Inside the Virtual Robotics Challenge: Simulating Real-Time Robotic Disaster Response

Carlos E. Agüero, Nate Koenig, Ian Chen, Hugo Boyer, Steven Peters, John Hsu, Brian Gerkey, Steffi Paepcke, Jose L. Rivero, Justin Manzo, Eric Krotkov, and Gill Pratt

***Abstract*—This paper presents the software framework established to facilitate cloud-hosted robot simulation. The framework addresses the challenges associated with conducting a task-oriented and real-time robot competition, the Defense Advanced Research Projects Agency (DARPA) Virtual Robotics Challenge (VRC), designed to mimic reality. The core of the framework is the Gazebo simulator, a platform to simulate robots, objects, and environments, as well as the enhancements made for the VRC to maintain a high fidelity simulation using a high degree of freedom and multisensor robot. The other major component used is the CloudSim tool, designed to enhance the automation of robotics simulation using existing cloud technologies. The results from the VRC and a discussion are also detailed in this work.**

***Note to Practitioners*—Advances in robot simulation, cloud hosted infrastructure, and web technology have made it possible to accurately and efficiently simulate complex robots and environments on remote servers while providing realistic data streams for human-in-the-loop robot control. This paper presents the software and hardware frameworks established to facilitate cloud-hosted robot simulation, and addresses the challenges associated with conducting a task-oriented robot competition designed to mimic reality. The competition that spurred this innovation was the VRC, a precursor to the DARPA Robotics Challenge, in which teams from around the world utilized custom human-robot interfaces and control code to solve disaster response-related tasks in simulation. Winners of the VRC received both funding and access to Atlas, a humanoid robot developed by Boston Dynamics. The Gazebo simulator, an open source and high fidelity robot simulator, was improved upon to met the needs of the VRC competition. Additionally, CloudSim was created to act as an interface between users and the cloud-hosted simulations.**

As a result of this work, we have achieved automated deployment of cloud resources for robotic simulations, near real-time simulation performance, and simulation accuracy that closely mimics real hardware. These tools have been released under open source licenses and are freely available, and can be used to help reduce robot and algorithm design and development time, and increase robot software robustness.

***Index Terms*—Cloud robotics, real-time robot simulation, robotic disaster response.**

Manuscript received March 25, 2014; revised August 13, 2014; accepted August 30, 2014. Date of publication January 19, 2015; date of current version April 03, 2015. This paper was recommended for publication by Associate Editor A. Aydemir and Editor S. Sanjay upon evaluation of the reviewers' comments. This work was supported in part by DARPA TTO under Project DARPA-BAA-12-39. The views, opinions, and/or findings contained in this paper are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense. Approved for Public Release, Distribution Unlimited.

C. E. Agüero, N. Koenig, I. Chen, H. Boyer, S. Peters, J. Hsu, B. Gerkey, S. Paepcke, and J. L. Rivero are with the Open Source Robotics Foundation, Mountain View, CA 94041 USA (e-mail: caguero@osrfoundation.org; nate@osrfoundation.org; ichen@osrfoundation.org; hugo@osrfoundation.org; scpeters@osrfoundation.org; hsu@osrfoundation.org; gerkey@osrfoundation.org; steffi@osrfoundation.org; jrivero @osrfoundation.org).

J. Manzo is with Booz Allen Hamilton, McLean, VA 22102 USA (e-mail: manzo_justin@bah.com).

E. Krotkov is with Griffin Technologies, Wynnewood, PA 19096-3307 USA (e-mail: ekrotkov@griffin-technologies.com).

G. Pratt is with the Defense Advanced Research Projects Agency, Tactical Technology Office, Arlington, VA 22203-2114 USA (e-mail: gill.pratt@darpa.mil).

Digital Object Identifier 10.1109/TASE.2014.2368997

I. Introduction

THE Defense Advanced Research Projects Agency (DARPA) Robotics Challenge (DRC) [1] of 2012–2015 is a prize competition that aims to advance the state of the art of disaster response. Using semi-autonomous robots, teams must solve a series of complex and heterogeneous tasks, such as driving a car and navigating on foot through a rubble pile. The goal of the contest is to advance the development of robots that can assist humans in mitigating and recovering from natural and man-made disasters, such as the catastrophic failure at the Fukushima I Nuclear Power Plant on March 11, 2011, resulting in a meltdown of several of its nuclear reactors.

Leading up to the final DRC event, a virtual competition was conducted to determine which teams would receive funding and access to an Atlas humanoid robot [2] developed by Boston Dynamics Incorporated (BDI). This virtual competition, the Virtual Robotics Challenge (VRC) [1], was the first of its kind where teams from around the world would use cloud-hosted resources to complete simulated tasks that mimic disaster response scenarios. During December 2013, teams competed in the DRC trials. This event tested the mobility, manipulation and dexterity of the robots in a real environment. The DRC Finals will be held during 2015 and the winning team will get a $2 million final prize.

In order to successfully host the VRC, significant advances in robot simulation and its integration with cloud resources were required. Improvements to simulated physics accuracy and performance made it possible to run complex scenarios in near real-time with human-in-the-loop robot control. These improvements also allowed teams to develop algorithms that perform nearly identically in simulation as on real hardware, thereby increasing team productivity and code quality. Automated deployment of simulation onto cloud machines simplified the process of running simulation, and provided a consistent and uniform environment in which teams could compete. This work has been released under open source licenses and is freely available to not only teams but anyone

interested in robotics through the open-source projects Gazebo [3], a robot simulator, and CloudSim [4], a tool to deploy and control simulation in the cloud, developed by the Open Source Robotics Foundation (OSRF).

Twenty-six teams were selected to participate in the VRC, held in June of 2013. The teams competed on the same DRC tasks recreated using Gazebo and hosted on CloudSim. The VRC was a real-time competition held over three days. Teams were required to complete 15 tasks grouped into three categories: locomotion, driving and manipulation. The purpose of the VRC was to accelerate the software development using a simulated environment, as well as to reduce the risk of working with complex and expensive robots. Multiple people can work concurrently using simulators, whereas a real robot requires attention from several people and infrastructure (batteries, cables, experiment environment setup). In addition, different algorithms, parameters, or tests can be massively and concurrently executed on simulation without any risk, improving the development efficiency by orders of magnitude.

The integration between Gazebo and the cloud services used to host the computing power was of primary importance in the development of the infrastructure for the VRC. During the competition, each team had an exclusive connection to a remote cluster of machines called a *constellation*. Each constellation simulated the team's tasks one at a time, isolated the environment from the other teams, and artificially induced latency in the communications to ensure equal latency across teams, regardless of geographical location.

In this paper, we describe the design and implementation of the VRC infrastructure used in the Virtual Robotics Challenge. One of the novel contributions presented in this work is the CloudSim tool. CloudSim creates the link between the Gazebo simulator and cloud services; it facilitates the deployment and execution of simulations in the cloud. CloudSim was instrumental in the architecture of the VRC and its use can be extended to education, automated simulation tests, and future competitions. The Gazebo simulator, used to create and execute the VRC tasks, is also presented in this work. Performance and accuracy enhancements made to Gazebo were essential for near-real time performance and stable robot controllers. The combination of CloudSim and advancements to Gazebo have produced the first low-cost method for accessing robot simulation and the first robot simulator capable of accurately simulating walking and grasping behaviors in a high-degree of freedom robot.

This paper is organized as follows. First, we present a review of other related contests. Section III describes the architecture developed for the Virtual Robotics Challenge, emphasizing the various components hosted in the cloud. We then present the primary contributions made by the Gazebo simulator, followed by discussion of the CloudSim tool and its main contributions to the robotics community. Next, the results obtained during the Virtual Robotics Challenge will be presented with analysis. Finally, we present further discussion and future research.

II. Related Work

DARPA has historically used competitions to promote research and development in the field of robotics. During 2004, 2005, and 2007, DARPA funded the Grand Challenge [5] and Urban Challenge [6] projects to advance autonomous driving capabilities. The DRC/VRC marks the first time that a simulation-based competition was included in a DARPA challenge. The annual RoboCup Rescue competition [7] aims to improve robots assisting in disaster response situations. The fundamental difference between RoboCup Rescue and the DRC/VRC is the breadth of capabilities required of the robots. In the former, competitors focus on mapping and victim identification, while in the DRC/VRC, the robots are required to be competent in a more diverse skill set, including: walking, grasping, driving, and ladder climbing.

Similar to these hardware-based contests are competitions run entirely in simulation. The RoboCup Rescue Simulation League [8] hosts one such simulation-based competition in which participants must program a team of agents to carry out search and rescue tasks. A competition using the TORCS open source simulator [9] challenges competitors to design a race car controller capable of driving autonomously while adhering to the usual rules of car racing. Finally, Robostadium [10] is a competition using the Webots simulator to recreate a soccer game consisting of teams of Nao humanoid robots.

To support the hardware infrastructure required to run such competitions, cloud services have proven invaluable. TopCoder [11] administers computer programming challenges to its user base, and executes the solutions on machines running in the cloud. As in the VRC, TopCoder leverages cloud resources to host the work of geographically dispersed competitors. The primary difference between TopCoder and the VRC is the real-time nature of the Virtual Robotics Challenge. As submissions come in at TopCoder, they are queued, validated, and the results are later reported to the competitors. In comparison, the VRC required a real-time communication flow between the competitors and the simulation running in the cloud. The cloud services supporting the VRC met the CPU/GPU and communications constraints required to support this real-time interaction.

Similar requirements exist in the context of multiplayer games. Users from around the world compete in virtual environments sharing information across servers. One of the challenges of executing the VRC was the need for high-fidelity, accurate simulation. For computer games, these requirements can be relaxed in favor of game play or graphics quality.

The use of robots to aid people who cannot or should not enter hazardous areas has been a topic of study for many years. Industrial environments contain many hazardous areas that could benefit from robot assistance. Mining operations [12] and structure inspection [13] are two such environments. Semi-autonomous behavior is a highlight of the DRC, and has been successfully used in urban search and rescue [14], industrial robots [15], and mining [16]. The work presented in this paper makes it possible to simulate dangerous scenarios, test control paradigms, and train operators to control robots. We believe advancing the tools available to robot designers and operators will help make robots more readily available to assist people when needed most.

A new topic of research has arisen in recent years under the term *Cloud Robotics* [17]. This new approach to robotics takes advantage of the unlimited resources that the cloud provides for computation and storage. The RoboEarth project [18] aims to create a shared worldwide database to be used by robots as

a knowledge exchange resource. C^2TAM [19] is an example of a cloud framework used to address the SLAM problem. Keohe *et al.* [20] presented a system architecture for a cloud-based robot grasping system. Unlike some of the cited works, the infrastructure developed for the VRC did not solve a specific robotic problem. However, it used the cloud resources to provide robotic simulations in a novel way: using the cloud.

III. Virtual Robotics Challenge (VRC) Architecture

The DRC/VRC tasks challenge participants by reproducing some of the situations that arise in a disaster. In many disasters, a robot must interact with equipment designed for humans, use limited resources, and operate in dangerous conditions. Multiple heterogeneous tasks must be performed, such as driving a vehicle and operating valves, which would be difficult to solve autonomously.

A realistic solution involves a semi-autonomous robot that operates with the assistance of one or more human operators. The VRC/DRC uses this concept of human-in-the-loop control to achieve greater robot capabilities than would be possible with a pure autonomous system. This type of robot control relies heavily on network communication between robot and operator. During a disaster, communication is likely subject to low bandwidth and high latency, which may alter control strategies used by the teleoperators.

This section details the set of specific problems that we identified when designing the VRC architecture. Following the problem statement, we describe the different components of our solution.

A. Problem Statement

- Create a simulation of Boston Dynamic's Atlas robot, a hydraulic humanoid robot with 28 degrees of freedom. The head-mounted sensor suite, MultiSense-SL [21] by Carnegie Robotics, has a stereo camera pair that can reach up to 2 megapixels of resolution. Additionally, the MultiSense-SL contains a Hokuyo laser range finder with a range of 30 meters. The Hokuyo laser is mounted on a rotary structure and includes an integrated accelerometer and gyroscope. Atlas's wrists accept third party hands. The simulation should support the dexterous hands [22] designed by Sandia National Laboratories and the Open Source Robotics Foundation, comprising four fingers, twelve degrees of freedom, tactile sensors, 3D accelerometers, 3D magnetometers, and a stereo camera.
- Simulate environments that recreate navigation, driving, and manipulation tasks. It is required to simulate the Polaris Ranger EV utility vehicle, roads, mud, debris, and objects such as a power drill for the manipulation tasks. Complex objects such as the Polaris vehicle must operate using standard human interfaces including the steering wheel, hand brake, forward-reverse switch, and the acceleration and brake pedals.
- A high fidelity simulation and real-time performance must be achieved to support control code developed by teams.
- The team resources must remain isolated from other teams. The architecture must provide mechanisms for each team to interact with its own simulation while preventing any interference or access to other teams' resources.
- The simulator must run on an independent machine that is not directly accessible by the teams. The interaction with the simulation must be performed by an API that enables sensor acquisition and robot actuators modification.
- Each team must have two dedicated machines for executing its own code and interacting with the simulator. These machines, called field computers, should run a GNU/Linux operating system and must be connected to the simulator machine by a high-speed link and not through the Internet. One of these machines will be GPU-equipped, and can be used at the discretion of each team.
- Each team should be able to remotely access its field computers or the simulation using a computer called *Operator Control Unit* (OCU). The operators of each team may teleoperate the robot from their OCUs and receive data generated by the simulator or their own programs on the field computers.
- There must be unlimited communication between the field computers and the simulation machine. However, each VRC task will impose a specific constraint on the bandwidth originated or targeted from/to the OCU. It will be required to establish a mechanism for accounting and logging the network usage between each team's OCU and its cloud computers. Furthermore, any communication that exceeds the limits set by each task will be discarded.
- The VRC architecture proposed should ensure that all teams have the same communication latency between the OCU and field computers or simulator machine.

B. Proposed Solution

The proposed solution consists of a set of machines and software infrastructure that are connected using the Robot Operating System (ROS) [23] and services offered by a cloud computing provider. Fig. 1 shows all the components of the infrastructure developed for the VRC. In the context of this work, a VRC constellation comprises four machines exclusively assigned to each team and connected on the same local network using a Gigabit Ethernet link: *DRC Simulator*, *Field Computer #1*, *Field Computer #2*, and *Router*.

The *DRC Simulator* is a software package that contains models required to simulate tasks in the DRC and plugins to facilitate communication between Gazebo and team control code. Gazebo uses the *DRC Simulator* package and together they run on a computer in the cloud (see *DRC Simulator* in Fig. 1). During the VRC, a separate instance of the DRC simulator ran on a separate, dedicated machine for each team. An additional firewall prevents the simulator from being directly accessed by its team. The teams can only interact with the simulation by using the message-based protocol exposed by ROS. The Open Source Robotics Foundation created the DRCSim [24] ROS package that contains the Atlas robot description and all the functionality to manipulate its actuators and read its sensors. The computer executing the *DRC Simulator* is equipped with two Intel Xeon E5-2690 eight-core chips, 16 GB RAM, 1 TB

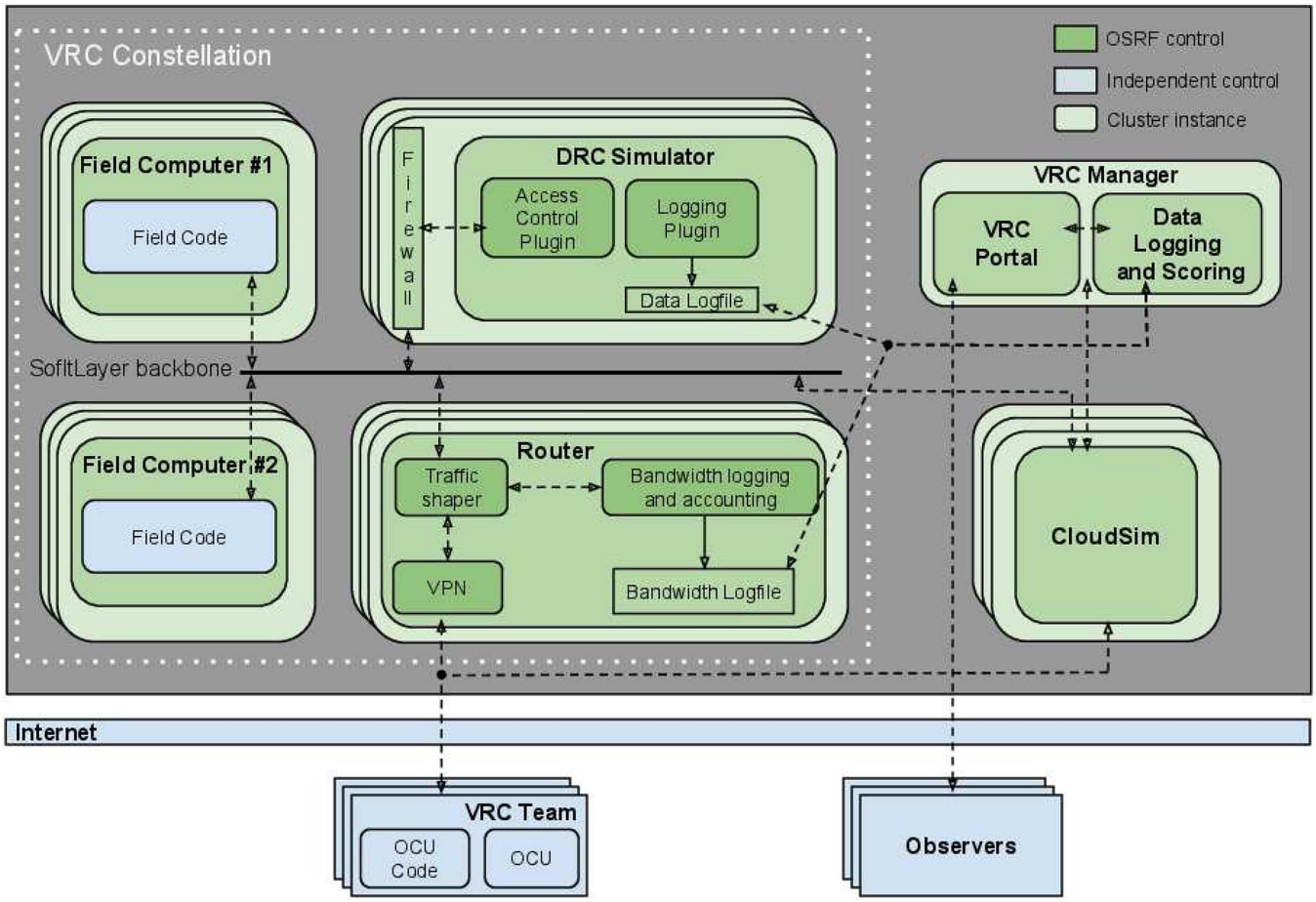

Fig. 1. VRC architecture.

HD, and a Nvidia K10 GPU, running a 64-bit Ubuntu Linux distribution.

Field Code runs on the two field computers, *Field Computer #1* and *Field Computer #2*. For the VRC, these are two cloud computers that serve as surrogates for the computer(s) that will be tethered to the Atlas robot in the DRC. Teams run all the processes requiring high-bandwidth communications with the robot on the field computers, such as the perception, planning and control algorithms. *Field Computer #1* has the same hardware specs as the computer running the *DRC Simulator*. *Field Computer #2* is equipped with 32 GB of memory, and 128 GB SSD, but does not have a GPU.

The *Router* is a lightweight cloud machine that runs a Virtual Private Network (VPN) server and scripts to measure and control latency, and record bandwidth usage. The target latency for the VRC was set to 250 ms for any incoming messages from the OCU to the field computers or simulation machine. Similarly, the target latency for any outgoing message originated by the simulator or field computers towards the OCU was set to 250 ms. We created the *Traffic shaper* subcomponent to guarantee the latency requirements by configuring Traffic Control in the Linux kernel at a frequency of 1 Hz. Bandwidth accounting between OCU and field computers was achieved by a combination of *iptables* rules and custom code to only count TCP/UDP payload. For every task, all the bandwidth statistics were stored on disk for future analysis using the *Bandwidth logging* subcomponent.

OCU code runs on a system of one or more computers at the team's facility (see *OCU Code* within the *VRC Team* in Fig. 1). This computer system serves as the operator control station that allows command and control of the robot. The operator controls (keyboard, joystick, gaming console, or equivalents) run here. Teams may have any number of machines at their end, including local or cloud-hosted machines, but all machines containing *OCU Code* are on the other side of the data-metered link from the *Field Code*.

The *VRC Manager* comprises the *VRC Portal* and the *Data Logging and Scoring* subcomponent running on a single machine in the cloud. *VRC Portal* is a web application that allows teams to access their scores, log files, as well as the URL of their CloudSim. *Data Logging and Scoring subcomponent* is synchronized with every constellation and retrieves task scores and simulation log files periodically. There are three different types of logs collected. First, the simulator log contains the raw log of a simulation episode. This log contains simulation state information suitable for in-simulation replay. The score log contains detailed information of the points scored by the team in every task, with timestamps attached to every extra point obtained. Finally, the bandwidth usage log saves all the inbound/outbound bandwidth usage during a task execution. The total size of log files for a single task might be up to 6–8 gigabytes. In order to speed up the log file transfer to the *VRC Manager*, we used HPN-SSH [25], a patched OpenSSH version allowing internal flow control buffers to be defined at runtime.

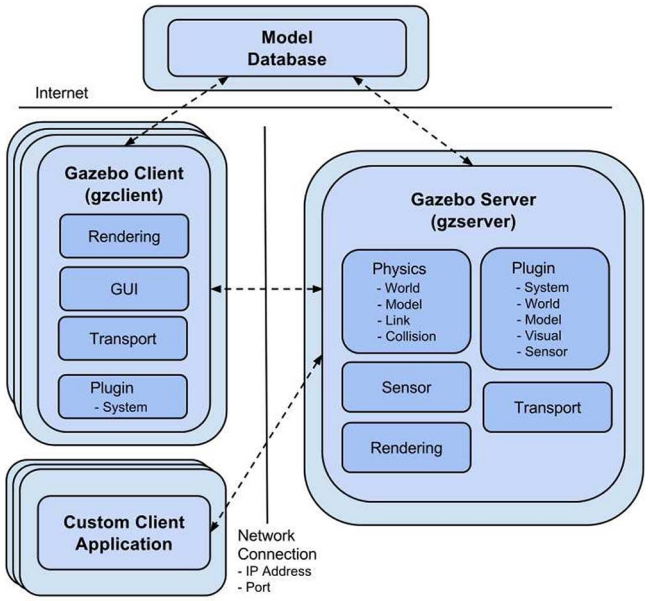

Fig. 2. Gazebo architecture.

CloudSim runs on a dedicated machine in the cloud for each team. Every team's CloudSim is accessible via a web browser, and is used to start and stop the simulations containing the task to be tackled, and provide access to some of the credentials required to log into the field computers. Essentially, CloudSim was the proxy between each team and its constellation.

IV. The Gazebo Simulator

Gazebo is the platform used to simulate robots, objects, and environments for the VRC.

A. Architecture

A key requirement of a robot simulator for the VRC is the capability to run on cloud hosted computers. Gazebo's separation of its simulation server from its graphical front-end and its use of network-based communication makes Gazebo well suited for the VRC. The simulation server is responsible for rigid body dynamics simulation and sensor data generation, while the graphical client provides visualization and user control. Communication between the two is achieved over sockets that enable inter-process communication over the network. Fig. 2 illustrates the server-client architecture of Gazebo.

The transport system uses a topic-based publisher-subscriber model. As an example, the Gazebo graphical client subscribes to topics that transmit simulation state information. This information is used to construct a 3D rendering of the simulation environment. Similarly, the server exposes a number of topics on which clients can publish in order to modify properties of simulation. In the context of the VRC, these topics are hidden from the teams and abstracted by an API specific for controlling the robot. See Section IV-A for more details. Additionally, a simple security measure in the form of an IP whitelist is put in place to ensure that no external clients can connect directly to the server to interfere with simulation while the competition is running.

B. Rigid Body Dynamics

Gazebo supports multiple physics engines for simulation of rigid body dynamics, including Open Dynamics Engine (ODE) [26], Bullet [27], Dynamic Animation and Robotics Toolkit (DART) [28], and Simbody [29].

Due to time constraints, only ODE had the full set of features needed for the VRC. A comparison of the different physics engines is the subject of current work.

An important aspect to consider when formulating the equations for articulated rigid body dynamics is the internal state representation.

In maximal coordinate formulations such as ODE, each rigid body has six degrees of freedom. Articulation is encoded as equality constraints on the dynamic states. This approach yields convenient sparse matrix structures, such as a block diagonal mass matrix and a sparse constraint Jacobian. Additionally, the approach treats contact and articulation constraints in a uniform manner.

In contrast, formulations based on internal (or generalized) coordinates (often using Featherstone's algorithm [30], [31]) consider articulation implicitly by adding the system degrees of freedom with each articulation joint. This yields more accurate kinematics and a smaller mass matrix, though the mass matrix structure is no longer sparse.

There was some discussion prior to the VRC regarding the relative merit of maximal and generalized coordinates, though a robust comparison is not presented here.

This section presents the Atlas modeling decisions that were made to improve real-time performance while maintaining sufficient physical accuracy. The kinematics, rigid body inertias, 3D mesh collision shapes, and maximum joint angles, velocities, and torques for the Atlas robot were provided by BDI. In the absence of details about the hydraulic systems, each joint was modeled as a pin joint with torque control subject to position, velocity, and torque limits. The torque and speed limits were not coupled with a torque-speed curve. Joint friction was not modeled, though viscous damping was applied at the joints in a heuristic manner to improve solver stability.

Although a full set of 3D concave meshes was provided for the Atlas robot, a union of convex sphere, box, and cylinder collision primitives approximated each collision shape. It was found that the collision primitives exhibited faster collision detection and more robust contact resolution than the 3D meshes. However, the 3D meshes were used in sensor generation for cameras and laser range finders. This was done in order to produce realistic output from the simulated Multisense-SL sensor head.

Some physical interactions were approximated due to insufficient fidelity at the required level of real-time performance. Threading a fire hose, for example, involves contact between millimeter-scale features. Instead of modeling the fine contact geometry, a screw joint was dynamically created when the fire hose coupling was sufficiently aligned with the standpipe. The coupling could be rotated in one direction to connect or in the opposite direction to release the coupling.

Sitting in the seat of the Polaris vehicle was another challenge, as the seat was modeled as a rigid body. This caused

difficulty in finding a stable seating position. To remedy this, a viscous damping field was created on the surface of the seat. This partially mitigated the problem, though stable sitting in the vehicle proved a continual challenge in the VRC.

C. Sensor Data Generation

One of the main challenges in the VRC was the generation of large volumes of sensor data in real time. There is a wide range of sensors available in Gazebo, sufficient to simulate those on Atlas. The real-time requirement is partly supported by having the sensor data generation process independent from the dynamics simulation. On the implementation level, there are two separate modules running in parallel and communicating over named topics. A sensor manager in the sensor module then provides finer control over the update rates of every sensor with the use of a priority scheme to ensure each sensor hits its target frame rate.

There are two methods of sensor data generation: image-based and physics-based. The image-based method relies on the use of the GPU for generating data. All vision sensors use this method to produce image data of the environment as seen from the sensor's perspective. To overcome performance limitations of conventional ray-casting methods for simulating range sensors, Gazebo also provides image-based range sensors which use depth images for generating range values. On the other hand, the physics-based method makes use of physics data such as forces, torques, velocities, and contacts associated with entities in the simulation. For example, an inertial measurement unit (IMU) sensor subscribes to pose and velocity data of a rigid body entity in simulation over time and outputs the sensor's orientation and accelerations. In both methods, Gaussian noise can be applied to the data before they are published to the clients.

Sensor data generation can be computationally expensive, especially if a high frame rate is required. Therefore, all sensors are inactive by default and are only enabled if there is at least one data consumer. The VRC utilized a sufficiently powerful multicore cloud machine to help satisfy performance requirements in the worst case where all physics-based sensor data were being consumed. The image-based sensors however hit a bottleneck mainly due to GPU stalling when transferring image data back to memory at a high frame rate. The teams were advised to reduce the frame rate or turn on the vision and range sensors only when needed.

D. Controller Architecture

The control architecture for the Atlas robot was chosen so that it can be made to match the real robot controller architecture as closely as possible. This includes two control modes: simple PID-based position control by actuating joint efforts individually, and behavior control. We also leveraged our experience with ROS control using joint states and joint command data structures. Close collaboration with BDI yielded the decision to model control in this way.

It is possible for teams to use their own behavior libraries by accessing the individual joint position PID controller and commanding joint torques. Team IHMC [32] implemented this approach successfully during the VRC.

High speed, low latency control over TCP/IP was possible by using a controller synchronization technique demonstrated in the DRCSim controller synchronization tutorial.[1]

E. Plugin Support

Gazebo plugins provide users direct access to the internal capabilities of Gazebo. This enables users to write custom code for controlling the behavior of various components in the simulation. Plugins operate on different levels, from the system level, down to individual models, and sensors attached to a model. For example, a model plugin can be created to simulate the differential drive of wheeled robots, and a world plugin can be used to dynamically spawn obstacles in front of the moving robot. The use of plugins helps separate application-specific code from the core Gazebo components, keeping them modular and reusable.

The VRC relies heavily on the use of plugins to provide model and task-specific capabilities. There are three primary use cases. The first is to create an abstraction layer over the internal workings of Gazebo, and offer a clean API-like interface for communication with Atlas. This interface in practice is a set of ROS topics, which is the same mechanism used for communicating with the physical robot in the DRC. This helps to minimize code modifications when transitioning to testing with Atlas in the real world.

The second use case is to provide aids for supporting goal-oriented, task-based simulation. For example, the manipulation task in the VRC requires Atlas to connect a fire hose to a standpipe and turn a valve. In this task, realistic simulation of the physical interactions between fire hose coupler and the threads of the standpipe can be difficult to achieve. In the design of a simulator with real-time constraints, it is important to understand the tradeoffs between accuracy and speed. In this case, it is more efficient to use a plugin to establish an artificial connection between the coupler and standpipe (using a screw joint) once the coupler has been successfully inserted a certain distance into the standpipe.

The third use case is to provide an automated scoring functionality for every task in the VRC. The plugin offers more accurate scoring in real time by monitoring the actions of the robot and its interaction with the environment. At the same time, it provides the teams and judges instant feedback on the completion rate of the task. A task is typically broken down into sub-tasks. For example, the driving task in the VRC requires Atlas to climb into the vehicle, drive it through a number of gates, exit the vehicle, then finally walk to the destination. Successful completion of each sub-task earns the team a point, and penalties are given if the robot falls over.

V. CloudSim

CloudSim was born out of the need to run the VRC simulation competition and handle the large number of competitors simultaneously. Furthermore, the competition had to guarantee a uniform real-time simulation performance, and be secure enough to prevent cheating, given the large prize at stake. Finally, the system had to collect all the necessary data to rank

[1]http://bitbucket.org/osrf/drcsim/tutorials/2.7/controller_synchronization

teams in a meaningful way, in order to reward the teams with the best solution.

To meet these criteria, we decided early on to host the competition in the cloud, where a large group of identical servers could be leased for a short period of time. We designed and validated a solution that would satisfy the computing and networking requirements of the VRC, with an architecture that mapped to the real hardware configuration used on board the real Atlas robot during the DRC.

At the core of CloudSim is a provisioning system that can configure a simulation network from a plain set of servers. Depending on the cloud provider's capabilities, CloudSim can:

- Create a virtual network inside a data center.
- Reserve a set machines of with varying specifications, and assign private IP address for each.
- Set up firewalls and packet routing rules between the machines.
- Install packages specific to each machine.
- Verify proper installation of the graphics system, networking and simulator.

CloudSim stores the various Secure Shell (SSH) and VPN keys for the cloud machines and makes them available to the users according to their access privileges.

CloudSim's unit of deployment is called a constellation, and CloudSim can deploy different types of constellations simultaneously. For the VRC, two constellations were deployed for each team:

- A CloudSim web application constellation, with a single server, where contestants can log in and start simulation tasks.
- A VRC simulation constellation, comprising a router machine, a simulator server and two field computers.

A simpler simulation constellation, with a single server, was also made available for practice purposes.

CloudSim offers a simulation controller via its web interface. Simulation tasks can be created and queued, and made accessible for specific periods of time. Finally, CloudSim supports three user roles (admin, officer, and user) that restrict access to the constellation machines and simulator.

The following deployment was used during the VRC: from a designated workstation running CloudSim (the master CloudSim), 30 CloudSim instances were deployed (one for each competing team, plus spares). Because the chosen cloud provider's API did not support the creation of virtual networks, those machines were deployed on 30 preconfigured Virtual Local Area Networks (VLANs). While the master CloudSim could have controlled every simulation constellation, having a CloudSim instance for each team was a precautionary measure to avoid any single point of failure during the competition.

From the master CloudSim instance, we deployed a VRC simulation constellation on each team's CloudSim. Team member credentials, the simulation tasks and the simulated worlds were then propagated to each team's CloudSim. Each CloudSim collected log files for each simulation task, and sent a copy to the VRC Portal server after each run.

Setting up and running Gazebo on a cloud server is no different from setting it up and running it on a personal UNIX workstation. Gazebo is built around a client/server architecture that allows any machine to connect to a simulation over an IP network. However, the Gazebo simulator generates sensor data for each camera of the robot, which requires OpenGL acceleration to be present on the simulation server.

At the time of the VRC, few cloud providers offered machines with physical GPUs. Amazon Web Services (AWS) was initially identified as the target cloud service provider for the VRC. They offered one type of server with a GPU coprocessor that could be made to handle OpenGL off-screen rendering under Ubuntu (the cg1.4xlarge type). But the AWS machines were found to be lacking in their ability to provide consistent timing of computation.

When running computations in a distributed system, latency and jitter between components is an important consideration, especially when doing closed-loop control. For security reasons, robotics code that the teams develop runs on a separate server from the Gazebo simulator, which is protected from untrusted access during the competition. We found that when using AWS machines, communication between the simulation server and the field computers was inadequate for the competition requirements, apparently due to use of a hardware virtualization layer (which is common among cloud providers). As a result, we switched away from AWS to bare metal servers (i.e., machines having no virtualization layer) offered by SoftLayer Technologies.

Because of the geographic distribution of the teams, while all the cloud hosted machines were in a single location (Dallas, TX, USA), latency between the teams' Operator Control Unit computers and their cloud machines was also an important factor that could affect the fairness of the competition. The solution to this problem was to impose a minimum latency of 500 ms. This was achieved by configuring Linux *tc* (traffic control) settings on the router machine.

VI. The VRC Competition

The VRC competition was comprised of three tasks that were representative of the challenges faced for the DRC trials:

- Walk a short distance, enter a utility vehicle, drive along a roadway, exit the vehicle, and walk to the finish area.
- Walk across progressively more difficult terrain, from a paved parking lot to a muddy pit to a rubble field.
- Connect a hose to a spigot and open the spigot by turning a valve.

Fig. 3 (left) shows a snapshot of the Gazebo simulator with Atlas and the vehicle used during the driving task. Fig. 3 (center) captures a snapshot of the walking task, where Atlas is navigating to the gate while avoiding rubble blocks. Fig. 3 (right) shows Atlas during the manipulation task holding the hose.

A. Rules and Scoring

For each of the three tasks, teams performed five runs, for a total of 15 runs. Each run took place with a unique configuration specifying the location of all objects in the environment, the starting position of the robot, the communications parameters (maximum uplink/downlink), contact friction properties, obstacle placement, object geometry, and other relevant parameters. The five configurations were distinct from one another;

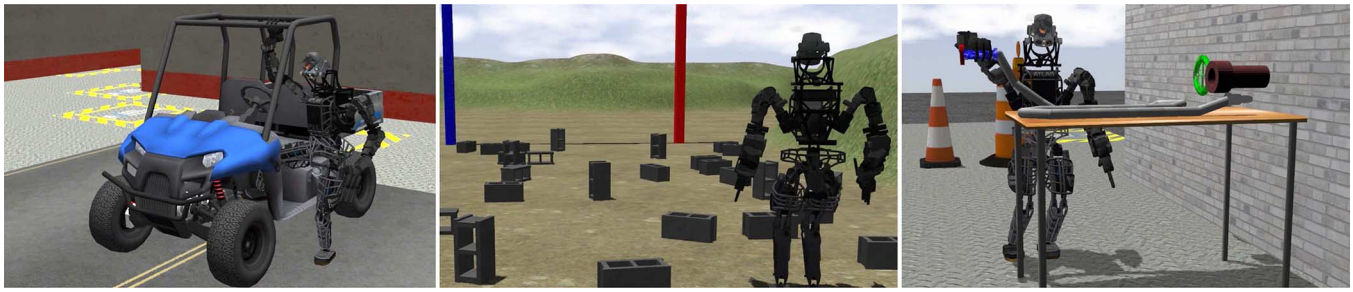

Fig. 3. Gazebo snapshots with some of the VRC tasks: Driving, walking and grasping.

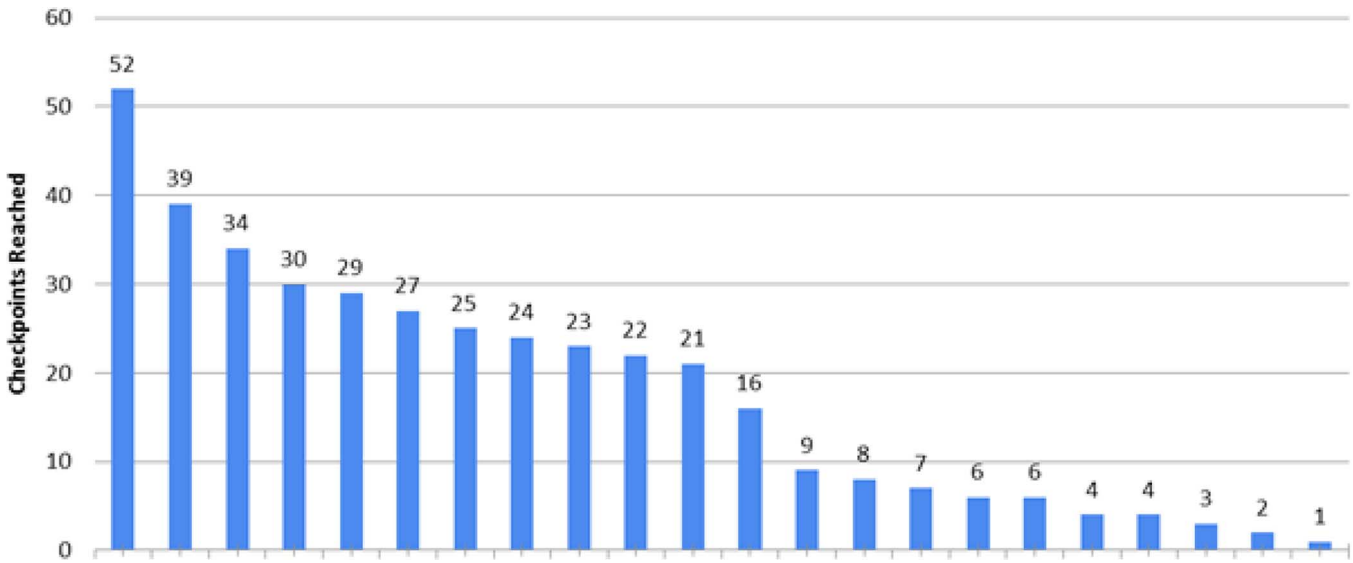

Fig. 4. Final VRC scores for all 22 participating teams.

however, each team had the same five configurations. The motivation for the task variability was to force the teams to avoid open loop behaviors and focus on solutions that can be adapted to a wider spectrum of problems. The time limit for a single run was 30 minutes.

Each task consisted of four checkpoints, such as getting into the vehicle after walking from the starting position, driving the first half of the road course, driving the second half of the road course, and walking to the finish gate after getting out of the vehicle. The robot had to complete the checkpoints in the order presented. Each successful checkpoint achieved by the team was awarded with one point. A run was considered terminated if the robot fell down three times. Teams were ranked initially by the number of checkpoints achieved. Teams with the same score were ranked using an alternating pattern: the team with the largest percentage of uplink bits remaining after all 15 runs ranked first, the team with the largest number of seconds remaining after all 15 runs ranked second, and this alternating pattern was repeated until all teams were ranked.

B. VRC Results

The results of the VRC showed strong variability between scores, which facilitated selection of the winning teams. Out of a maximum of 60 points (4 points per run for 15 runs), scores ranged from 52 points down to 1, with no duplicated scores until the 16th place team was ranked with 6 points (see Fig. 4). This meant that the scoring formulas mentioned above could look purely at the number of points awarded (along with the three fall maximum per run), as opposed to considering bits and time used. Ultimately, the top 9 teams moved forward with follow-on funding and/or provision of an Atlas robot.

The bit usage for the VRC was restricted to only 3.5 MB of uplink data across a 30 minute run for the most relaxed run configuration, and 0.014 MB for the most restrictive. This required significant changes to communications strategy as compared to conventional wired or line-of-sight wireless communications. Interestingly, although very little upload data was available, teams still managed to conserve many of their available bits, with only slight variability between high-scoring and low scoring teams. As seen in Fig. 5, the unused bits (referred to as being "banked," or saved) across all runs could be quite high, in some cases reaching 98% of the total starting bit allocation. This could be attributed to the fact that complete teleoperation is still a low bandwidth operation, but as will be discussed below, is also indicative of carefully tailored communications strategies promoting greater autonomy.

C. Analysis of the VRC Data

Inspection of the VRC data can provide insight into team performance. In particular, by reviewing the data usage history stored in the log files, information about the participants' controls strategies can be obtained. What will be shown is that

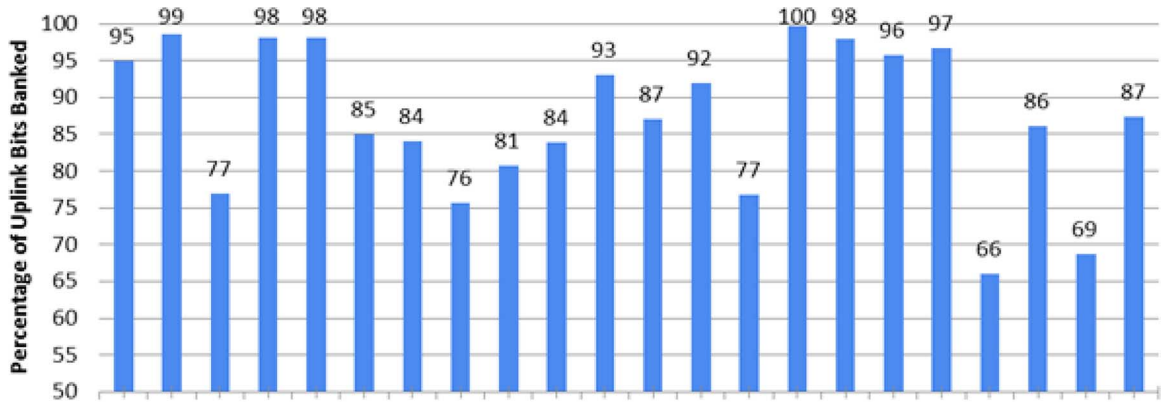

Fig. 5. Total uplink bits "banked" (conserved from starting allocation) for each team in runs scoring points. Highest scoring team at left, lowest at right.

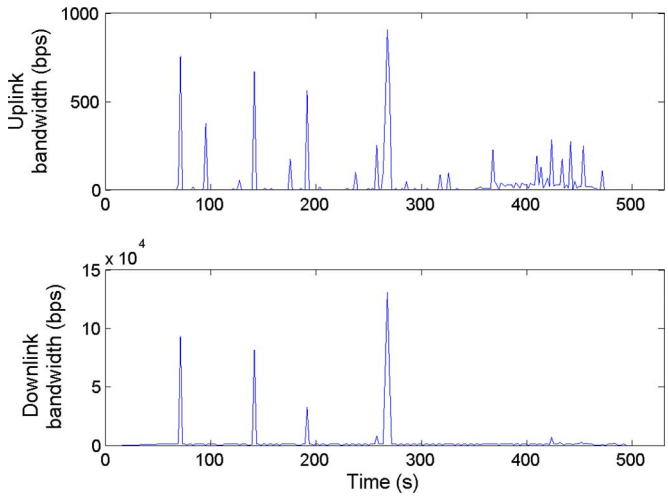

Fig. 6. Bandwidth history (2 second and 10 second averages) for a high-scoring team during one task run.

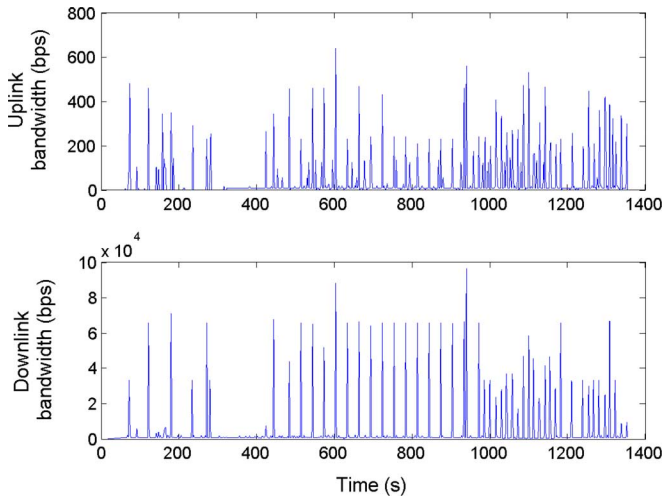

Fig. 7. Bandwidth history for a high-scoring team showing periodic intervention windows.

communications were highly bursty as opposed to steady over time, that teams varied significantly in their usage of the available communications bandwidth, and that winning teams tended to adopt a more conservative data usage strategy than lower scoring teams. Overall, the results showed that the data allocations for the runs were viable to complete the tasks, and provided insights into the means by which teams communicated with their robots. The VRC architecture, and the data that was gathered in the background through the competition, allowed for postprocessing of results to gain insight into how to set parameters for subsequent challenges. Finally, trends seen in information usage point to the importance of carefully choosing the scoring metrics and simulated restrictions to drive progress towards particular objectives, and for the Robotics Challenge program have helped to expose and quantify the tradeoffs between autonomy and information usage.

Strategies for robot communications can be seen through investigation of an approximated uplink and downlink bandwidth, obtained by averaging the bit usage over finite windows of time. The best approximation of instantaneous bandwidth can be seen on 2-second averages of the data usage, as shown in Figs. 6 and 7. These two figures show two of the three common themes in data usage. Fig. 6 represents performance of a high-scoring team during a run with the hose connection task, and shows the common trend among high-scoring teams to have highly bursty communications. Runs such as this demonstrated long periods with little or no uplink and, in the best cases, downlink as well, with almost all data transferred occurring over a finite number of discrete communications bursts. Bursty communication was brought about by two sets of circumstances: unexpected behavior (i.e., a fallen robot) requiring complex maneuvers, or high-risk operations such as manipulation of the hose nozzle to prevent it from slipping from a grasp. In this figure, at the 270-second mark a review of operator footage revealed that data was being used to send complex signals to the robot, most likely the high degree of freedom hand, in conjunction with a high volume of information back to the operator to confirm the grasp behaviors. This mode was correlated with high degrees of robot autonomy except for contingency operations, and was seen in some form for all nine of the winning teams.

Fig. 7 shows the second trend seen for a variety of teams, which was to use periodic or frequent bursty communications. Periodic or routine communications with the robot were indicative of fact-checking and repeated human-in-the-loop commands at a frequency related to the gait of the Atlas robot, with

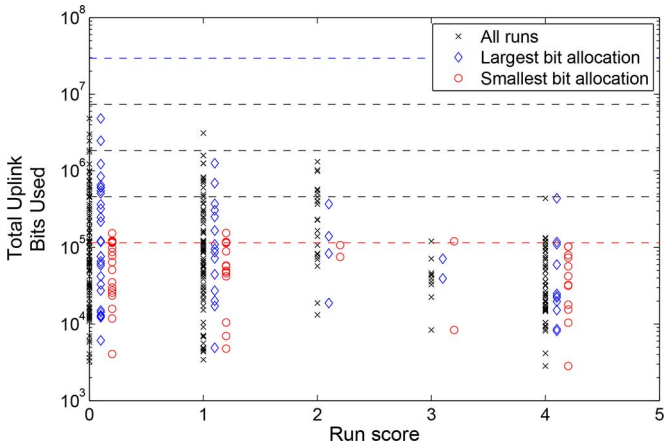

Fig. 8. Total uplink bit usage for all runs by score, with overlay of bit allocation thresholds (dotted lines).

periodic strategies occurring more with walking and hose grabbing runs than with driving runs. A hypothesis for this is that the complex maneuver of vehicle ingress was often highly automated, and as it could not be teleoperated, the need for periodic communications was less valuable. Of the top nine teams, all but one demonstrated this form of communication, which indicated more routine and/or low-level operator intervention.

The third trend in data usage strategy, seen predominantly among low scoring teams, was one of a steady-state usage of data in both uplink and downlink. By comparison with runs from high-scoring teams, these runs consistently had large and sustained data uplink and downlink rates, in some cases two orders of magnitude higher than what high-scoring teams would use. This observation indicates that high-scoring teams were able to put effort into carefully manipulating their data structure to drive down steady-state data usage rates, whether through more autonomous operations with less sensor feedback to the operators or through more careful data handling. Winning participants confirmed after the VRC that packet shaping was one of their strategies, and based on feedback about the level of effort required for these hand-optimized commands, it was decided to increase the bandwidth cap for the DRC Trials to avoid focus on data structures and transport overhead, and to instead focus more predominantly on autonomous behavior development.

Regardless of a team's intervention-based, periodic, or continuous communication strategy, most teams were able to get well below the competition's data thresholds. For uplink usage, 75% of all scoring runs used less than 10% of the uplink bits they had available, and 43% of the runs used less than 1%. For downlink, the trend was slightly more information-hungry: 55% of the runs used less than 10% of their downlink bits, and only 9% used less than 1%. Analysis showed that higher scoring runs tended to operate within the smallest bit allocation limits regardless of what bit allocation was actually available for that run, as opposed to lower scoring runs which scaled their information usage policy based on what data allocation was available by run. As seen in Fig. 8, uplink bit usage is more uniform and conservative for the high-scoring runs on the right, with a more greedy information usage policy with greater spread for lower scoring runs at left. The trends in conservative data usage for high-scoring teams is correlated with greater autonomy demonstrated and corresponding intervention-based controllers, but also confounded by the fact that usage of uplink data was penalized in the scoring, discouraging information-hungry approaches.

VII. Discussion

The VRC provided a number of critical insights feeding into the DRC Trials, which were completed in December, 2013. First, the VRC results prompted the organizers to adjust the difficulty of the physical tasks upon witnessing the challenge in vehicle ingress. Since a team could earn zero points by failing to reach the first checkpoint, the success criteria for the Trials event were adjusted to begin the driving task with the robot already seated in the vehicle.

Next, it provided insight into the usage of available bandwidth. Most teams did not use anywhere near the total amount of uplink or downlink bandwidth available to maintain consistency over maximum utilization of information, although this was remarked to be a preferable strategy for some teams. The VRC provided a baseline measure of bandwidth usage, although the simplified texture mapping allowed for higher than realistic data compression capabilities. It was projected that for a physical event such as the DRC Trials, bandwidth values approximately 100x greater than those experienced in the VRC would be required, which precipitated the decision to increase the bandwidth from values around 1000 bps to 100 kbps and higher. Also, the push for minimal data usage combined with the high latency values imposed on the teams promoted higher levels of autonomy for robot actions, requiring operator control schemes such as error mitigation rather than pure teleoperation. While this was the desired outcome, the tradeoff between maximum information gain and minimal bandwidth usage was not one considered by the organizers, and studies on effective bandwidth usage remain an open research area.

Overall, the results of the VRC substantiated the claim that viable robot programming teams can be identified through a virtual competition, in lieu or in advance of physical testing. The competition serves as a future incentive to promote both cloud-based competitions as well as software-specific competitions, with a strong potential for cost-savings and the ability to pull in a wider array of competitors with a lower barrier to entry.

Looking at the results, we are confident about the VRC's contribution. At the Trials, which saw sixteen teams compete, five of the top eight teams were previously top finishers in the VRC (the top eight from the Trials qualified for continued DARPA funding). And their relative performance at the Trials almost matched their relative performance in the VRC:

Team	DRC Trials finish	VRC finish
IHMC Robotics	2nd	1st
MIT	4th	3rd
TRACLabs	6th	4th
WRECS	6th	2nd
TROOPER	8th	8th

This means that a majority of the top teams at the Trials had first competed and won in simulation, and were then able to transition their software from simulation to hardware in a very short period of time (they had also received a very capable Atlas robot, which surely contributed to their success).

The VRC was designed to push the boundary between teleoperation and on-board autonomy by intentionally degrading the quality of the communication channel used for teleoperation. Controlling a robot over degraded communication channels is a necessary skill when operating robots in disaster scenarios; it also drives the need for more autonomous capabilities on the robot itself as effective teleoperation over degraded communication channels requires more on-board autonomy. As seen during the VRC, the winning teams demonstrated that the required tasks can be achieved with less bandwidth usage between the cloud and the OCU and more autonomy on the simulated cloud robot. In fact, the better performing teams in the VRC used less bandwidth than other teams, as shown in Fig. 8.

CloudSim was built with the VRC in mind, and a lot of effort was put into making it reliable for the purpose of the contest. There are a few changes that could benefit CloudSim and help its adoption outside of the VRC. CloudSim relies on a scoring subcomponent which contains custom code that is very specific to the VRC worlds, and is not reusable with other robotic simulations. Another limitation is that a Gazebo workstation is required to visualize a cloud simulation in 3D. Finally, the provisioning system in CloudSim is custom-built, rather than relying on an existing provisioning system. These limitations are being addressed as we are starting to use CloudSim in new projects, especially in the e-learning applications where on-demand simulation worlds with automated evaluation can help students learn about robotics through their browser. A WebGL front-end for the simulations called Gzweb has been created in a separate project.

Cloud usage for real-time robotic simulation is novel, and a useful tool for robot control development: Traditional cloud simulation consists of mostly high fidelity non-real-time simulations, such as finite-element method (FEM) and computational fluid dynamics (CFD), using the fire-and-forget model where the engineers/scientists "fire-up" jobs in the cloud and come back a few days/weeks/months later to collect data. On the other hand, the VRC demonstrated the feasibility of leveraging cloud simulation resources for onboard-robot-capabilities and operator-controller-interface developments. To control robots over high latency, low bandwidth, degraded communication network, a cloud-based real-time robot simulation platform such as CloudSim is a developer tool for exactly this scenario.

Gazebo and CloudSim have a promising future. Gazebo is constantly evolving with new features. As an example, Digital Elevation Terrains, logging capabilities, video creation/edition, as well as better support for Mac OS have recently been integrated. Improving the physics has been an ongoing effort in Gazebo and we will continue pushing the limits with new physics engines and optimizations for trying to improve the combination of fidelity and performance. CloudSim was born during the VRC but is maturing quickly. We currently support multiple cloud providers, and we are in the process of creating a simplified CloudSim user interface. This will allow simulation users to launch different simulation tasks with specific robots and environments inside each simulation. We believe that this feature could change the way simulation is used, because the users will not deal with the complexity of installation and maintenance of these tools anymore. Researchers, professors, or hobbyists will be able to explore this technology at a lower barrier to entry.

ACKNOWLEDGMENT

The authors would like to thank all of the VRC participants for their valuable work and constant feedback during the contest.

REFERENCES

[1] Defense Advanced Research Projects Agency (DARPA), "DARPA Robotics Challenge (DRC) and Virtual Robotics Challenge (VRC)." 2015. [Online]. Available: http://theroboticschallenge.org/about

[2] Boston Dynamics, "Boston dynamics atlas robot." 2015. [Online]. Available: http://bostondynamics.com/robot_Atlas.html

[3] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot Syst. (IROS)*, Sep. 2004, vol. 3, pp. 2149–2154.

[4] Open Source Robotics Foundation, "CloudSim." 2015. [Online]. Available: http://gazebosim.org/wiki/CloudSim

[5] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Winning the DARPA grand challenge," *J. Field Robot.* vol. 23, no. 9, pp. 661–692, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1002/rob.v23:9

[6] C. Urmson, J. Anhalt, J. A. D. Bagnell, C. R. Baker, R. E. Bittner, J. M. Dolan, D. Duggins, D. Ferguson, T. Galatali, H. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, A. Kelly, D. Kohanbash, M. Likhachev, N. Miller, K. Peterson, R. Rajkumar, P. Rybski, B. Salesky, S. Scherer, Y.-W. Seo, R. Simmons, S. Singh, J. M. Snider, A. T. Stentz, W. R. L. Whittaker, and J. Ziglar, Tartan racing: A multi-modal approach to the DARPA urban challenge Robotics Inst., Pittsburgh, PA, USA, Tech. Rep. CMU-RI-TR, Apr. 2007.

[7] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada, "Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 1999, vol. 6, pp. 739–743.

[8] H. L. Akin, N. Ito, A. Jacoff, A. Kleiner, J. Pellenz, and A. Visser, "RoboCup rescue robot and simulation leagues," *AI Mag.*, vol. 34, no. 1, pp. 78–86, 2013.

[9] TORCS, "TORCS, The open racing car simulator." 2015. [Online]. Available: http://torcs.sourceforge.net

[10] Cyberbotics, "Robostadium, online robot soccer competition," [Online]. Available: http://www.robotstadium.org

[11] TopCoder, "TopCoder." [Online]. Available: http://www.topcoder.com

[12] T. Berglund, A. Brodnik, H. Jonsson, M. Staffanson, and I. Soderkvist, "Planning smooth and obstacle-avoiding b-spline paths for autonomous mining vehicles," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 1, pp. 167–172, Jan. 2010.

[13] K. H. Cho, Y. H. Jin, H. M. Kim, H. Moon, J. C. Koo, and H. R. Choi, "Caterpillar-based cable climbing robot for inspection of suspension bridge hanger rope," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2013, pp. 1059–1062.

[14] J. Vilela, Y. Liu, and G. Nejat, "Semi-autonomous exploration with robot teams in urban search and rescue," in *Proc. IEEE Int. Symp. Safety, Security, and Rescue Robot. (SSRR)*, Oct. 2013, pp. 1–6.

[15] H. Ding, J. Heyn, B. Matthias, and H. Staab, "Structured collaborative behavior of industrial robots in mixed human-robot environments," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2013, pp. 1101–1106.

[16] A. Bonchis, E. Duff, J. Roberts, and M. Bosse, "Robotic explosive charging in mining and construction applications," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 1, pp. 245–250, Jan. 2014.

[17] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 398–409, Apr. 2015.

[18] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft, "RoboEarth," *IEEE Robot. Autom. Mag.*, vol. 18, no. 2, pp. 69–82, 2011.
[19] L. Riazuelo, J. Civera, and J. M. M. Montiel, "C^2TAM: A cloud framework for cooperative tracking and mapping," *Robot. Auton. Syst.* vol. 62, no. 4, pp. 401–413, Apr. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2013.11.007
[20] B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg, "Cloud-based robot grasping with the Google object recognition engine," in *Proc. IEEE ICRA*, 2013, pp. 4263–4270.
[21] Carnegie Robotics, "MultiSense-SL datasheet." 2013. [Online]. Available: http://www.theroboticschallenge.org/local/documents/MultiSense_SL.pdf
[22] M. Quigley, C. Salisbury, A. Y. Ng, and K. K. Salisbury, "Mechatronic design of an integrated robotic hand," *Int. J. Robot. Res.,* 2014. [Online]. Available: http://ijr.sagepub.com/content/early/2014/02/17/0278364913515032, published online before print.
[23] 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.
[24] Open Source Robotics Foundation, "DRC simulator," 2015. [Online]. Available: https://bitbucket.org/osrf/drcsim/wiki
[25] C. Rapier and B. Bennett, "High speed bulk data transfer using the SSH protocol," in *Proc. 15th ACM Mardi Gras Conf.: From Lightweight Mash-Ups to Lambda Grids: Understanding the Spectrum of Distributed Computing Requirements, Applications, Tools, Infrastructures, Interoperability, and the Incremental Adoption of Key Capabilities*, New York, NY, USA, 2008, pp. 11:1–11:7 [Online]. Available: http://doi.acm.org/10.1145/1341811.1341824, ser. MG'08, ACM
[26] R. Smith, "Open dynamics engine ODE. Multibody dynamics simulation software," 2015. [Online]. Available: http://www.ode.org
[27] E. Coumans, "Bullet physics engine for rigid body dynamics," 2015. [Online]. Available: http://bulletphysics.org/
[28] K. Liu and M. Stilman, "DART dynamic animation and robotics toolkit," 2015. [Online]. Available: http://dartsim.github.io
[29] M. A. Sherman, A. Seth, and S. L. Delp, "Simbody: Multibody dynamics for biomedical research," in *Procedia IUTAM Symp. Human Body Dynamics*, 2011, vol. 2, pp. 241–261. [Online]. Available: https://nmbl.stanford.edu/publications/pdf/Sherm2011.pdf,
[30] A. Jain and G. Rodriguez, "Sensitivity analysis of multibody dynamics using spatial operators," in *Proc. 6th Int. Conf. Methods Models Autom. Robot.*, 2000, pp. 28–31.
[31] R. Featherstone, *Rigid Body Dynamics Algorithms*. Boston, MA, USA: Springer, 2008.
[32] T. Koolen, J. Smith, G. Thomas, S. Bertrand, J. Carff, N. Mertins, D. Stephen, P. Abeles, J. Englsberger, S. McCrory, J. van Egmond, M. Griffioen, M. Floyd, S. Kobus, N. Manor, S. Alsheikh, D. Duran, L. Bunch, E. Morphis, L. Colasanto, K.-L. H. Hoang, B. Layton, P. Neuhaus, M. Johnson, and J. Pratt, "Summary of team IHMC's virtual robotics challenge entry," in *Proc. IEEE-RAS Int. Conf. Humanoids Robots*, 2013.

Carlos E. Agüero received the M.S. degree in computer science and the Ph.D. degree from Universidad Rey Juan Carlos, Madrid, Spain.

He is a Research Engineer with the Open Source Robotics Foundation, Mountain View, CA, USA. His research has been focused on improving robotics simulation, multi-robot object localization, task allocation, and multitarget object localization. He has been an active member of the SPL RoboCup Community since 2005. He co-developed a complete robot architecture for the Nano robot and applied it to robot soccer.

Nate Koenig received the B.S. degree in computer science from the Rochester Institute of Technology, Rochester, NY, USA, and the M.S. degree in computer science and the Ph.D. degree from the University of Southern California, Los Angeles, CA, USA. His Ph.D. work focused on Bayesian approaches to Learning from Demonstration (LfD) and improved interfaces for human robot communication approaches to LfD and improved interfaces for human robot communication.

He is the CTO of the Open Source Robotics Foundation (OSRF), Mountain View, CA, USA. Prior to joining OSRF, he was a Research Engineer at Willow Garage, where he conducted human robot interaction studies and continued development of Gazebo, and open source robot simulator.

Ian Chen received the B.E. and Ph.D. degrees from the Electrical and Computer Engineering Department, University of Auckland, Auckland, New Zealand, in 2007 and 2011, respectively. His Ph.D. research focused on applying mixed reality techniques to enhance robot simulations, creating a world where robots are capable of sensing and interacting with virtual objects.

He is Research Engineer with the Open Source Robotics Foundation, Mountain View, CA, USA. He also developed augmented reality systems for real-time visualization of robot sensory and task-relevant information.

Hugo Boyer received a Mechanical Engineering degree from Ecole Polytechnique of Montreal, Montreal, QC, Canada.

He is a Software Engineer with the Open Source Robotics Foundation, Mountain View, CA, USA. He has over 15 years of professional development experience. At Makerbot Industries, he created Miracle-Grue, Makerbot's first tool path engine fA, USAr 3D printers. At McGill University, he worked on parallel GPU algorithms for the computational chemistry lab. He is also an Architect of SIMSAT, the real-time spacecraft simulator of the European Space Agency.

Steven Peters is a Software Engineer with the Open Source Robotics Foundation, Mountain View, CA, USA. He joined OSRF after completing his dissertation from the Robotic Mobility Group, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, under the direction of Karl Iagnemma. At MIT, he developed sensing and control techniques for collision avoidance and rollover prevention of passenger vehicles. He has experience with high-fidelity simulation of robot dynamics as well as implementation of real-time control algorithms on passenger vehicles and robotic manipulators.

John Hsu received the Ph.D. degree in aeronautics and astronautics from Stanford University, Stanford, CA, USA, in 2004.

He is Chief Scientist with the Open Source Robotics Foundation (OSRF), Mountain View, CA, USA. His previous work includes numerical algorithm development for solving complex unsteady transonic flows and coupled aerodynamics and finite element structural dynamics simulations. His research interests include solving accurate physics simulation of constrained rigid body dynamics at near real-time, realistic robotic perception simulation, and development of a robotic simulation platform.

Brian Gerkey received the B.S.E. degree in computer engineering, with a secondary major in mathematics and a minor in robotics and automation from Tulane University, New Orleans, LA, USA, in 1998, and the M.S. and the Ph.D. degrees in computer science from the University of Southern California, Los Angeles, CA, USA, in 2000 and 2003, respectively.

He is CEO of the Open Source Robotics Foundation (OSRF), Mountain View, CA, USA. Before, he was Director of Open Source Development at Willow Garage.

Dr. Gerkey was recognized by *MIT Technology Review* with the TR35 Award in 2011.

Steffi Paepcke received the B.S. degree in psychology from the University of California, Santa Cruz, CA, USA, and the M.Sc. degree in human-computer interaction from Carnegie Mellon University, Pittsburgh, PA, USA.

Before pursuing her graduate degree, she worked as a Human-Robot Interaction Research Assistant with Leila Takayama, Willow Garage. She is the Lead User Experience Designer at the Open Source Robotics Foundation, Mountain View, CA, USA. Her interests include interaction design, user research and making open source software more user-friendly.

Jose L. Rivero received the Computer Engineering degree from Carlos III University, Madrid, Spain.

He was working as a Research Support Engineer at the Institut de Robòtica (IRI). Currently, he is Building Engineer with the Open Source Robotics Foundation, Mountain View, CA, USA.. He strongly believes that open source is the best way to bring science and technology beyond its present limits. He has been contributing to free software as a Gentoo Developer and is one of the coordinators of The Humanoid Lab, a robotics and open source initiative to bring students into the robotics world.

Justin Manzo received the B.S., M.S., and Ph.D. degrees in robotics from Cornell University, Ithaca, NY, USA.

He currently works with Booz Allen Hamilton, McLean, VA, USA, as a Technology Consultant with subject matter expertise in robotics. He is the Manager of Booz Allens Center for Robotic Systems and Simulation (CRoSS), McLean, VA, USA, and develops novel unmanned platforms as well as provides integration and testing oversight for government robotics programs, to include work with simulation-based robotic testbeds.

Eric Krotkov received the Ph.D. degree in computer and information science from the University of Pennsylvania, Philadelphia, PA, USA, in 1987, for pioneering work in active computer vision.

He is the President of Griffin Technologies, Wynnewood, PA, USA, a consulting firm specializing in robotics and machine perception. Before founding Griffin Technologies, Wynnewood, PA, USA, he worked in industry as an executive in a medical imaging technology start-up, in government as a program manager at DARPA, and in academia as a faculty member of the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA.

Gill Pratt received the Doctorate degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA.

He is a Program Manager with the DARPA Tactical Technology Office, Arlington, VA, USA, and runs the DRC Program. His primary interest is in the field of robotics: declarative design methods that enhance the symbiosis between designer and design tool, rapid fabrication methods, interfaces that enhance human/machine collaboration, mechanisms and control methods for enhanced mobility and manipulation, innovative actuators, and the application of neuroscience techniques to robot perception and control.