CarSpeak: A Content-Centric Network for Autonomous Driving

Swarun Kumar, Lixin Shi, Nabeel Ahmed, Stephanie Gil, Dina Katabi and Daniela Rus
Massachusetts Institute of Technology
{swarun, lixshi, n3ahmed, sgil, dk, rus}@mit.edu

ABSTRACT

This paper introduces CarSpeak, a communication system for autonomous driving. CarSpeak enables a car to query and access sensory information captured by other cars in a manner similar to how it accesses information from its local sensors. CarSpeak adopts a content-centric approach where information objects – i.e., regions along the road – are first class citizens. It names and accesses road regions using a multi-resolution system, which allows it to scale the amount of transmitted data with the available bandwidth. CarSpeak also changes the MAC protocol so that, instead of having nodes contend for the medium, contention is between road regions, and the medium share assigned to any region depends on the number of cars interested in that region.

CarSpeak is implemented in a state-of-the-art autonomous driving system and tested on indoor and outdoor hardware testbeds including an autonomous golf car and 10 iRobot Create robots. In comparison with a baseline that directly uses 802.11, CarSpeak reduces the time for navigating around obstacles by 2.4×, and reduces the probability of a collision due to limited visibility by 14×.

Categories and Subject Descriptors C.2.2 [**Computer Systems Organization**]: Computer-Communications Networks

Keywords Autonomous Vehicles, Content-Centric, Wireless

1. INTRODUCTION

Autonomous vehicles have been the topic of much recent research [5, 31, 18]. The goal of these systems is to drive from point A to point B in an efficient and safe manner, while dealing with continuous changes in the environment due to pedestrian and object movements, and the potential of unexpected events, such as roadwork and accidents. To achieve their goal, autonomous vehicles need detailed realtime information about their surroundings [22]. They typically use laser rangefinder sensors to discover the surfaces of nearby objects and represent this information as a *3D-point cloud* similar to that shown in Fig. 1. Using only the car's on-board sensors, however, prevents autonomous vehicles from uncovering hidden objects that are not directly in their line-of-sight, e.g., a kid running around the corner, or a car pulling out of an occluded driveway. These sensors also cannot deliver long-range data with sufficient accuracy, which limits the car's ability to plan ahead [31]. Further, they are costly (e.g., the sensors alone on an autonomous vehicle can cost several hundred thousand US dollars [17, 18]). For

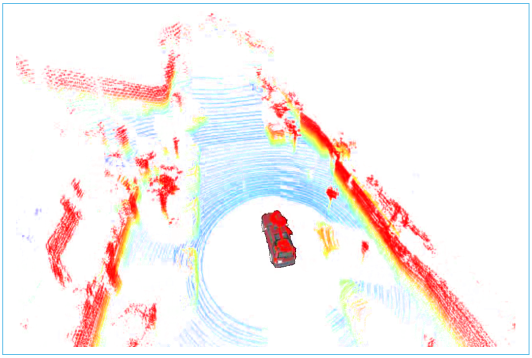

Figure 1—Example of sensory information used in autonomous driving. The figure shows a 3D-point cloud of a road obtained by a Velodyne laser sensor, where colors refer to elevation from the ground. Note that a 3D-point cloud provides the (x, y, z) coordinates of points lying on the surface of obstacles.

these reasons, the report from the recent DARPA Urban Challenge identifies the need for information sharing between autonomous vehicles as a key lesson learned from the contest [3]. However, 802.11 is ill-suited for this application. Navigation sensors can generate realtime streams at Gb/s from each car, leading to a scenario where there is always more data than bandwidth to send it. Furthermore, a communication protocol that cannot capture the importance of different pieces of information for the application will end up inundating the medium with irrelevant or stale data, and potentially denying access to important and urgent information.

This paper introduces CarSpeak, a communication system that addresses the needs of autonomous vehicles. CarSpeak enables cars to request and access sensory information from other cars, as well as static infrastructure sensors, in a manner similar to how they access their own sensory information. To achieve its goal, CarSpeak adopts a content-centric design, where information objects are first class citizens. CarSpeak's information objects are regions in the car's environment (e.g., a cube of 1 m^3). In CarSpeak, a car can request a realtime stream of a 3D-point cloud data from a particular region along the road. It can also zoom in to get a more detailed description, or zoom out for a wider view.

CarSpeak delivers its design via three components that address the main challenges in sharing navigation sensor data:

- *How does a car describe the information it wants, at a particular resolution, if that information describes a region along the road?* In order to name and find road regions, CarSpeak divides the world recursively into cubes; smaller cubes provide a finer description of the encompassing cube. Each cube refers to a region. To efficiently represent this data, CarSpeak uses an Octree, a data structure commonly used in graphics to represent 3D objects [28, 11]. Each node in the Octree refers to a cube, and the sub-tree rooted at that node refers to finer details inside that cube, as shown in Fig. 2. The Octree representation allows CarSpeak to name regions at different resolution and according to their lo-

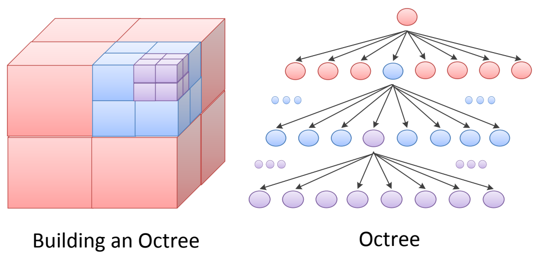

Figure 2—Representation of regions using Octree.

cation in the world. Specifically CarSpeak names a region by referring to the root of the region's sub-tree; it expresses the resolution of the region using the depth from the root of the region's sub-tree. The Octree also enables a car to store its data efficiently because, though the world is huge, each car needs to only expand the part of the Octree in its neighborhood.

- *How does the system allocate the wireless bandwidth to the most recent data from the region, given that multiple cars may sense the same region and each car does not know what information other cars know?*
  CarSpeak adopts a content-centric MAC where information objects, as opposed to senders, contend for medium access. Further, each information object (i.e., 3D-point cloud stream) obtains a share of the medium proportional to the number of requests it receives.
  CarSpeak implements this abstraction using a distributed protocol, where nodes that sense a region contend on its behalf. Requests for region data are broadcast on the medium. Nodes compute a summary value of the quality of the information they have of each region (which is a measure of the timeliness and completeness of this information). CarSpeak uses a low overhead protocol to share this information among the nodes as annotations on their transmitted data packets. Each car uses these annotations to compute how much sensory data it should transmit so that its contribution to each stream is proportional to the completeness and freshness of the data it has from the corresponding region. CarSpeak then enforces this allocation by controlling the 802.11 contention window appropriately.
- *How does the system compress the redundancy in the transmitted sensor data while being resilient to packet loss?*
  CarSpeak makes each packet self-contained by assigning it an independent set of branches in the Octree that are derived from the root. As a result, each received packet can be correctly inserted into the tree independent of other packets. CarSpeak also reduces the overlap between data transmitted by cars that sense the same region. Recall that each region is a cube that encompasses many smaller cubes, whose values keep changing in realtime due to the arrival of new sensor data. In CarSpeak even if multiple cars receive a request for the same region (i.e., the same encompassing cube), each of them will pick a different permutation according to which they transmit the sub-cubes in the region. Thus, if only one car has sensor data about the region, it will eventually transmit all the sub-cubes from the region. However, if multiple cars have data about the same region, then they are likely to cover all sub-cubes in the region, while limiting the overlap in their transmissions.

We built a prototype of CarSpeak in ROS, the Robot OS [26] and integrated it with a state of the art path planner, whose earlier version was used in the DARPA Urban Challenge. We evaluated CarSpeak on two testbeds: 1) an indoor testbed of iRobot Create programmable robots connected to netbooks with Atheros AR9285 cards and gathering sensor data from Xbox 360 Kinects, and 2) an outdoor testbed composed of an autonomous Yamaha G22E golf car mounted with Hokuyo laser range sensors, and exchanging sensory information with the Create robots. We compared CarSpeak with a baseline inter-vehicle communication protocol that directly uses the existing 802.11 protocol.

Experiments from the indoor testbed show that compared to the 802.11 baseline, CarSpeak reduces the time taken to navigate an environment with obstacles by 2.4×, and the probability of a collision due to limited visibility by 14×.

Outdoor experiments with the a Yamaha golf car tests the role of communication in enabling cars to react safely to pedestrians who suddenly exit a blind spot and cross the car's path. Empirical results show that use of CarSpeak allows for the receiver on the golf car to issue a stop command with a maximum average delay of 0.45 seconds which is 4.75× smaller than the *minimum* delay of 2.14 seconds using 802.11. These relatively small delays using CarSpeak allow the vehicle to safely stop before the crosswalk if the pedestrian appears at distances as small as 1.4 meters on average, even when the vehicle is traveling at its maximum velocity of 2 meters per second. In contrast, using 802.11 the vehicle is unable to stop before reaching the crosswalk if the pedestrian appears when the vehicle is closer than four meters from the crosswalk on average.

Contributions: To our knowledge, CarSpeak is the first communication system for multiple autonomous vehicles that focuses on maximizing the utility of information for this application, and that is fully integrated with autonomous vehicle systems. It is evaluated on a testbed of autonomous vehicles, and demonstrated to reduce path length and the probability of collisions. Its content-centric design that operates on realtime rich sensory data sets it apart from past work on VANET. This design is delivered via three components including a multi-resolution naming and addressing scheme, a content-centric MAC, and a new approach to compressing rich sensory data that is suitable for lossy and dynamic environments.

2. RELATED WORK

Recent years have witnessed major advances in designing and building autonomous vehicles to realize safer and more fuel efficient future cars [5, 31, 18]. Past work in this domain [14, 8, 17], including the DARPA Urban Challenge and the Google autonomous car, focuses on issues related to perception, efficient path planning, obstacle detection, etc. In contrast, this paper focuses on designing a communication protocol that is most suitable for sharing sensory data between autonomous vehicles.

Our work is related to a broad area in robotics that studies networks of robots. Past work in this area can be divided into two categories: The first category uses communication as a black-box, and focuses on algorithms that enable robots to collaborate on a desired task, for instance, cooperative exploration [23] or pursuit evasion [15]. The second category considers the application as a black-box and focuses on harvesting robot mobility to improve network connectivity or throughput [24, 9]. In contrast, our work is based on designing the communication protocols around the needs of the application, and takes neither as a black box.

A large number of research papers have focused on the problem of Vehicular ad-hoc networks (VANETs). Work in this area focuses on efficient routing [19, 27, 30], delay tolerant networks [20], reliable delivery of emergency messages [2, 6], or specific applications such as detecting accidents [13]. None of these papers, however, present a content-centric architecture or design a MAC protocol where information objects contend for the medium. Also, none of them present a solution that is particularly suitable for autonomous driving.

Our work builds on past work on content-centric networking.

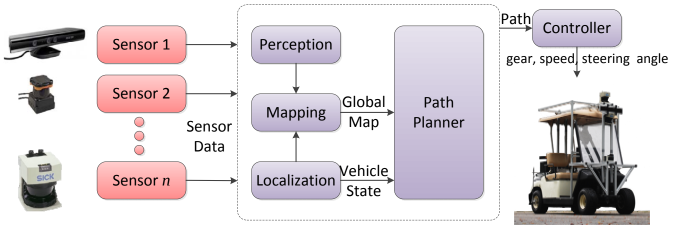

Figure 3—High-Level Architecture of Autonomous Vehicular Systems. The path planner module uses information from various sensors to compute a safe path for the vehicle.

Past work in this domain is mostly focused on the Internet [12, 16]. The few papers that apply this concept in the wireless domain are focused on storage or routing information content [25, 29, 4]. Our work differs from all these papers in that it is focused on resource sharing at the MAC layer. Also, it uses a multi-resolution naming system and is fully integrated with an autonomous driving in terms of design, implementation and evaluation.

3. PRIMER ON AUTONOMOUS VEHICLES

In this section, we provide a quick background of autonomous driving software so that it is clearer how CarSpeak interfaces with these systems. Successful performance of autonomous vehicles relies on their ability to sense and process information about the environment around them. To obtain this information, autonomous vehicles and robots are typically equipped with ranging sensors, which deliver realtime measurements of the distance of the vehicle to the surrounding 3D objects. The vehicle may use laser scanners, ultrasonic range finders for outdoor settings and Kinect for indoor settings [31, 5, 18, 10]. Other sensors like cameras and light detectors are also used for additional information.

Most autonomous vehicles use the Robot Operating System (ROS) framework [26]. ROS provides a publish/subscribe architecture, where a module (e.g., sensor) publishes a topic (e.g., /sensor_data) that can be subscribed to by multiple modules. We discuss the commonly defined high-level modules below (Figure 3):

- **Sensor Infrastructure:** Each sensor attached to the autonomous vehicle has an associated module which converts raw sensor information obtained from the driver into a generic sensor format. The most widely used format is a *3D-point cloud* which provides the 3-D (x, y, z) coordinates of points lying on the surface of obstacles. The point cloud, along with a timestamp t denoting the time of retrieval of sensor data, is published by each sensor module.
- **Planner:** The planner's goal is to use sensory information to plan an obstacle-free path for the vehicle to navigate along. The planner typically has access to a detailed global map of the environment. The planner is sub-divided into four modules:
  - *Perception module* subscribes to point cloud information from the sensors and applies complex obstacle detection algorithms to recognize obstacles in the frame of reference of the vehicle. It publishes a map of these obstacles.
  - *Localization module* publishes the vehicle's position within the global map based on GPS, odometry or more advanced sensory infrastructure, some of which can be as accurate as a few centimeters [18].
  - *Mapper* subscribes to information from the localization and perception modules and publishes a global map incorporated with locations of obstacles.
  - *Path planner* subscribes to the vehicle's location and the global obstacle map and publishes a path for the vehicle to travel along.

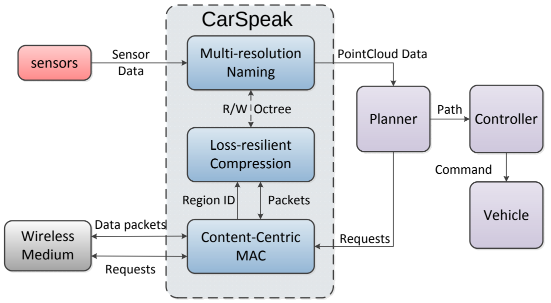

Figure 4—Information Flow in CarSpeak. CarSpeak has three components: A Multi-Resolution Naming and Addressing system, A Loss-Resilient Compression system and a Content-Centric MAC

- **Controller:** The controller subscribes to the vehicle's path and issues steering and velocity-control commands to the vehicle, so that it navigates along the computed path. In many cases, the controller may execute emergency maneuvers if there is substantial change in the obstacle map due to moving obstacles.

4. CARSPEAK'S ARCHITECTURE

CarSpeak's design aims to interface effectively with the ROS (Robotic Operating System) architecture for autonomous vehicles. From the perspective of the ROS planner, CarSpeak looks like a local sensor that streams sensory information obtained from other vehicles and static infrastructure sensors.[1] CarSpeak receives requests from the car's planner by subscribing to /query_region topic. It propagates these requests over the network to CarSpeak modules on other vehicles to direct them to transmit information from the requested region. When it receives sensory data in response to requests it sent, like other sensors, CarSpeak publishes this data as a stream of 3D point-cloud data (under the topic /car_speak). The planner may now subscribe to the information from the requested regions. Unless refreshed, a subscription (as well as the corresponding requests broadcast on the medium) expires after one minute. Timing out subscriptions is done for efficient use of bandwidth as cars are expected to lose interest in some regions and gain interest in others, as they move around.

CarSpeak's guarantees are best effort, i.e. CarSpeak aims to make the best use of the available bandwidth to send as much relevant information as possible, in a loss-resilient manner. CarSpeak has three components: A Content-Centric MAC, a Multi-Resolution Naming and Addressing system, and a Loss-Resilient Compression system. Fig. 4 illustrates how these components interact with each other, the planner, and the wireless channel.

The MAC receives region requests from the planner and broadcasts these requests on the medium. It also keeps track of requests received from other cars over the wireless medium. It evaluates the importance of different regions based on how many requests they have recently received and tries to satisfy these requests by working with the other CarSpeak components.

The multi-resolution naming and addressing system subscribes to 3D-point cloud information published by local sensors and builds an Octree-representation of this data. The Octree is read by the compression module whenever CarSpeak sends data packets in response to outside requests, and is written by the compression module whenever CarSpeak receives data packets in response to requests generated by the car. The multi-resolution naming and ad-

[1]Sharing information requires a notion of trust. One option is to use the IEEE 1609.2 security standard for inter-vehicular networks to digitally sign and verify all messages. However, the details are beyond the scope of this paper.

dressing system also publishes updates to the Octree caused by the arrival of external data as a 3D-point cloud under the topic /car_speak, which is subscribed to by the planning system of the autonomous vehicle.

The loss-resilient compression is triggered by the MAC to generate compressed data packets for transmission on the medium or to decode received data packets and insert them in the Octree.

In the following sections, we discuss the design and functioning of each of these modules in more detail.

5. MULTI-RESOLUTION NAMING

In autonomous vehicles, sensory information is typically represented as a 3D-point cloud. The point-cloud representation however is unstructured, and hence does not facilitate requesting information about a specific region. It is also inefficient because it does not compress the information by leveraging the fact that points close to each in space other tend to have similar properties: empty or occupied. Sharing information across vehicles requires a naming scheme in which a car can name a specific region of interest. It also requires an efficient representation that compresses the exchanged data and reduces bandwidth consumption.

5.1 Information Naming and Representation

CarSpeak uses the *Octree naming system* to identify and represent sensor information from the environment. Specifically, CarSpeak divides the world recursively to cubes. It starts with a known bounding cube that encompasses all points observed by vehicles in the environment. Thus this bounding cube, or root cube, is known and agreed upon by all vehicles . Each cube is then recursively divided into 8 smaller cubes as shown in Figure 5. A cube is set to be either: (1) *occupied*, if the point cloud representation has points within it (i.e., the cube has some object and the car should not drive through it); (2) *unoccupied*, if there are no points within it (i.e., the cube is vacant and the car may drive through it); and (3) *unknown* if there is no sufficient sensory information about it (i.e., the cube may have some object but the car does not yet have sensor data to identify it). We note that a parent is occupied if any of its descendants are occupied. A parent is unoccupied only if all of its descendants are unoccupied. Otherwise the parent is unknown.

CarSpeak maintains this recursive structure in an Octree, where each vertex in the Octree represents a cube, and the sub-tree rooted at that vertex refers to the recursive divisions of the cube. The Octree representation allows CarSpeak to name road regions at different resolutions and according to their locations in the world. Specifically, in CarSpeak a region is an encompassing cube, which is nothing but a sub-tree in the Octree, truncated to L levels, where L is the resolution at which the region is described. In principle, one can allow regions of any size to have any resolution. This would allow a car to request the whole world at the finest resolution. Such a design is both inefficient and unnecessary.

Thus, CarSpeak expresses large regions at coarse resolutions and smaller regions at finer resolutions. Specifically, CarSpeak partitions the Octree into mutually disjoint sub-trees, where each subtree is truncated to L levels. Each of these truncated sub-trees denotes a region and forms a hierarchy, as shown in Fig. 5. All regions are described completely by their corresponding truncated sub-tree, which contain up to $8^L - 1$ vertices, labeled either “occupied”, “unoccupied”, or “unknown”. Regions at a higher-level in the hierarchy provide a zoom-out view and are represented at a coarser spatial granularity, whereas regions at a lower level in the hierarchy provide a zoom-in view and are represented at a finer spatial granularity. A key point to note is that truncated sub-trees corresponding to any pair of regions, regardless of their hierarchy, do not overlap.

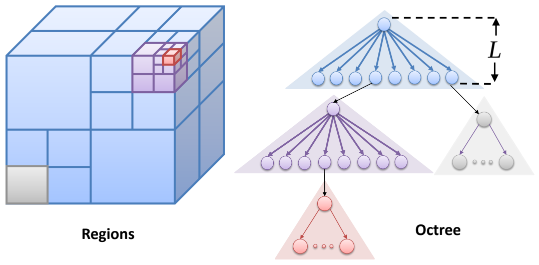

Figure 5—Naming Regions in CarSpeak. The figure on the left shows four different regions at different hierarchies with $L = 2$. The figure on the right depicts the truncated sub-trees corresponding to these regions in the Octree.

To assign regions globally unique names, we number them in ascending order in a top-down, breadth-first manner starting from the region containing the root of the Octree (that provides a coarse view of the entire environment). The number of regions and L are design parameters that can be chosen to fit the specific application.

Benefits: CarSpeak Octree-based naming system has two advantages. First, it facilitates requesting and accessing sensory data at different resolutions. Second, it compresses the data both for storage and transmission. In particular, while the world may be huge, each vertex need not expand regions that are far from its location. Further, when exchanging information, a large subtree can be sent as one vertex if all vertices in the subtree have the same value (e.g., if a whole subtree has occupied vertices, CarSpeak sends a single value expressing the fact that the whole subtree is occupied).

5.2 Information Retrieval and Maintenance

The multi-resolution naming and addressing module subscribes to information from local sensors and incorporates this data into its Octree. It also populates its Octree using sensor data received from other vehicles over the wireless medium. This octree provides information to other nodes that may failed to obtain it from the original sender due to packet loss or disconnectivity. Further, it publishes the data it receives from other vehicles through the topic /car_speak after converting it to the 3D point cloud representation. (Note that data from local sensors is already published by the sensors themselves.)

Garbage Collection. Each vertex of the the Octree datastructure is annotated with a time_stamp field that is set to the timestamp of the most recent sensor information stored in this cube. CarSpeak runs a thread to periodically traverse the Octree and removes vertices which are outdated. If time_stamp is older than the current time by over a fixed threshold, then the entire sub-tree is dropped. We note that in CarSpeak, all cars use NTP to synchronize their time to within tens of milliseconds, which is sufficient for the purpose of the application.

5.3 Information Quality

The 3D stream of a region contains data from multiple nodes that sense that region. On average, for each region, CarSpeak aims to provide each node a share of the medium proportional to the *quality* of information it possesses about that region. Let $Q(i,r)$ denote the quality of information that node i has regarding region r. CarSpeak evaluates this function based on the following metrics:

- *Time*: Sensor information gathered more recently is of greater value, than sensor information gathered in the past. CarSpeak decreases the quality of sensor information $Q(i,r)$ based on delay, by a factor of $\mu^{(t_{curr}-t_{sense})}$; where t_{curr} is the current timestamp,

t_{sense} denotes the timestamp at which sensor information was last obtained from the region and $0 < \mu < 1$ is a constant.

- *Completeness*: The more complete the information a node has about a region, the higher the value of its information. Hence, CarSpeak linearly increases the quality of sensor information based on its completeness, i.e., $Q(i,r) \propto C(i,r)$, where $C(i,r)$ is a measure of completeness of information that node i possesses about region r. The Octree presents an effective metric for measuring $C(i,r)$. Let $C(i,r)$ denote the number of vertices not labeled unknown in the section of the Octree, possessed by node i, representing region r divided by the number of possible vertices in region r. Hence, $C(i,r)$ represents the fraction of region r that node i has sensed.

Thus, CarSpeak's multi-resolution naming and addressing system on node i keeps track of the quality of sensor information, $Q(i,r)$, possessed by its node for each region r, as:

$$Q(i,r) = C(i,r)\mu^{(t_{curr} - t_{sense})}, \qquad (1)$$

where t_{sense}, t_{curr}, and $C(i,r)$ are as defined above and $0 < \mu < 1$ is a constant ($\mu = 0.5$ in our implementation).

6. Loss-Resilient Compression

3D streams of region data are highly redundant. The information at each node is correlated because occupied cubes tend to be co-located. The same applies to unoccupied and unknown cubes. Further, different nodes that sense the same region may have overlapping information (though the overlap is typically not complete due to differences in perspective and viewing distances). Thus, efficient transmission requires compressing the data. A good compression scheme however should deal with the following issues: 1) The compression algorithm should have a low computational overhead to maintain the realtime nature of the data; 2) It should not require the cars to track how their information relate to each other; and 3) It should be resilient to packet loss. We describe how CarSpeak meets these requirements below.

State-of-the-art compression schemes in the graphics community for 3D-point clouds leverage the fact that the corresponding Octrees can be efficiently encoded to provide a compressed high-entropy representation [28, 11]. Similarly, CarSpeak exploits this compression capability of the Octree.

The standard Octree encoding technique is as follows: Let each vertex in the Octree be represented by a tuple of length 8 representing the occupancy of each of its children. We traverse the tree in a top-down and breadth-first fashion and read off the corresponding tuples to obtain the compressed data. The root vertex can always be assumed to be occupied. Then, for each vertex, its occupied and unknown children are recursively encoded. In contrast, no further information needs to be encoded for unoccupied vertices (as they are assumed to be entirely unoccupied). Additionally, vertices whose descendants are entirely occupied or unknown are terminated by a special 8-tuple (with all entries marked occupied or unknown respectively) and are not further encoded, as shown in Fig. 6. The decoder can then faithfully recover the Octree by following the same traversal rules as the encoder.

This algorithm is very efficient since it simply walks the Octree representation. The problem however with using traditional Octree compression is that the loss of even a single byte in the Octree representation affects the correctness of all data following that byte. Consequently, Octree based compression frameworks are highly sensitive to packet loss. Furthermore, the above algorithm cannot deal with data overlap across nodes that sense the same region.

We introduce a new algorithm that leverages the above standard Octree compression, but deals with packet loss and overlap. Specifically: Let r be the region from which a packet is requested by the Content-Centric MAC. Let $\mathcal{O}_r$ denote the truncated sub-tree in the Octree corresponding to this region. Let $v_0, v_1, \ldots, v_{K-1}$ denote the leaf vertices of this sub-tree. The algorithm to generate a packet proceeds as follows: Let variable j be set to a random integer between 0 and $K-1$, and variable $\mathcal{T}$ be set to an empty tree.

(1) Add vertex v_j to the tree $\mathcal{T}$ as well as all its ancestors tracing all the way up to the root of $\mathcal{O}_r$.
(2) Set $j \leftarrow j+1$ and repeat step (1), while the encoding of $\mathcal{T}$ fits within the maximum size of a packet.

The algorithm writes the Octree encoding of $\mathcal{T}$ to a packet and transmits it. Note that all packets are not generated in one shot, i.e. even if two packets are describing the same region, the one with the more recent time stamp will have more current information. As nodes pick different random seeds, it is highly probable that different nodes that sense the same region do not transmit overlapping information, at any point in time.

The compression algorithm can be further improved by using predictive coding techniques [28, 11] as the sub-graph $\mathcal{T}$ preserves edges between parents and children in the original Octree $\mathcal{O}_r$ (Our current implementation however does not use predictive coding).

The resulting compression algorithm has the following desirable properties: 1) it is computationally efficient because it is linear in the size of the region's Octree; 2) it is robust to losses because each packet is self-contained; 3) packets received from different cars that sense the same region have minimal overlap because at any point in time, they transmit different parts of the region's Octree; and 4) it supports a form of unequal error protection because the transmitted subtrees contain paths of all vertices to the root of the region. Hence, vertices at higher levels in the tree are less likely to be lost than vertices at lower levels. Thus, the loss of a packet typically results in the loss of resolution as opposed to complete loss of information.

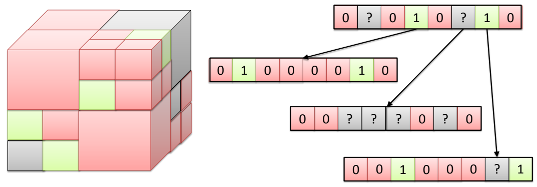

Figure 6—Octree Representation Representation of 3D-point cloud using a 2-level Octree. Vertices that are unoccupied are not expanded; vertices that are completely occupied or unknown are terminated by special leaf vertices; these have been omitted for clarity.

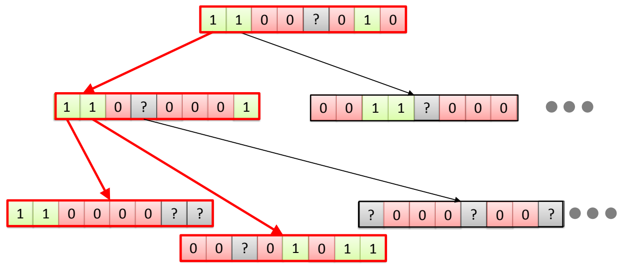

Figure 7—Picking tree $\mathcal{T}$ for compression Tree $\mathcal{T}$ picked with j initialized to 0 is highlighted by red contours.

7. Content Centric MAC

CarSpeak adopts a content-centric MAC that focuses on the application's goals and requirements. In CarSpeak, regions, as opposed to senders, contend for the medium. The MAC tracks region requests and allocates to each region a medium share proportional

Symbol	Meaning
S_i	Node i's desirable share of the medium
$CW_{min,i}$	Contention window of node i
R_r	Region r's desirable share of the medium
$P_{i,r}$	Probability of transmitting packets from region r at node i
$Q(i,r)$	The quality-function at node i for region r

Table 1— Table of Notations. We use i to iterate over nodes, and r to iterate over regions

to how often it is requested. Further, it ensures that the number of transmissions each node makes on behalf of a region is proportional to the quality of information that the node has about the region as measured by the function $Q(i,r)$. Below we describe how the MAC performs these functions.

7.1 Tracking Region Requests

The content-centric MAC handles requests for different regions both from its own vehicle and other vehicles. The module records REQ_s, a measure of requests made for each region s by various nodes. In our implementation, REQ_s is set to one plus the number of requests made for region s. This is to ensure that in the absence of requests all regions get equal share of the medium.

Internal Requests: When the MAC receives requests from its own car, it broadcasts them to other vehicles over the wireless medium. It also keeps track of past requests and times them out after a minute. When packets with region data arrive, the module checks whether they answer a request that has not timed out, in which case it passes them to the compression module for decoding.

External Requests: The MAC actively listens on the medium to track the requests made for various regions, and to identify which regions are observed by vehicles in the network. When a vehicle receives a request for region s, it updates REQ_s accordingly, which biases it to transmit more information about region s, if available.

7.2 Region Contention

CarSpeak aims to share the medium among regions proportionally to the number of requests they receive.

(a) Sharing the Medium Among Regions. Let R_r be region's r share of the wireless medium, i.e., the percentage of transmissions that should describe region r. We can write:

$$R_r = \frac{\text{pos}(\sum_i Q(i,r))REQ_r}{\sum_s \text{pos}(\sum_i Q(i,s))REQ_s} \tag{2}$$

where REQ_s is a measure of requests made for region s, and $\text{pos}(x) = 1$ if $x > 0$ and 0 otherwise. The function $\text{pos}(\sum_i Q(i,r))$ ensures that only regions for which some node has information acquire a share of the medium. Regions that no node has sensed (i.e., $Q(i,r) = 0, \forall i$) do not get a share of the medium.

But how does a node obtain the information it needs to substitute in the above equation in order to compute R_r? CarSpeak disseminates this information as annotation on the data packets transmitted by each node. Specifically, every CarSpeak packet sent by node j includes a list of region ids for which node j has information and their corresponding $Q(j,r)$'s. By default this list has 5 entries for a total of 40 bytes (6 bytes for region ids and 2 bytes for $Q(j,r)$).

CarSpeak nodes listen on the medium and collects information about the different regions and quality of information that other nodes have for these regions. They use this information to populate a table of region ids, and the quality of information the various nodes have for each region. A garbage collection thread that runs every 10 seconds multiplies $Q(i,r)$ values by a factor μ, $(0 < \mu < 1)$ in order to age-out quality information that is outdated it also timeout requests that have not been refreshed in the past minute.

(b) Controlling Medium Access. Using its estimate of the share of the various regions of the medium, R_r's, a node can estimate how often it should transmit, i.e., its own share of the medium. Let S_i be the medium share of node i. Node i's share of the medium is the sum of its contribution to the transmissions related to all regions for which i has data. This contribution is also proportional to the quality of information the node has about each of these regions. Thus:

$$S_i = \sum_r R_r \frac{Q(i,r)}{\sum_j Q(j,r)}. \tag{3}$$

Conceptually, once a node knows its share of the wireless medium, it should be able to transmit according to that share. At first, it seems that the node can achieve this goal by simply waiting for a transmission opportunity – i.e., the medium being idle – and using such opportunities as often as its share permits. For example, if its share is 20% of the medium time, it then transmits once every five times it senses the medium to be idle. Unfortunately, this approach does not work in practice. In practice, the decision to transmit upon the detection of an idle medium is performed in the card itself and cannot be controlled in software.

Thus, we will enforce the node share indirectly by controlling its contention window CW_{min}. The relation between the contention window and the resulting share of the wireless medium is given in [1] as:

$$CW_{\text{min},i} = \frac{2 - S_i}{S_i}. \tag{4}$$

The above relation is derived from a detailed Markov chain model of the progression of the contention window in 802.11 [1]. Intuitively, however, one can understand it as follows: In 802.11 a node picks a random value between 0 and CW_{min}. Thus, the average contention value is $\frac{CW_{min}+1}{2}$. Thus, on average the node accesses the medium once every $\frac{CW_{min}+1}{2}$, and hence its share of the medium $S_i = \frac{2}{1+CW_{min}}$.

(c) Partitioning a Node's Transmissions Among Regions. While the above ensures that the node gets the proper share of the medium, the node still has to divide this share between various regions depending on: 1) each region's share of the medium, and 2) the quality of information the node has about the region. To achieve this goal, whenever the node has an opportunity to transmit a packet, it picks the packet from region r with the following probability:

$$P_{i,r} = \frac{R_r \times \frac{Q(i,r)}{\sum_j Q(j,r)}}{\sum_s R_s \times \frac{Q(i,s)}{\sum_j Q(j,s)}} = \frac{R_r}{S_i} \times \frac{Q(i,r)}{\sum_j Q(j,r)} \tag{5}$$

Clearly $\sum_r P_{i,r} = 1$, for every wireless node i.

The above is implemented using a non-blocking UDP socket. Whenever the socket has space for new packets, the node picks those packets from the regions according to the probabilities $P_{i,r}$'s.

7.3 Scaling

The above design has an important side benefit: it provides congestion control for 802.11 broadcast mode. Specifically, the presence of many 802.11 senders can lead to excessive collisions and a congestion collapse. This effect is countered in 802.11 unicast mode by the fact that a node that does not receive an ACK for its packet, backs off and doubles its contention window. Hence, during congestion, nodes tend to back off and reduce the number of collisions. In contrast, 802.11 broadcast mode does not have ACKs and hence it cannot use the lack of ACK as a signal of congestion to which it reacts by backing off. This leaves the broadcast mode with no protection against medium congestion. The resulting problem is typically referred to as a broadcast storm [19, 30]. In contrast, CarSpeak scales with a large number of senders because senders do not contend for the medium. It also scales with a large number of regions because as the number of regions increases the share of each region decreases because R_r depends on a region's share of the total number of requests.

8. DISCUSSION

In this section, we discuss some design considerations in implementing CarSpeak:

Communicating Processed Information. An important design decision is whether CarSpeak nodes should send processed sensor information, such as locations of pedestrians or whether a road is congested, instead of raw sensor information. While this approach may be sufficient for specialized scenarios, they are not suitable for general-purpose communication between autonomous vehicles. In the most general applications, transmitting nodes in networks of autonomous vehicles need not know how receivers plan to process this information. Furthermore, different receivers may process the same sensor information to achieve different objectives. Native sensor information, available at different resolutions, is the only representation generic enough to cater to varied objectives, such as evaluating road congestion, detecting pedestrians, avoiding vehicles, enabling better localization, route planning, and curb-detection amongst others.

One Hop vs. Multi-Hops. One design decision is whether CarSpeak nodes should relay requests, in an attempt to find the relevant information at vehicles that are multiple hops away from the originator. We chose not to do so, i.e., we do not make vehicles forward region requests. Our reasoning is based on the tradeoff between bandwidth consumption and the value of information about relatively distant locations. CarSpeak targets urban environments and speeds lower than 20 miles per hour. For autonomous driving applications, and even with a conservative estimate, a car should not need information from locations that are farther than half to one minute away. At the above speeds, this translates into locations that are 100 to 200 meters away, which are typically within radio range.[2] Hence, we believe that limiting access to only information that is within the radio range of the requester is a reasonable design choice that enables each region to expend its wireless bandwidth on serving its local, and hence most urgent, requests.

Regular Traffic. CarSpeak can support 802.11 traffic unrelated to autonomous driving as well. Such traffic can be represented simply as a virtual region in space. The designer can decide how to weigh this region in comparison to autonomous driving regions. For example, one may want to divide the medium equally between autonomous driving and other applications by setting $R_{virtual} = 0.5$, in which case the autonomous driving application can use half the medium share (as well as any resource unused by the virtual region).

9. IMPLEMENTATION

We implement CarSpeak's multi-resolution naming, addressing, and information sharing system as a module ("ROS node") in the

[2]For example, the Dedicated Short Range Communication (DSRC) technology, which was adopted by the intelligent transportation system (ITS) has a radio range of up to 1000 meters [21].

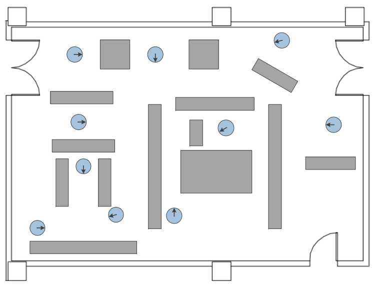

Figure 8—CarSpeak's Indoor Testbed. The blue circles denote candidate locations for robots and gray boxes denote obstacle locations.

Robot Operating System. We operate ROS on the Ubuntu 11.04 distribution (with linux kernel version 2.6.38-8-generic), that runs on the ASUS netbooks attached to the iCreate robots. Our implementation of CarSpeak's multi-resolution naming system maintains the Octree datastructure with $L = 8$ and three levels of region sub-trees. We also implement CarSpeak's garbage collection as a ROS timer thread with a threshold of 10 seconds for the freshness of sensor information.

Our implementation of CarSpeak's multi-resolution naming system subscribes to multiple topics containing sensor information in ROS's PointCloud format. It publishes the /car_speak topic, in ROS's PointCloud format, based on UDP packets received from the MAC layer. In this sense, CarSpeak behaves as any other sensor module in ROS. We implement CarSpeak's Octree-based compression framework to sub-sample the Octree and generate UDP packets to be forwarded to the MAC module.

CarSpeak's content centric MAC implementation has two key requirements: 1) The ability to modify channel access parameters such as the contention window size and, 2) Accurate timing to ensure packets are transmitted by the driver with minimum queuing delay. We chose the open-source ath9k driver+firmware for Atheros 802.11n based chipsets because it met our requirements. In our implementation, whenever the driver receives a packet (over-the-air or from userspace), it searches for a CarSpeak header within the payload of the packet to identify it as a CarSpeak packet. If the packet is from userspace, the driver places it in a queue corresponding to the region for which the packet contains information. The driver does not directly transmit the packet because the next packet to transmit (based on region sampling probabilities) may not correspond to the region for which the packet contains information. For actual transmission, we create a separate high priority thread within the driver to schedule packets based on the region sampling probabilities discussed in Section §3. Once a region is chosen for transmission, the thread dequeues the packet from the region's queue, sets the CWMin for the hardware's queue, and writes the packet into the hardware's queue. To minimize waisted airtime, we schedule this thread as fast as possible with the help of *High-Resolution Timers* available in the 2.6.x version of the Linux kernel. HR Timers are very accurate, with scheduling errors as low as 10us.

10. EVALUATION ENVIRONMENT

Below we describe the testing environment and the evaluated schemes.

Testbeds: We evaluate CarSpeak in both indoor and outdoor settings. Our indoor testbed uses a Vicon motion capture system for robot localization, and contains 10 iRobot Create robots equipped with Xbox 360 Kinect sensors. Asus EEPC 1015PX netbooks equipped with Atheros AR9285 wireless network adapters are mounted on each robot. Our testbed includes several large and small

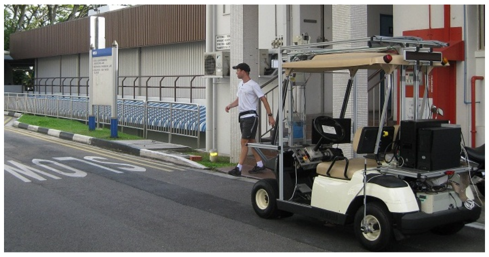

Figure 9—CarSpeak's Outdoor Setup. Image of the actual golf car route demonstrating the lobby area that poses a hazardous blind spot for the golf car and makes visual confirmation of a pedestrian difficult before he enters the road.

obstacles as shown in Fig. 8. The testbed is divided to 40 high resolution regions. Low resolution regions are specified per experiment.

Our outdoor testbed contains an autonomous Yamaha G22E golf car mounted with various sensors, such as cameras, SICK and Hokuyo range finders. The autonomous car, navigating in a campus-like environment, needs to detect pedestrians and other vehicles. We implement CarSpeak on the golf car and several iRobot Create robots equipped with Kinect sensors situated in multiple locations. The setup was deployed over an area of 20 $\times$ 40 m. The robots assist the golf car's navigation system by providing sensor information useful in detecting pedestrians in the environment. Figure 9 shows the actual pedestrian crosswalk and depicts that the lobby adjacent to the crosswalk is a blind spot for the vehicle.

Compared Schemes: We evaluate three schemes including CarSpeak and two baseline implementations:

- **802.11**: An 802.11 based inter-vehicle communication system, which allows vehicles to make requests for regions, Responses are in the form of UDP/broadcast packets and are provided by all wireless nodes which possess any information about the given set of regions. The system uses the standard 802.11 MAC protocol to transmit information. The protocol keeps track of requests and causes requests older than one minute to expire. It also discards sensor data older than 10 seconds. The system however does not implement Octree-based naming or compression and instead transmits raw 3D-point cloud information. It also does not implement the functionalities of the content centric MAC.
- **802.11+Naming**: This baseline includes CarSpeak's Octree based naming and compression modules. It tracks requests and transmits packets from each region proportionally to the number of requests it received for that region, i.e., REQ_r. It also times out requests after one minute and discards sensor data older than 10 seconds. However it does not implement region-based contention or other CarSpeak MAC functions.
- **CarSpeak**: CarSpeak with all of its components including the content-centric MAC.

Metric: We compare CarSpeak against these baseline implementations based on a utility function, computing the *rate of useful sensor information*, received per second. A 3D point cloud is considered *useful*, if it contains sensor information only from the requested region(s), at the right resolution. For e.g., if a region is requested at a coarse resolution, fine grained high resolution information from that region are aggregated into the requested resolution and then their contribution to the useful information is computed. If all the fine grained information covers only 1 point in the requested coarse resolution, their contribution to the utility metric will be 1 point.

11. RESULTS

We evaluate CarSpeak in both indoor and outdoor environments. Our indoor testbed contains several obstacles that create blind-spots for the robots. Figure 8 depicts candidate robot locations in the test-bed. Experiments are repeated with robots assigned to different randomly chosen locations and moving towards different randomly chosen destinations. Our experiments allow robots to obtain sensor information from a diverse set of regions at various points in time.

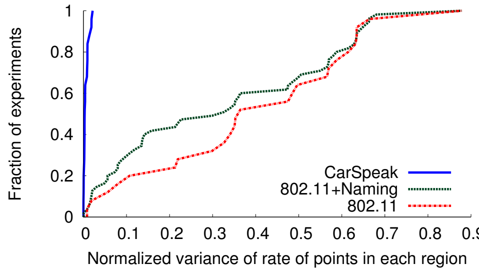

Figure 10—Region Contention. CDFs of variance in rate of points received from two regions. The variance obtained using CarSpeak is significantly smaller, across a variety of topologies and in the presence of mobile robots.

11.1 Region Contention

CarSpeak's key goal is to enable regions to share the medium efficiently, regardless of the location or number of nodes. We verify if CarSpeak delivers on that promise.

Method. We place robots in randomly chosen locations in the indoor testbed. We issue an equal number of requests for two different regions in the environment at regular intervals from two wireless nodes in the testbed. We measure the variance of the rate of 3D-points received from the two regions by both robots, by CarSpeak the standard 802.11 MAC protocol, and a hybrid approach 802.11+Naming. We repeat the experiment for 20 different topologies, with requests generated from different pairs of robots.

Results. Figure 10 shows the cumulative distribution function (CDF) of the variance (normalized by the average square) of the rate of points received from the two regions by the robots. The mean variance obtained using CarSpeak is 0.0015, while that of the standard 802.11 protocol and 802.11+Naming are 0.101 and 0.081 respectively. The higher 802.11 variance is due to the fact that 802.11 allocates bandwidth to senders not regions. Hence, the region that was observed by more robots received a greater share of the medium compared to the other region. The exact difference in the shares of the two regions varied from one experiment to another depending on the topology and mobility pattern. 802.11+Naming had a slightly lower variance. This is because the protocol enforces the desired region rates locally – i.e., if one robot has information from both regions the amount of data it transmits is balanced between the two regions – but cannot guarantee the desired medium allocation across different nodes. In contrast, CarSpeak's region based contention mechanism ensures that the medium is shared equally between the two requested regions, across a variety of topologies and mobility patterns.

11.2 Region Requests

In this experiment, we test CarSpeak's region request module and verify an increased number of requests for a given region leads to a proportional increase in the number of 3D points received from that region.

Method. We place robots in randomly chosen locations in the indoor testbed. We issue queries for two regions in the environment. We fix the query rate for the first region (5 requests/sec) and vary the query rate for the second region across experiments. We measure the ratio of the number of points received from the two regions at the requesting robots, when the experiments are carried using

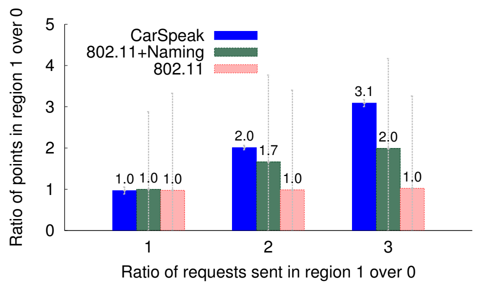

Figure 11—Region Requests. Ratio of the number of points received from two requested regions plotted against ratio of the number of requests made for the two regions. CarSpeak ensures the wireless bandwidth is allocated to region proportionally to the number of requests they receive.

CarSpeak, 802.11 and 802.11+Naming. We repeat the experiment for 20 different topologies, with requests generated from different pairs of robots.

Results. Figure 11 plots the ratio of the number of 3D points received from the two requested regions as a function of the ratio of the number of requests made for the two regions. The figure shows that, for CarSpeak, the ratio of received points is roughly equal to the ratio of requests. This holds across a variety of topologies and mobility patterns. In contrast, for 802.11, the ratio of points is totally independent of the ratio of requests. 802.11+Naming performs slightly better showing some correlation between the ratio of points from the two regions and the ratio of their requests.

11.3 Scaling

In this experiment, we demonstrate that CarSpeak scales to environments with a large number of vehicles.

Method. We conduct the experiment with two regions that have equal request rates. However, we increase the number of transmitters and explore the impact on the protocols. We measure the number of points received by the requesting receivers for CarSpeak and the two baselines. We repeat the experiment for different topologies and pairs of regions.

Results. Figure 12 plots the number of received 3D points, with CarSpeak, 802.11, and 802.11+Naming as a function of the number of contending nodes. While CarSpeak's performance scales gracefully, the performance of both the 802.11 baselines deteriorates when there are over 6 nodes. This is due to the large number of collisions that occur when multiple nodes transmit using the 802.11 broadcast mode, causing a broadcast storm. CarSpeak's content centric MAC protocol solves this problem by adapting the nodes' contention window so that it stays independent of the number of transmitters.

11.4 Compression

In this experiment, we evaluate the performance of CarSpeak's compression module. We verify if our compression scheme is robust to packet loss while providing significant compression over sending uncompressed point cloud data.

Method. Since the level of possible compression depends on the scene, we place the robots in a typical outdoor setting containing several buildings and obstacles, with Kinect sensors receiving depth information.[3] We vary the distance between the robots to achieve a wide range of loss rates. We evaluate CarSpeak's compression module against the following two compression schemes:

[3]Kinect does not work in sunny outdoor settings. Hence, we pick afternoon hours and locations a lot of shades.

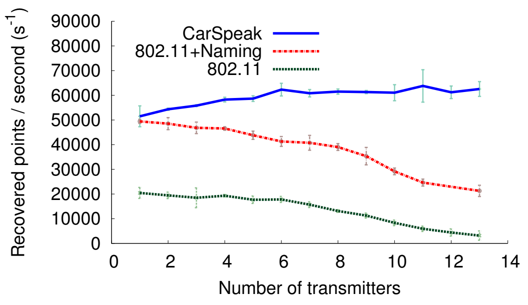

Figure 12—Scaling. Number of 3D points received at a receiver by CarSpeak, 802.11 and 802.11+Naming as a function of the number of contending nodes. We observe that while CarSpeak scales gracefully, 802.11's performance deteriorates when there are over 6 nodes, due to an excessive number of collisions and the lack of a backoff mechanism in the broadcast mode.

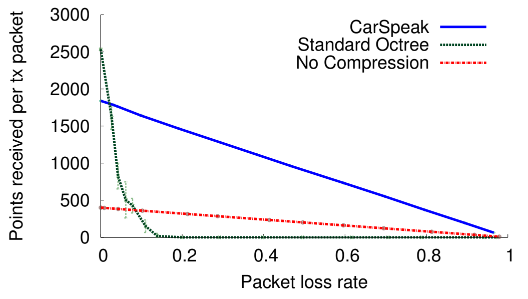

Figure 13—Compression module. Ratio of the number of points received over the number of transmitted points measured across packet loss rates when packetizing compressed point cloud data, using CarSpeak's compression module and without using compression. CarSpeak's compression module provides a consistent gain of 4.5x over sending uncompressed data, while packetizing compressed point cloud data performs poorer than CarSpeak for packet loss rates as low as 2%.

- *No Compression*: 3D-point cloud information is transmitted directly without any compression but with random sub-sampling.
- *Standard Octree Compression*: 3D-point cloud data from the environment obtained from the sensor at regular intervals is compressed using the standard Octree compression algorithm described in §6. The resultant data is packetized and broadcast on the medium.

We repeat the experiment for different locations of the robots in an outdoor setting.

Results. Figure 13 plots the number of received 3D points divided by the number of transmitted packets, as a function of the packet loss rate. CarSpeak's compression module provides a consistent gain of 4× over sending uncompressed data. While packetizing compressed point cloud data achieves a greater compression at very low loss rates, the scheme deteriorates to poorer than sending uncompressed data at a packet loss rate of 10% (which we found to be typical in our mobile outdoor scenarios). Since point cloud data is sought by several receivers whose channel to the transmitter varies with time due to mobility, a practical compression scheme must be robust to a wide range of packet loss. CarSpeak delivers on this promise.

11.5 Resolution

In this experiment, we evaluate the performance of CarSpeak when observing regions at different resolutions. We verify if Car-

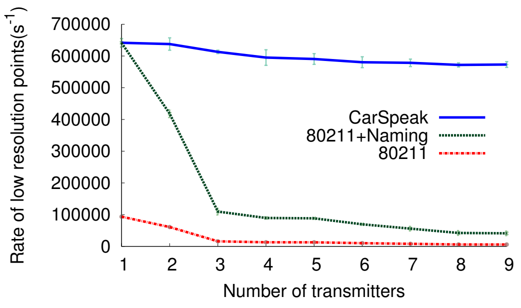

Figure 14—Resolution. The figure shows the rate at which information is received from a requested low-resolution region, for different schemes. Increasing the number of robots that do not have the requested resolution can dramatically impact the performance of 802.11 and 802.11+Naming. In contrast, CarSpeak maintains high rate of information from the desired region.

Speak responds with high quality information at the appropriate resolution, when a region is requested.

Method. We experiment with a scenario in which a robot requests a region at a low resolution. The environment has one robot who has the region at the proper resolution and many other robots that have incomplete and higher resolution information of the region. We measure the number of the 3D points received from the large region at the requester, in each of the three compared schemes. Note that fine grained high resolution information from within the requested region are aggregated into the requested resolution and then their contribution to the useful information is computed. For example, if all the fine grained information ends up covering only one point in the requested coarse resolution, their contribution to the utility metric will be one point. We repeat the experiment 20 times under different topologies.

Results. Figure 14 plots the rate at which the requester receives points from the desired resolution. The figure shows that adding robots observing smaller regions does not reduce CarSpeak's performance, as it recognizes that the robot observing the entire region has a greater quality of sensor information and deserves greater access to the medium. However, 802.11's performance is reduced as the medium is increasingly shared by wireless nodes observing only a small fragment of the requested region. Note that as the 802.11 baseline does not implement Octree-based compression, its rate of received sensor information is lower, compared to 802.11+Naming or CarSpeak. Overall, across experiments, CarSpeak delivery rate of the desired data is 4.5× higher than 802.11+Naming and over 29× higher than 802.11.

11.6 Planning Efficiency

In this experiment, we demonstrate CarSpeak's capability to provide the path planner with more efficient routes in an environment with obstacles.

Method. Consider a topology of the robots as shown in figure 15(a). Robot A seeks to navigate to location X, via the shortest possible path. However, the road ahead of X is blocked, and this information is available only with Robot B. Robot A does not have a line-of-sight view of the road block. The environment also has several other robots positioned at various other locations with sensor information of lower importance, also contending for the medium. Robot A makes several requests for regions close to X, for which its own sensors have no information. In the presence of timely sensor information from Robot B, Robot A can make a detour at the intersection to reach its destination via a marginally longer route. However, without this information, Robot A reaches the road-block and must U-turn to take the detour. We repeat the experiment with a different number of vehicles contending for the medium in the environment.

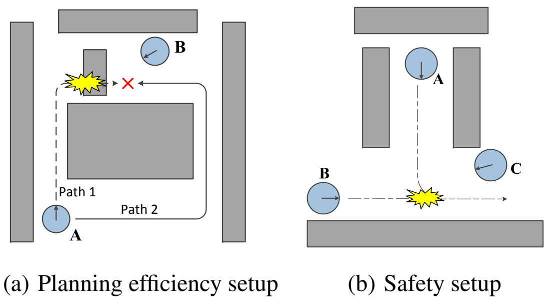

Figure 15—Experiment setup for (a), The shortest path 1 from A to destination is blocked, efficient communication from sensor B should enable A to take path 2; for (b), There is a potential collision of A with B when A tries to merge into the traffic in another road at the T intersection. The collision can be avoided if A can hear from C about the other side of the road.

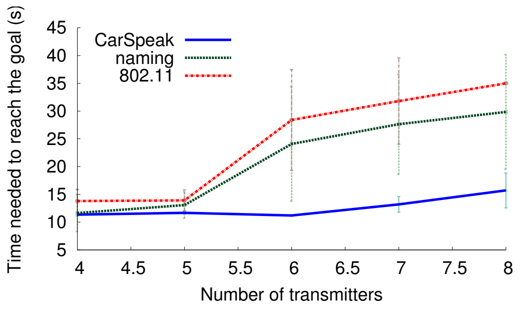

Figure 16—Planning Efficiency. Time taken by the robot to navigate an obstacle ridden environment across number of contending wireless nodes. CarSpeak performs 2.4× better, on average, than the baseline 802.11 implementation in a network of over 6 robots.

Results. Figure 16 plots the time taken by robot A to navigate to location X vs. the number of contending nodes when running CarSpeak as well as the two 802.11 benchmarks. We observe that in a network of over 6 contending wireless transmitters, CarSpeak performs, on average, 2.4× better than the 802.11 baseline and 2.1× better than 802.11's MAC with CarSpeak's multi-resolution naming system. In this network, 802.11 predominantly picks the incorrect path to the destination, while CarSpeak correctly picks the detour at the intersection, with high probability. While, 802.11+Naming performs marginally better than 802.11 due to a more effective compression scheme, its performance remains poor as much of the available wireless bandwidth is used by other nodes, with sensor information of much lower importance.

11.7 Safety

In this experiment, we evaluate CarSpeak's effectiveness in improving the safety of autonomous driving by detecting obstacles outside the field of view of the vehicle.

Method. Consider a topology of the robots as shown in figure 15(b) emulating the common scenario of vehicles at an intersection. Robot A is navigating towards a T-intersection and seeks to merge with other traffic on the main roadway. Ideally, Robot A must yield to Robot B (emulating a human-driven car without sensors), which is currently traveling on the main road. However, Robot A's sensors have a limited field of view and cannot detect Robot B. Negotiating such intersections is one of the most challenging problems in designing autonomous vehicles, often requiring human intervention or additional information regarding obstacles on the road [7]. In this topology, Robot C has access to sensor information capturing

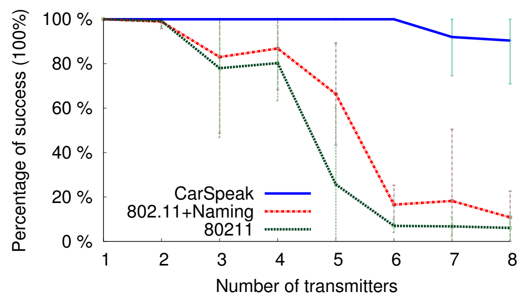

Figure 17—Safety. Percentage of successful detection of Robot B across number of wireless nodes contending for the medium. For a network of over 6 contending transmitters, CarSpeak's probability of successfully detecting Robot B is 14× that of 802.11 and 6.5× that of 802.11+Naming.

Robot B. The network has several other robots contending for the wireless medium, placed in randomly chosen locations. We evaluate the ability of Robot A to detect Robot B, while implementing CarSpeak against the 802.11 baseline implementations. We repeat the experiment with different numbers of vehicles contending for the medium in the environment.

Results. Figure 17 plots the percentage of successful detection of Robot B vs. the number of wireless nodes contending for the medium. While the performance of 802.11 and 802.11+Naming deteriorate to as low as 6.1% and 11.9% as the number of robots increases, CarSpeak successfully detects Robot B with 91% probability. In a network of over 6 transmitters, CarSpeak's probability of detecting Robot B is 14× that of 802.11 and 6.5× that of 802.11+Naming.

11.8 Outdoor Experiments on an Autonomous Vehicle

CarSpeak was implemented in an outdoor setting at a pedestrian crosswalk in a campus-like environment. This pedestrian crosswalk presents a hazardous setting where the two buildings on either side of the crosswalk completely block the view such that vehicles on the road are not aware of pedestrians before they emerge onto the street. See Figure 9. We present empirical results demonstrating CarSpeak's capability of improving the stopping time of an autonomous Yamaha G22E golf car over 802.11 when point cloud sensor data for pedestrians in the vehicle's blind spot is transmitted to the vehicle. In particular, our results show that CarSpeak enables the vehicle to make a stop decision before the crosswalk even at full speeds, if a pedestrian appears when the vehicle is one to two meters away from from the crosswalk.

Method. Our setup consists of a total of six Kinect sensors placed adjacent to the pedestrian crosswalk, i.e., the vehicle blind spot. The experiments were conducted in the presence of multiple collision domains, and hidden terminals.[4] Five out of six of these Kinects are monitoring a different section of the environment and thus are inconsequential for detecting pedestrians entering the crosswalk. Only one of the Kinect sensors is strategically placed to monitor the pedestrian crosswalk blind spot and thus obtains information relevant for the vehicle. Each Kinect broadcasts its point cloud sensor information using the Asus netbook described in §10. A receiver node on the autonomous golf car, a Vaio VPCF23BFX laptop with an Intel Core 17-2670QM processor, processes the sensor data that it receives from the Kinects to infer the presence of a pedestrian in the critical region (i.e., vehicle blind spot) viewed

[4]Pairwise pings show that only a subset of the pairs can directly hear each other, and in some pairs, the two nodes do not receive each other's pings though a third node can receive pings from both nodes, which indicates a hidden terminal scenario.

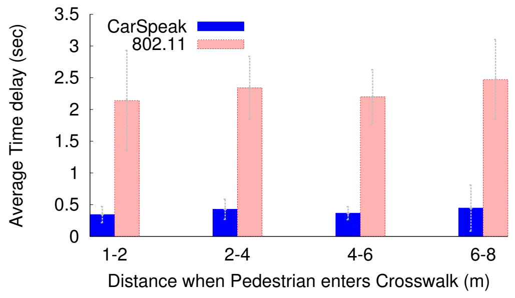

Figure 18—Time Delay Averages for CarSpeakand 802.11. Distances on the x-axis are grouped into bins of two meters representing distances of the golf car from the crosswalk at the time the pedestrian enters the crosswalk.

by the high priority Kinect. The pedestrian detection module issues a positive reading if the number of point cloud data points within the critical region is above a threshold of 1000 points. Upon detecting the presence of the pedestrian, the receiver node immediately publishes a stop command to the golf car through a ROS publish/subscribe interface. We compare CarSpeak against the benchmark of the traditional 802.11 protocol for data transmission.

For the purposes of obtaining our performance metric, we make the node attached to the Kinect log the sensor data to detect the exact time when the pedestrian appears in the lobby in front of the transmitting Kinect. This time is then compared against the timestamp of when the receiver issues a stop command to the vehicle. Using the vehicle's on-board localization paired with the two timestamps recorded, we also compare the distance of the vehicle from the crosswalk when the pedestrian enters the crosswalk and when the golf car is issued a stop command by the receiver. We note however, that processing is not necessary at the transmitter and is only done for computing our performance metrics.

The golf car drives from 15 meters away towards the crosswalk. We perform the experiment by allowing the pedestrian to enter the crosswalk's blind spot when the golf car is traveling at a full speed of two meters per second at distances of roughly ten meters, eight meters, six meters, four meters, and two meters from the crosswalk. For all of our results we assume the pedestrian takes an additional 0.5 seconds to enter the crosswalk from the time he is detected at the Kinect in the lobby and this is the time value we use on the x-axis of our plots. The results of these experiments are averaged over five runs for each of these distances using both CarSpeak and the traditional 802.11 protocols and are compared in the next section.

Results. Our results, in Figure 18, show a clear improvement in the vehicle's ability to safely stop before the crosswalk using CarSpeak as compared to 802.11. In particular CarSpeak allows for the receiver to issue a stop command with a minimum average delay of as little as 0.3 seconds from when the pedestrian appears in the field of view of the Kinect and a maximum average delay of 0.45 seconds. The maximum average delay of a positive pedestrian detection using CarSpeak is 4.75 times smaller than the *minimum* delay of 2.14s using 802.11.

These relatively small delays using CarSpeak allow the vehicle to safely stop before the crosswalk even when it is one to two meters away and traveling at a speed of two meters per second when the pedestrian appears. Use of the traditional 802.11 protocol, however, fails to stop the car before the crosswalk if a pedestrian appears when the vehicle is closer than four meters from the crosswalk, on average. See Figure 19. Using CarSpeak allows for a larger portion of critical information requested by the golf car from the priority Kinect sensor to reach the receiver, whereas an 802.11 protocol floods the receiver with proportionally more data from the irrelevant

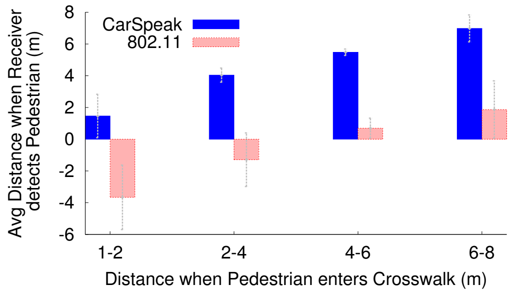

Figure 19—Comparison of golf car distance from the crosswalk when the pedestrian enters the crosswalk to the distance from the crosswalk at which the receiver issues a stop command to the car. Distances on the x-axis are grouped into bins of 2 m.

Kinect sensors, inhibiting the receiver's ability to process a positive pedestrian detection. In particular, using CarSpeak the receiver obtains 7.5× as many pedestrian critical 3D points as 802.11, averaged over twenty runs. Thus, using CarSpeak allows the receiver to gain several folds more information about regions of the environment that it considers important, even with several contending non-relevant transmitters, allowing more timely usage of important data to make critical decisions on actual autonomous vehicles.

12. CONCLUSION

This paper introduces CarSpeak, a content-centric communication system for autonomous driving, enabling cars to query and access sensory information captured by other cars in a manner similar to how they access information from their local sensors. Field tests using a combination of iRobot robots and a Yamaha instrumented car show that, in comparison with a baseline that directly uses 802.11, CarSpeak improves safety, increases information throughput, and provides several folds reduction in the time to navigate an obstacle-ridden environment.

Acknowledgments: We thank Brice Rebsamen and Tirthankar Bandyopadhyay for assisting us with the golf car in Singapore. We thank Sam Powers, Edward Mugica, Sachithra Hemachandra for helping us with our indoor testbed. We also thank Haitham Al-Hassanieh, Fadel Adib, and the NETMIT group for their support. This work is funded by NSF and SMART-FM. We thank the members of the MIT Center for Wireless Networks and Mobile Computing, including Amazon.com, Cisco, Intel, Mediatek, Microsoft, and ST Microelectronics, for their interest and support.

13. REFERENCES

[1] G. Bianchi. Performance analysis of the ieee 802.11 distributed coordination function. *Selected Areas in Communications, IEEE Journal on*, pages 535 –547, 2000.
[2] L. Bononi and M. Di Felice. A cross layered mac and clustering scheme for efficient broadcast in vanets. In *MASS*, pages 1–8, 2007.
[3] M. Campbell, M. Egerstedt, J. P. How, and R. M. Murray. Autonomous driving in urban environments: approaches, lessons and challenges. *Philosophical Transactions of the Royal Society Series A*, 368:4649–4672, 2010.
[4] J. Chesterfield and P. Rodriguez. Deltacast: efficient file reconciliation in wireless broadcast systems. MobiSys '05, 2005.
[5] Z. Chong, B. Qin, T. Bandyopadhyay, T. Wongpiromsarn, E. Rankin, M. Ang, E. Frazzoli, D. Rus, D. Hsu, and K. Low. Autonomous personal vehicle for the first- and last-mile transportation services. In *CIS 2011*, pages 253 –260, 2011.
[6] C. Cuyu, X. Yong, S. Meilin, and L. Liang. Performance observations on mac protocols of vanets in intelligent transportation system. In *CMC '09.*, pages 373 –379, 2009.
[7] M. Darms, P. Rybski, and C. Urmson. Classification and tracking of dynamic objects with multiple sensors for autonomous driving in urban environments. In *Proceedings of the 2008 IEEE Intelligent Vehicles Symposium*, pages 1197–1202, 2008.
[8] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *Int. J. Rob. Res.*, pages 485–501, 2010.
[9] S. Gil, M. Schwager, B. Julian, and D. Rus. Optimizing communication in air-ground robot networks using decentralized control. In *ICRA 2010*, pages 1964 –1971, 2010.
[10] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. Visual odometry and mapping for autonomous flight using an rgb-d camera. In *ISRR*, 2011.
[11] Y. Huang, J. Peng, C.-C. Kuo, and M. Gopi. A generic scheme for progressive point cloud coding. *Visualization and Computer Graphics, IEEE Transactions on*, 2008.
[12] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *CoNEXT '09*, pages 1–12, 2009.
[13] S. Jiang and H.-L. Chao. Linear cooperative detection for alarm messages in cluster-based vehicular ad hoc networks. In *WCNC, 2010 IEEE*, pages 1 –6, 2010.
[14] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *CoRR*, 2011.
[15] A. Kolling and S. Carpin. Multi-robot pursuit-evasion without maps. In *ICRA 2010*, pages 3045 –3051, 2010.
[16] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. In *SIGCOMM '07*, pages 181–192, 2007.
[17] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. Antone, R. Galejs, S. Krishnamurthy, and J. Williams. A perception-driven autonomous urban vehicle. *J. Field Robot.*, pages 727–774, 2008.
[18] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun. Towards fully autonomous driving: Systems and algorithms. In *IEEE IV 2011*, pages 163 –168, 2011.
[19] F. Li and Y. Wang. Routing in vehicular ad hoc networks: A survey. *Vehicular Technology Magazine, IEEE*, pages 12 –22, 2007.
[20] P. Luo, H. Huang, W. Shu, M. Li, and M.-Y. Wu. Performance evaluation of vehicular dtn routing under realistic mobility models. In *WCNC 2008. IEEE*, pages 2206 –2211, 2008.
[21] B. Maitipe and M. I. Hayee. Development and field demonstration of dsrc-based v2i traffic information system for the work zone. Technical report, 2010.
[22] F. Maurelli, D. Droeschel, T. Wisspeintner, S. May, and H. Surmann. A 3d laser scanner system for autonomous vehicle navigation. In *ICAR 2009.*, pages 1 –6, june 2009.
[23] Mendez-Polanco and Munoz-Melendez. Collaborative robots for indoor environment exploration. In *ICARCV 2008*, 2008.
[24] J. L. Ny, A.Ribiero, and G. Pappas. Robot deployment with end-to-end communication constraints. In *Proceedings of the Conference on Decision and Control*, 2011.
[25] S. Oh, D. Lau, and M. Gerla. Content centric networking in tactical and emergency manets. In *Wireless Days IFIP*, pages 1 –5, 2010.
[26] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.
[27] D. Reichardt, M. Miglietta, L. Moretti, P. Morsink, and W. Schulz. Cartalk 2000: safe and comfortable driving based upon inter-vehicle-communication. In *Intelligent Vehicle Symposium, 2002*, pages 545 – 550 vol.2, 2002.
[28] R. Schnabel and R. Klein. Octree-based point-cloud compression. In *Symposium on Point-Based Graphics*, 2006.
[29] N. Thompson, R. Crepaldi, and R. H. Kravets. Locus: A location-based data overlay for disruption-tolerant networks. In *Fifth Workshop on Challenged Networks (CHANTS 2010)*, 2010.
[30] O. Tonguz, N. Wisitpongphan, F. Bai, P. Mudalige, and V. Sadekar. Broadcasting in vanet. In *2007 Mobile Networking for Vehicular Environments*, pages 7 –12, 2007.
[31] C. Urmson, C. Baker , J. Dolan, P. Rybski, B. Salesky, W. Whittaker, D. Ferguson, and M. Darms. Autonomous driving in traffic: Boss and the urban challenge. *AI Magazine*, pages 17–29, 2009.