, light-weight clustering protocol and its inter-cluster multi-link connections. These two components coexist and cooperate to efficiently support intra-cluster and inter-cluster traffic. Since the multi-radio architecture and the choice of using separate channels for the two paradigms implies that there is no harmful interaction between the two components, we evaluated the components separately.

The goal of cluster management is to locally build a CML without the need for a cluster coordinator. However, the fully distributed nature of QuickSilver comes at a price: when clusters are very dynamic, a node's CML might not be accurate and the CMLs might differ from one node to another. When a node enters a cluster, the lack of an explicit join mechanism implies that for some time the cluster members are not aware of the new node and cannot communicate with it. We refer to this situation as a *false negative*. In the same way, when a node leaves a cluster for a transitory time its ID remains in the CML of the other nodes, which might still try to communicate with it, wasting resources. This is a *false positive*.

To evaluate the effectiveness of QuickSilver's cluster management, we tested it in the most unclustered environment: mobile nodes following a random way point model. Essentially, the RWP mobility model causes a very unstable clustering compared to that expected when mobility is constrained to road configurations. These evaluations show the worst-case performance of QuickSilver. However, QuickSilver can take advantage of existing traffic to improve its accuracy (i.e., when a `RREQ` to a node currently in the CML fails, the node can be removed even if the timeout for its entry did not expire yet). We evaluated the clustering mechanism using two metrics: the ratio between false positives and the total number of nodes in the CML per each node (FPr), and the ratio between false negatives and the total number of nodes in the cluster (FNr). The two metrics are computed per node at intervals equal to the minimum duration of a stable link, and for each run the result is an average of the metrics for all nodes over the total simulation time. A high FPr wastes resources trying to reach nodes that are

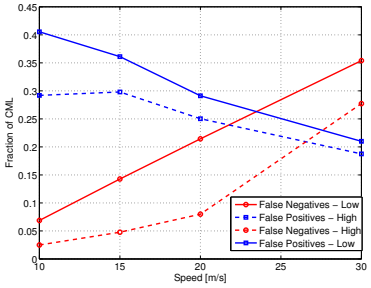


Figure 6: Average False Positives ratio (FPr) and False Negatives ratio (FNr) for different node density and average speed.

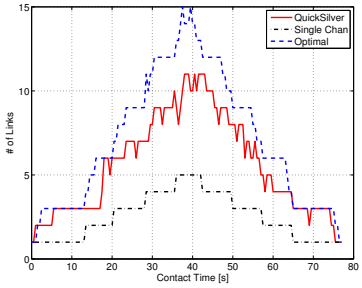


Figure 7: Number of inter-cluster links active during a contact, compared to the optimal and the single channel solutions.

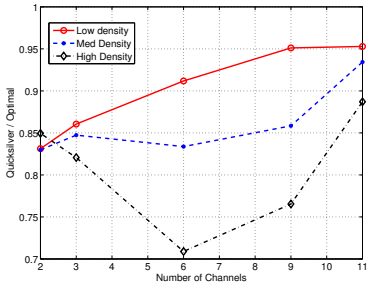


Figure 8: Comparison between QuickSilver and the optimal solution for a cluster contact with different node density.

no longer in the cluster. However, when real traffic is present, the lack of response to a route request can help to limit this problem. An high FNr means an increased delay after a node joins a cluster before every other node can initiate a peer-to-peer connection to it. All evaluations were performed using the ONE simulator [21] in a random mobility scenario in a $3 \times 0.2Km$ area with two levels of density (60 and 100 nodes) and different average speeds.

FPr and FNr respond differently to changes in node speed (see Figure 6): FNr increases with increased speed while FPr is lower at higher speeds. A higher node density always reduces the value of both metrics, because nodes are on average closer and partitioning is less frequent. The reason for this behavior can be explained with a simple example: consider a stable cluster with 10 nodes that suddenly splits in half. Each node will have a false positive ratio of 0.5, which is also the cluster's average FPr until the cluster stabilizes. If instead 8 of the nodes stay together and two generate a smaller cluster, the average FPr becomes $\left(2 \cdot \frac{8}{9} + 8 \cdot \frac{2}{9}\right) = 0,3\overline{5}$. At higher speeds, in the RWP mobility model, this second scenario is more likely to happen due to the high mobility. The tendency of the lower mobility scenario to have smaller clusters instead of a single large cluster and a few outliers also affects FPr. When two clusters merge in the high mobility scenario, most of the network is affected, while when the network is partitioned in small clusters, only a few nodes must update their CMLs. The RWP scenario is more representative of an urban environment, in which low speed and high density are common factors, and in this case the low FNr is desirable, while the `RREQ` mechanism mitigates the effect of a high FPr. In high mobility scenarios, which for VANETs generally means in highways in which the duration of a cluster membership is longer, the large initial delay caused by a high FNr has less influence. We verified, but omit for space constraints, that when cluster membership becomes stable (i.e, no more nodes join or leave the cluster), (1) and (2) represent the convergence times after which FNr and FPr become 0 for all nodes.

On the second radio, nodes hop through the available orthogonal channels trying to establish a link with a node from a different cluster. For this component, QuickSilver's approach is also completely distributed, and the number of contention free links that can be established affects the total amount of data that can be transferred over an inter-cluster connection. Solutions such as [3], in which single-hop clusters are formed around a cluster coordinator, which is also the only node entitled to inter-cluster communication on a single channel, represent a lower bound for QuickSilver. Ideally, an omniscient protocol with perfect knowledge of the position of each node, can determine an optimal configuration that maximizes the number of contention free links, but at a cost in terms of computation and control overhead that cannot be sustained in dynamic environments such as VANETs. We compared our distributed solution with these two extremes to evaluate its efficiency. It is important to note that QuickSilver is completely opportunistic and does not require any overhead. For these evaluations, we implemented the inter-cluster gateway management protocols in MATLAB.

To capture an interesting scenario, we focus on the number of contention free channels found by QuickSilver, the single-channel and the optimal approaches for a contact between two clusters traveling in opposite directions (see Figure 7). The clusters are both 1000 meters long, each composed of 30 nodes. The relative distance between a consecutive pair of nodes is modeled with a normal distribution centered around 33 meters, the speed of both clusters is $15m/s$. For this evaluation, we used 5 orthogonal channels. At $t = 0s$, only the edges of the two clusters overlap. In this configuration, there is only one pair of nodes in the reciprocal coverage range, and the three algorithms tie, with only one link possible. As the clusters keep moving and the overlapping region increases with time, the gap between the optimal solution and the single channel increases. Both solutions respect the bounds in Section 3.3. QuickSilver's performance sits in between the two lines. QuickSilver always takes advantage of the multiple orthogonal channels, performing in some cases as well as the optimal solution, and in no situation less than 70% of it. The sub-optimal performance of QuickSilver can be explained with two factors. First, of course, QuickSilver is a completely randomized solution, and does not have enough information to try to compute the optimal configuration. Additionally, once a link is established, QuickSilver does not drop it until the distance between the two nodes is larger than the radio coverage range. This means that even when QuickSilver reaches a configuration close to the optimal, the continued mobility soon brings the nodes to a different geographic distribution for which the current channel assignment is not optimal anymore. The optimal approach we compare against is computed for each instant, without trying to preserve the existing links.

More orthogonal channels increase the possibility of having multiple concurrent links in the same collision domain. The number of available orthogonal channels is defined by the different technologies and range from 3 for 802.11g to 16 or more for 802.11a or 802.15.4. However, this is not the only factor that influences the performance of QuickSilver. We studied the effect of channel assignment and node density by repeating the simulations for the inter-cluster contacts with low, medium and high density clusters (i.e., nodes are spaced by 65, 50 and 33 meters respectively). The ratio between the number of links established by QuickSilver and by the optimal solution is shown in Figure 8 as a function of the number of available channels. Interestingly, in the low density scenarios, a larger number of channels improves QuickSilver's per-

formance, from 84% of the optimal with two channels up to 95% with 9 channels. For the high density scenario, QuickSilver's performance does not keep up with the optimal scenario between 2 and 6 channels, where QuickSilver tendency to preserve existing links causes a sub-optimal configuration, in which more nodes are available but cannot establish a link because they would cause interference with the existing ones. This is a measure of how close QuickSilver performs with respect to the optimal solutions, and does not mean that the bandwidth achievable with six channels is less than that achievable with two. The results simply show that the randomized approach is less close to optimal. When the number of available channels increases again, the larger number of options helps QuickSilver to recover and approach again to the optimal solution. Although QuickSilver always uses all but one available channels for inter-cluster links, this result might suggest that better resource allocation is possible if the system is aware of nodes' density and speed. The study of the possible benefits achievable with a more complex resource management and the cost in terms of overhead to achieve it in in our research agenda.

5. CONCLUSION

The characteristics and requirements of applications in vehicular networks are unique and so need specifically designed protocols to unleash their potential. In this paper, we presented the design of QuickSilver, a framework that leverages clustering and data patterns typical of vehicular networks to enhance the quality of communication. In particular, QuickSilver uses a lightweight distributed clustering protocol to integrate a traditional source routing protocol for intra-cluster node-centric communication and the construction of a multi-channel link for contention-free inter-cluster data-centric communication. The limited bandwidth and short contacts typical of VANETs call for a careful resource allocation between the two communication paradigms to prevent harmful contention and wasted resources.

The benefits of using QuickSilver come from the use of a common framework for an uncoordinated set of protocols. First, both the inter-cluster and the intra-cluster benefit from the same cluster detection mechanism and avoid redundant control overhead. Second, the framework can coordinate the utilization of the available resources, preventing starvation or harmful competition. Finally, the system can react to the network dynamics and coordinate its various parts to maximize the service provided to the single node and the whole cluster. For example, when a cluster contact happens, each node running QuickSilver makes local decisions but is aware of its neighbors' behavior to establish the largest possible number of contention-free channels between the two clusters.

Although QuickSilver is able to create inter-cluster links between two clusters , more research must be done in the networking component and the resource management. We are currently working on an implementation of QuickSilver in our vehicular testbed. We are also interested in investigating different distributed algorithms to elect which nodes should act as gateways for the inter-cluster communication, to maximize the data exchange during a cluster contact.

[1] A. Skordylis and N. Trigoni, "Delay-bounded routing in vehicular ad-hoc networks," in *Proc. of ACM MobiHoc*, 2008.

[2] V. Namboodiri, M. Agarwal, and L. Gao, "A study on the feasibility of mobile gateways for vehicular ad-hoc networks," in *Proc. of ACM VANET*, 2004.

[3] Z. Zhang, "Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges," *IEEE Communications Surveys Tutorials*, vol. 8, no. 1, 2006.

[4] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proc. of ACM WDTN '05*, 2005.

[5] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, "CarTel: a distributed mobile sensor computing system," *Proc of ACM SenSys*, 2006.

[6] K. Ibrahim and M. C. Weigle, "CASCADE: Cluster-Based Accurate Syntactic Compression of Aggregated Data in VANETs," in *Proc. of IEEE Globecom Workshops*, 2008.

[7] M. Piórkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "On clustering phenomenon in mobile partitioned networks," in *Proc. of ACM MobilityModels*, 2008.

[8] J. S. Otto, F. E. Bustamante, and R. A. Berry, "Down the Block and Around the Corner The Impact of Radio Propagation on Inter-vehicle Wireless Communication," *ICDCS*, 2009.

[9] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks," in *Proc. IEEE INFOCOM*, 2006.

[10] N. Thompson, R. Crepaldi, and R. Kravets, "Locus: a location-based data overlay for disruption-tolerant networks," in *Proc. of CHANTS*, 2010.

[11] J. Ott, E. Hyytia, P. Lassila, T. Vaegs, and J. Kangasharju, "Floating content: Information sharing in urban areas," in *Proc. of IEEE PerCom*, 2011.

[12] E. Hyytia, J. Virtamo, P. Lassila, J. Kangasharju, and J. Ott, "When does content float? Characterizing availability of anchored information in opportunistic content sharing," in *Proc. IEEE INFOCOM*, 2011.

[13] J. Wu and H. Li, "A Dominating-Set-Based Routing Scheme in Ad Hoc Wireless Networks," *Telecommunication Systems*, vol. 18, 2001.

[14] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *Proc. of ICC*, vol. 1, 1997.

[15] C. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE JSAC*, vol. 15, no. 7, pp. 1265–1275, 1997.

[16] Y. Gunter, B. Wiegel, and H. P. Grossmann, "Cluster-based Medium Access Scheme for VANETs," in *Proc. of IEEE Intelligent Transportation Systems Conference*, 2007.

[17] D. B. Johnson, D. A. Maltz, and J. Broch, "DSR: the dynamic source routing protocol for multihop wireless ad hoc networks," *Mobile Computing*, vol. 353, 2001.

[18] C. Perkins and E. Belding-Royer, "Ad hoc On-Demand Distance Vector (AODV) Routing," in *Proc. of IEEE WMCSA*, 1999.

[19] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7, 2003.

[20] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks," in *Proc. of MobiCom*, 2004.

[21] A. Keränen, "The ONE Simulator for DTN Protocol Evaluation," in *Proc. ICST SimuTools*, 2009.