David et al. Journal of Internet Services and Applications 2013, 4:16
http://www.jisajournal.com/content/4/1/16

Journal of Internet Services and Applications
a SpringerOpen Journal

RESEARCH Open Access

A DDS-based middleware for scalable tracking, communication and collaboration of mobile nodes

Lincoln David*, Rafael Vasconcelos, Lucas Alves, Rafael André and Markus Endler*

Abstract

Applications such as transportation management and logistics, emergency response, environmental monitoring and mobile workforce management employ mobile networks as a means of enabling communication and coordination among a possibly very large set of mobile nodes. The majority of those systems may thus require real-time tracking of the nodes and interaction with all participant nodes as well as a means of adaptability in a very dynamic scenario. In this paper, we present a middleware communication service based on the OMG DDS standard that supports on-line tracking and unicast, groupcast and broadcast with several thousand mobile nodes. We then show a Fleet Tracking and Management application built using or middleware, and present the performance results in LAN and WAN settings to evaluate our middleware in terms of scalability and robustness.

Keywords: Mobile communication; Middleware; Adaptability; DDS; Collaboration; Scalable communication

1 Introduction

Advances in mobile communication, GPS positioning and sensor technology networks are some of the driving forces pushing computing to mobile-networked systems, enabling new services and applications. Many current distributed systems such as transportation and logistics, emergency response, environmental monitoring, homeland security and mobile workforce management employ mobile networks as a means of enabling communication, collaboration and coordination among the mobile nodes, which might be people, vehicles [1,2] or autonomous mobile robots [3,4]. With the rapid increase of embedded mobile devices, many such applications are faced with the challenge of supporting several thousands of nodes, requiring both real-time tracking of their context/location information and efficient means of interaction among all nodes. Moreover, in many of these applications, the set of participating mobile nodes can vary constantly, as nodes may join and leave the system at any time, either due to application-specific circumstances or because of intermittent wireless connectivity. Such large-scale mobile systems thus require a scalable communication infrastructure that supports reliable and almost instantaneous data and context dissemination between all mobile nodes [5] as well as monitoring and dynamic adaptation capabilities that enable automatic adjustment of the infrastructure to the very dynamic load demand caused by the mobile nodes. In this paper, we present a scalable communication middleware that addresses most of these requirements. We also present a Fleet Tracking and Management application built using our middleware and show performance results of our middleware for thousands of mobile nodes, both for Local Area and Wide Area Network settings.

A common characteristic of the distributed mobile applications considered in our work is that the mobile nodes periodically produce data about them, i.e., context information probed from sensors. Examples of produced context information may include the node's position, speed, and ambient temperature. These produced data are then published to be processed or visualised by other nodes, which can be either stationary or mobile. We also assume that each mobile node has some wireless network interface that is capable of running the IP protocol, which in fact most current wireless networks do. In these applications, the main requirement is that if the mobile node has connectivity and is generating context data or other application messages, this data should be delivered to all other interested nodes almost instantaneously, i.e., with minimum delay. Moreover, all messages

* Correspondence: lnsilva@inf.puc-rio.br; endler@inf.puc-rio.br
Department of Informatics, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Rio de Janeiro, Brazil

addressed to the connected mobile nodes should also be delivered reliably and with minimum delay.

In the past, much research has been performed in Publish/Subscribe (Pub/Sub) [6-10], but only a few support large-scale mobile networks and simultaneously offer QoS (Quality of Service) guarantees for the mobile communications, especially the aforementioned reliability and low-latency message delivery [11-13]. On the other hand, the OMG's Data Distribution Service for Real-time Systems (DDS) standard [14,15] offers high-performance communication capabilities and is currently used for several real-world distributed mission-critical applications. DDS specifies a decentralised (Peer-to-Peer) scalable middleware architecture for asynchronous, Publish-Subscribe-like data distribution, supporting several QoS policies (e.g., best effort or reliable communication, data persistency, data flow prioritisation, and several other message delivery optimisations). Unlike traditional Publish-Subscribe middleware, DDS can explicitly control the latency and efficient use of network resources through fine-tuning of its network services, which are critical for soft real-time applications (e.g., its QoS policies deadline, latency budget or transport priority) [16]. Moreover, because Publish-Subscribe communication is widely acknowledged as being one of the most suitable paradigms for mobile systems, we were sure that DDS would be very appropriate for large-scale mobile applications.

However, despite its advantages, DDS cannot be efficiently deployed directly on mobile nodes or in wide-scale wireless networks [17-19], where the obtainable performance may become unpredictable [20]. The main reasons for this problem are the extensive use of IP multicast in DDS domains, the lack of proper mechanisms to handle intermittent connectivity and IP address variability, and the that resource-limited (mobile) devices cannot perform well as DDS peers because they must cache and route data for other peers. These mobile-specific limitations of DDS motivated us to design and implement a middleware that extends DDS' high-performance communication capabilities to wireless-connected mobile devices. Another limitation of current DDS implementations is their poor support for deployment and efficient data exchange among nodes in Wide-Area-Network (WAN) settings, which is the predominant scenario in current cellular network services. As the most important requirements of our middleware, we considered scalability, simplicity and high communication performance, even in the presence of intermittent connectivity, handovers and slower wireless communications.

The main contributions of this paper are the following:

1. We present our DDS-based communication middleware, give evidence of its scalability, and show how it supports efficient and reliable unicast, groupcast and broadcast message delivery to mobile nodes regardless of IP address changes, temporary disconnections, and Firewall/NAT traversal.
2. We describe support for two types of node groups, explicit and implicit, or context-defined groups, and we show how the latter are efficiently computed/updated in our middleware.
3. We show how the communication workload can be balanced among Gateways that are special DDS nodes responsible for acting as a bridge between the DDS domain and the mobile communication protocol. We also show how the system supports reliable message delivery, even in the presence of frequent handovers of mobile nodes among the Gateways.
4. We also present the results of several performance tests made in LAN and WAN settings, showing the apparent suitability of our middleware for context information distribution and communications in large-scale mobile applications with thousands of nodes.

This work is part of a larger project called ContextNet [21,22], which aims to develop middleware for (soft)-real-time communication, coordination and collaboration in large-scale distributed mobile applications. Within the scope of this project, the middleware presented in this paper is the basic layer for communication and context information sharing. This middleware, called the Scalable Data Distribution Layer (SDDL), is available for download at www.lac-rio.com/sddl.

Paper outline: In the next section, we present the goals and the main characteristics of the SDDL communication layer. In section 3, we describe a Fleet Tracking and Management application, and in section 4 we present results on performance tests. Section 5 discusses related work on scalable middleware for such mobile systems, and in section 6 we argue the benefits of our system. Finally, in section 7 we draw conclusions and point to future work.

2 Overview of the Scalable Data Distribution Layer (SDDL)

Scalable Data Distribution Layer (SDDL) is a communication middleware that connects stationary DDS nodes in a wired "core" network to mobile nodes with an IP-based wireless data connection. Some stationary nodes are information and context data processing nodes, others are gateways for communication with the mobile nodes, and yet others are control nodes operated by system managers. A control node (or Controller) is used to display all the mobile nodes' current positions (or any other context information), manage groups of nodes, and send unicast, broadcast, or groupcast messages to the mobile nodes (MNs). Figure 1 shows these types of nodes within the context of an implemented Fleet Management Application.

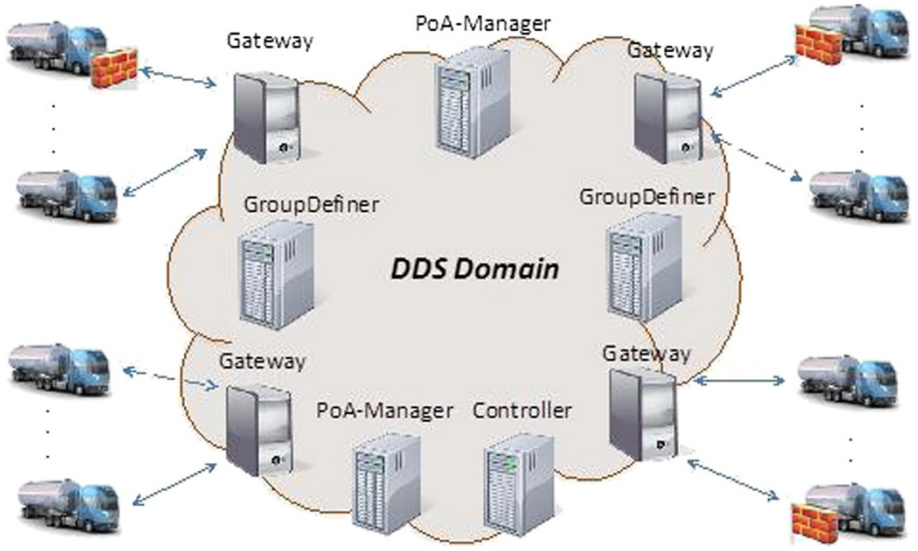

Figure 1 Fleet management application architecture.

The Scalable Data Distribution Layer (SDDL) employs two communication protocols: DDS' Real-Time Publish-Subscribe RTPS Wire Protocol [16] for wired communication within the SDDL core network, and the Mobile Reliable UDP protocol (MR-UDP) for inbound and outbound communication between the core network and the mobile nodes. The core elements rely on the DDS Data Centric Model, where DDS Topics are defined to be used for communication and coordination between these core nodes. The MR-UDP protocol will be explained in section 2.1. As part of the core network, there are three types of SDDL nodes with distinguished roles:

The *Gateway (GW)* defines a unique Point of Attachment (PoA) for connections with the mobile nodes. The Gateway is thus responsible for managing a separate MR-UDP connection with each of these nodes, forwarding any application-specific message or context information into the core network and, in the opposite direction, converting DDS messages to MR-UDP messages and delivering them reliably to the corresponding mobile node(s). Being the handler of connections to the mobile nodes (MNs), the Gateway is also responsible for notifying other SDDL core network nodes when a new MN becomes available or when MNs disconnect from it. This information is necessary for implementing other SDDL core nodes, such as nodes that cache messages addressed to temporary offline mobile nodes for later delivery.

The *PoA-Manager* is responsible for two tasks: to periodically distribute a list of Points of Attachments (PoA-List) to the MNs and to eventually request that some MNs switch to a new Gateway/PoA. The PoA-List is always a subset of all available Gateways in SDDL, and the order in the list is relevant, i.e., the first element points to the preferred Gateway/PoA and so forth. By having an updated PoA-List, an MN may always switch its Gateway if it detects a weak connection or a disconnection with the current Gateway. Moreover, by distributing different PoA-Lists to different groups of mobile nodes, the PoA-Manager is able to balance the load among the Gateways as well as announce to the mobile nodes when a new Gateway is added to or an existing Gateway is removed (or failed) from the SDDL core.

GroupDefiners are responsible for evaluating group-memberships of all mobile nodes. To do so, they subscribe to the DDS topic where any message or context update is disseminated (e.g., those sent by mobiles and forwarded by the corresponding Gateway), and they map each node to one or more groups according to an application-specific group membership processing logic. This group membership information is then shared with all Gateways in the SDDL core network using a specific DDS Topic for control, to which all Gateways subscribe so they can update their cached mobile node's membership information. Whenever a new message is sent to a group, each Gateway queries its group-to-MN mapping to know to which of the connected MNs it must send the message. The current groups of a node can be determined, for example, by its node ID, its current position (e.g., if it is inside some region), or by any other attribute/field of its context information (e.g., a node's energy level). In any case, it is important to note that the logic to define the groups is always application-specific, is to be implemented by the application developer and is added to the GroupDefiner as a *Group-selection module* plug-in.

Figure 2 shows all the types of SDDL nodes and the communication protocols they use. On the mobile side, a mobile client app – currently, we support only Android mobile apps - uses a ClientLib (CNCLib) for establishing and managing MR-UDP connections and sending and receiving application-specific messages to/from a Gateway or other mobile node. At the SDDL core network side, all nodes use a DDS implementation-independent Universal DDS Interface (UDI), whose API classes and methods are mapped to the different primitives for setting up and configuring the communication entities of each DDS product, which in turn uses the DDS standard, high-performance RTPS protocol. The Gateways are the only nodes in the SDDL core that also use the CNCLib to manage (an arbitrary number of) mobile connections. As seen, the GroupDefiners, PoA-Manager and Gateways are all Publishers and Subscribers of several DDS Control or Application Topics, by which they are able to interact with each other for processing and classifying data transmitted to/from the mobile nodes. The *Controller* is a Java Applet that interacts with a JavaScript for displaying all MNs' current locations on a map using a Web browser window. The Web browser also displays the current groups of MNs (which can be defined and managed by the user) and supports operations to send (uni-/group-/broadcast) messages to, as well as receive text messages from, the MNs. Most of the elements shown in Figure 2 will be explained in mode detail in the remainder of this section.

2.1 MR-UDP

The Mobile Reliable UDP (MR-UDP)[a] protocol is the basis for the Gateway-mobile node interaction. This protocol implements TCP-like functionality at the top of UDP and has been customised to handle intermittent connectivity, Firewall/NAT traversal and robustness to changes of IP addresses and network interfaces. Each message, in either direction, requires an acknowledgement that, if not received, causes each transmission to be retried several times before the connection is considered broken. In addition, MR-UDP implements the following optimisations: a reduced number of connection-check packets; the transparent continuation of an MR-UDP connection regardless of IP address changes; a small number of connection maintenance packets for Firewall/NAT traversal; and simple data-flow control. Because the mobile device has its own restrictions, such as limited battery life, it is also important that the communication protocol not use too much processor resources. These optimisations are very important because cellular wireless networks are not fully reliable everywhere, and resources must be used wisely and only when truly necessary. For example, when a mobile node connected to a Gateway enters an area with no, or weak, connectivity, it may suffer a temporal disconnection; and when the signal comes back, the device will most likely have obtained a new IP address. In our MR-UDP implementation, the previous connection to the mobile node will be maintained, and all buffered UDP packets will be delivered in the original order if the disconnection time is shorter than a threshold timeout.

2.2 Handling mobile node handover

A Handover (HO) happens when a mobile node connected to a Gateway drops or loses its connection and connects itself to a different Gateway. SDDL supports

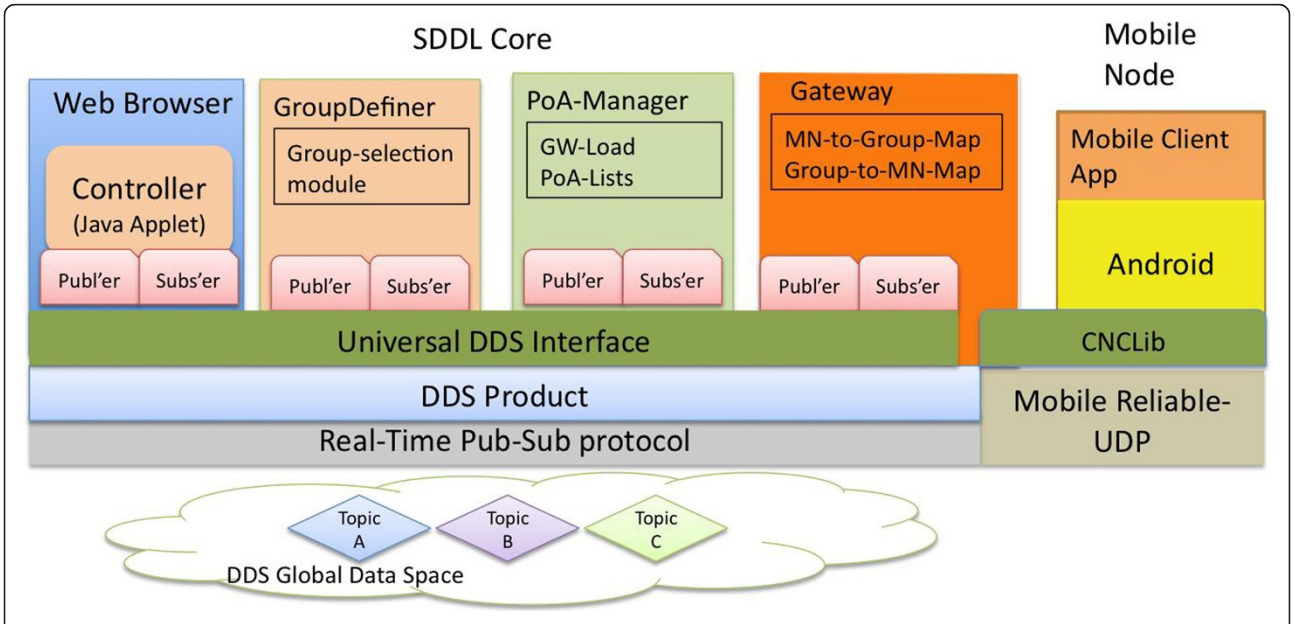

Figure 2 Nodes and protocols used in SDDL.

both Core-initiated Handover, i.e., when a mobile node is requested by the PoA-Manager to connect to a new GW, and Mobile-initiated Handover, i.e., when the mobile node spontaneously decides to connect to a new GW. In either case, it is the mobile node that actually chooses another PoA from its PoA-List and reconnects to the corresponding GW.

While performing a handover between Gateways, i.e., during the period when the node is temporarily disconnected, it is possible that some messages will fail to be delivered to it. To enable the reliable delivery of messages during a handover (a.k.a. *smooth handover*), SDDL also supports the *Mobile Temporary Disconnection* (MTD) Service, which can run on any node(s) of the SDDL core network.

The MTD Service is responsible for listening to *disconnected-MN* messages produced by Gateways and, thereafter, to collect all messages that could not be delivered to the mobile node during its HO or offline period. However, as soon as the node is connected to a new GW, which will also be announced by the corresponding GW, the MTD Service will resend all the buffered messages through the DDS domain to deliver them to the node through the new GW. Because not all applications require such reliable delivery, the MTD service is optional in SDDL, and of course, the buffering capacity of MTS is limited by the amount of memory allocated to it at deployment time. Thus far, we have not implemented any specific garbage-collection algorithm for minimising message loss due to buffer overflow.

2.3 Node Ids and message delivery

In SDDL, each mobile node and each Gateway has a unique identifier (ID). While MR-UDP messages carry only the mobile node's ID, the ID of the GW currently serving the mobile node is automatically attached to any message (or context update) entering the SDDL core network. Thus, any corresponding node can learn which is the mobile's current GW, and most messages addressed to a mobile node will thus also carry a Gateway ID, allowing them to be directly routed only to the corresponding Gateway via DDS content filtering. However, if the mobile node suddenly becomes unreachable/disconnected, its most recent Gateway will notify all other nodes in the SDDL core network of this issue; thus, the Gateway ID will be omitted in future messages to the mobile node. However, in cases where the current Gateway of a mobile node is not known or specified, messages addressed to the mobile node will still be delivered because they will be received by all Gateways.

As part of SDDL's basic functionality, it is possible to send two types of messages: unicast messages to a specific node or broadcast messages to all active nodes. Messages can be sent by a SDDL-specific node within the core network, by an arbitrary application node in the core network, or by any mobile node. The communication from a mobile node to another mobile node is achieved by using the mobile nodes´ GWs as *brokers* to deliver the message from the sender to the receiver through the SDDL core. SDDL was designed to be a robust, high-performance message exchange middleware even under high load periodic context/location update messages from all mobile nodes. The main goal is to offer a scalable communication infrastructure for the development of collaborative mobile applications.

Applications developed using SDDL can also include groupcast communication, whose group-definition logic is processed at the GroupDefiner nodes and the group-membership information is disseminated to all Gateways, by which they are able to update their MN-to-Group and Group-to-MN mappings, as mentioned previously and will be explained in more detail below.

2.4 Group communication and management

Groups of nodes may be either long-lived/explicit or context-defined. In the former category, they are explicitly defined by the application developer/operator, e.g., nodes belonging to a certain user group, to the same company or administrative domain, or to nodes of the same type. For context-defined groups, the membership of a node is dynamically determined by its most recently updated context data (its ContextUpdate – CxtU). For example, if the context means the "geographic position", then all nodes located within a certain region (e.g., a metropolitan area or within the boundaries of a state), can form a context-defined group. Alternatively, nodes could also be grouped by their current type of connectivity (3G vs 2G), their residual energy level, accelerometer data, local weather condition, or any other dynamic context information. Hence, context-defined group membership has to be continuously updated according to the most recent CxtU sent by the nodes, which is performed by the GroupDefiners in tandem with the Gateways: for each CxtU, the GroupDefiners check whether some membership changed and, if such is the case, disseminate this node's group change to all Gateways, which update their mappings accordingly.

Each GroupDefiner internally consists of a generic CxtU message processing part, and an application-specific, *Group selection module*. The generic part is responsible for reading CxtU messages from the DDS domain, recording the current groups related to the message, and handling the CxtU object to the Group Selection module. This module will execute a specific group-mapping algorithm to determine the group/s that the corresponding producer of the CxtU is a member of and must be implemented accordingly with any application-specific rule.

This split between the generic and specific group membership processing parts has certain advantages: (i) it is possible to deploy several GroupDefiners in the SDDL

core, each of which execute a Group selection module that examines a certain type of CxtU object independently of the other modules; and (ii) Group selection modules can be easily exchanged in the GroupDefiner without compromising the remaining function of the SDDL group management and communication capabilities.

2.5 Universal DDS interface and general application topics

The Universal DDS Interface (UDI) is a library that fully abstracts the DDS implementation utilised, promoting the reusability and interoperability of SDDL components. The main goal is to hide away the idiosyncrasies of the APIs of each DDS implementation and simplify the set-up and configuration of DDS entities.

UDI supports the creation of DDS topics (and content-filtered topics), Domain Participants, Publishers, Subscribers, Data Readers and Data Writers, as well as the definition of QoS policies for each such entity, all in a straightforward and uniform way. As mentioned previously, the DDS standard defines 22 possible QoS policies [16,23], but each DDS vendor may have different behaviours and contracts associated with each policy implementation as well as different ways of configuring the corresponding network services, which makes the proper use of QoS of DDS a cumbersome task. Moreover, there are several DDS products that only support DDS setting at build time. Hence, one of our goals in designing UDI was also to simplify this process and to support QoS setting at deployment time. In UDI, therefore, QoS policies are defined by passing a single QoS policy object at the initialisation method, which aggregates the chosen QoS parameter settings for all DDS entities in a single place. This approach bears some similarly to the concept of QoSProvider, present in C++ and Java APIs for DDS. Through UDI, whenever a DDS implementation is to be replaced or added, one needs only to implement the new UDI port to the chosen DDS implementation. UDI is also topic-independent in that it is able to manipulate any DDS topic, not only the SDDL topics. Thus far, we have implemented SDDL's UDI layer for CoreDX DDS[b] and RTI Connext[c].

As already mentioned, all SDDL core components interact through DDS topics. Some of these topics are used for control purposes, e.g., for coordination among GroupDefiners, Gateways, and PoA-Manager, while other topics are used for Application messages. For the latter topic, SDDL defines a single and generic Application Topic type that is to be used by the application programmer to create its application topics. The main components of this topic type are a content attribute, which holds any Java-serialised object, and a list of group IDs for the exchanged message. This single generic topic type makes SDDL a general-purpose communication middleware that is completely agnostic to the application-specific classes and that is responsible only for reliable and efficient message delivery to/from the mobile nodes and for the management of group memberships of the nodes.

2.6 ClientLib

ClientLib, or just CNCLib, is a detached software component of SDDL that must be used to implement the mobile client applications. CNCLib hides most communication protocol details and handles several connectivity issues with the Gateways on the SDDL core network.

Until now, CNCLib has been implemented using only the MR-UDP protocol, described in the previous sections. However, we also plan to map the CNCLib primitives to other protocols, such as HTTP, so that the developer can select the protocol that best suits the developer's application needs, as not all mobile nodes will necessarily have stringent resources and network limitations.

The CNCLib also implements and hides from the application developer all low-level SDDL protocol features, such as the handovers. In this matter, the CNCLib is responsible for handling and managing the PoA-List, deciding and performing both the mandatory and the spontaneous handover. If the application tries to send any information during the short disconnection period between handovers, the CNCLib also buffers packets and sends them as soon as a new connection is available.

When the client application is running in a very unstable network, with frequent temporary disconnections, the CNCLib tries to shield these reconnection attempts from the application so that the mobile client application may behave as if the client had a stable, continuous connection. If a new connection cannot be made, the CNCLib informs the application that there was data that could not be sent to the Gateway. All communication is asynchronous, i.e., the application is informed in a Listener when new messages have been received or when any information could not be sent. For all communication, CNCLib uses a single abstract class *Message* that must be implemented by the application developer.

The CNCLib has also a *server* part, which is used by the Gateways to wait for and handle mobile client connections. Thus, the CNCLib is also responsible for the serialisation and deserialisation of all exchanged data. CNCLib also implements other features that are hidden, such as the reception of and response to *Ping Messages*, which are used by the SDDL to collect statistics about the latency of mobile connections.

For the next version, we are implementing two additional APIs that extend CNCLib with asynchronous communication modes. The first, *Group API*, will offer methods to subscribe to group messages and to send messages to specific groups. The second, the *Pub/Sub API*, will offer a generic content-based publish-subscribe communication mode and allow the application developer to implement

applications where any mobile node may subscribe to messages and context updates produced by any other mobile node and which will use the SDDL core as its communication platform.

3 Fleet tracking and management application

SDDL has been deployed in a real-world Fleet Tracking and Management application (InfoPAE Móvel) of a major gas distribution company that operates throughout the entire country of Brazil. Using this application, the company's Operations Center is able to track trajectories of its trucks in real-time, to optimise the trucks' itineraries, to detect and give notice of obstructions or jams on roads and to monitor the vehicle driver's actions (e.g., elapsed time on both planned and involuntary stops). Moreover, it performs simple text messaging with drivers to send them instructions or alerts, both individually as well as to subgroups of the vehicle's drivers, according to the country region they are currently located. For communication with the vehicles, the company uses any of the four Brazilian cellular network operators because one or the other operator(s) better serves each region of the country. Moreover, in each region, there are significant differences in connectivity quality (e.g., 2G vs. 3G) and extension of the wireless coverage. Thus, during a long journey, vehicles may experience several IP address changes and temporary data link disconnections (due to weak coverage or handover latency). Finally, in most cases their 2G/3G connections will be behind firewalls of the cell operators.

Figure 1 shows our application architecture, with all nodes in the SDDL core network (DDS Domain) and our Fleet Tracking and Management application.

3.1 Implementation

Using the SDDL as the middleware to implement this application, the mobile nodes are represented by the company's trucks. Once connected, the mobile client at the vehicle sends up to 20 location updates (probed from the GPS sensor) every 30 seconds to the Gateway. This on-line tracking of all mobile nodes can improve the quality of collaboration among the operator at the Fleet Management Operations Center (FMOC) and the drivers, and among the drivers themselves. Because all participants can be made aware of each other's location (in fact, it could also be other context information about the truck or its environment), it is possible to react immediately to any abnormal situation and perhaps initiate a communication session with the drivers. For example, one could ask a driver why he/she has stopped or is traveling at low speed, thereby receiving information about a traffic jam or an accident, allowing other drivers to choose a different route.

As part of the fleet management system, we implemented another specific element, the Controller. The Controller runs at the FMOC and is used to display, in real-time, the vehicle's position on a map as well as to send unicast, broadcast and groupcast messages to groups of vehicles. In the current version, the Controller is a Java Applet that interacts with a JavaScript to display vehicle positions, groups and text boxes for messaging in a Web browser window.

Figure 3 shows a screenshot of the FMOC Controller (*InfoPAE Móvel Monitor*) browser window, with vehicles (blue icons) with their traces and road problems/alerts (red icons) displayed on the map as well as a "bubble window" for messaging with one specific vehicle (the green icon). On the right hand side, from top to bottom, are a section for editing and sending a message to a group of vehicles (with a group selector), a control panel for measuring round-trip delays to individual vehicles or groups of vehicles, and a window displaying a log of message exchanges.

As the mobile client for this application, we have implemented a prototype using the Android framework (version 2.3). This prototype uses the CNClib in an Android's AsynchTask to connect to a Gateway (the first in the PoA-list) and is capable of sending and receiving simple text messages to/from any other mobile and stationary nodes, including the Controller. Also, using Android's MapView, the prototype displays on a map the current vehicle's position (the green icon), other vehicles' up-to-date positions and traces (blue icons), and road problem/alerts in its vicinity (red icons) see Figure 4 for a screenshot of the client map view.

4 Performance tests and results

Thus far, we have tested our middleware only in lab experiments and not in a real-world Fleet Management application for two reasons. First, our Controller and mobile client are still prototypes and do not implement all the required Vehicle Management functionality. Second, we wanted to test our system with thousands of nodes, and performing such a large-scale deployment of the client software is currently not feasible. Therefore, we used a program to launch and simulate an arbitrary number of concurrent MNs that connect to some Gateways and periodically send their position.

The main goal of the tests was to evaluate SDDL's performance, in terms of communication latencies within the SDDL core network and on the Gateway-mobile links, both of them for unicast, broadcast and groupcast messages from the Controller to the mobile nodes.

We performed two separate tests, one with all participants' nodes and simulated MNs executed in a local area network (LAN) and another with the simulated MNs connected through a remote link on the WAN. The local area test was primarily for evaluating the SDDL

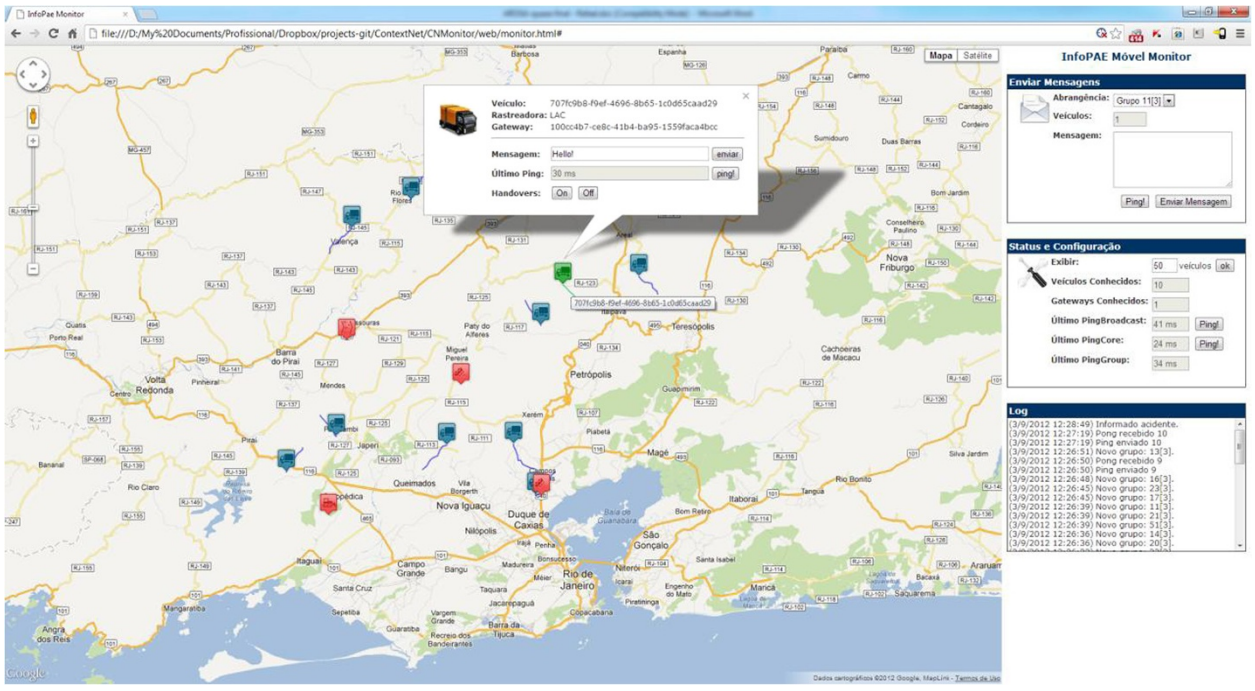

Figure 3 Fleet management application controller.

performance at serving a large number of mobile nodes with a significant amount of data being exchanged, while the wide-area tests showed the reliability of the mobile communications using the MR-UDP protocol, even in the presence of handovers. The PoA-Manager and handovers were active only in the WAN tests.

Although we tested SDDL only with simulated mobile nodes, their communication behaviour is very similar to the expected behaviour of real mobile clients, except for the lack of mobile-initiated messaging; moreover, these nodes use the same CNCLib/MR-UDP implementations. For example, in the WAN tests we also let the simulated MNs randomly disconnect from their current Gateway and try to connect to another Gateway. Therefore, the simulated MNs did in fact produce quite realistic traffic data, allowing us to measure the system's performance in a high workload scenario, i.e., with a huge volume of data exchanged between the mobile nodes and the

Figure 4 Mobile client app prototype running in android.

SDDL core nodes. Thus, we believe that analysing the system's performance graphs gives a realistic picture of SDDL's scalability and robustness.

4.1 LAN tests

The main goal of the Local Area Network experiments was to evaluate SDDL's performance under a high traffic load of LocationUpdates (i.e., Context updates) generated by thousands of mobile nodes.

4.1.1 Configuration and simulation parameters

Our mobile node simulation program, MN-Simulator, uses a thread pool with a size of 30 to indefinitely execute an arbitrary number of MNs, where each MN is scheduled to periodically send 20 simulated coordinates (pairs latitude, longitude) packed into the ClientLib Message object to one of the Gateways. Thus, the total size of this LocationUpdate (LU) message is approximately 1 KB[d]. In addition to sending LUs, each MN also receives sporadic ping messages from the Controller in the SDDL core and immediately replies with a pong message.

The performance tests were executed with following system configurations and simulation parameters: (a) 2,000, 4,000, 6,000, 8,000 and 10,000 MNs connected to each Gateway; (b) one or two Gateways; (c) LocationUpdate frequencies of 2 LU/min, 4 LU/min and 10 LU/min; and (d) one GroupDefiner.

4.1.2 Experimental setup

To test the communication performance, in each test round we connected all simulated MNs to the Gateways and then sent unicast messages to some MNs, broadcast messages to the Gateways on the DDS domain (Core) and broadcast messages to all MNs. For each type of message we calculated the round trip delay as the difference between the moments the message was sent and the moment the confirmation response was received.

Our hardware test setup comprised 4 computers (virtual and real), 2 of them running Gateways and 2 others running the MN-Simulator. The GroupDefiner was run on one of the simulation machines. All machines were connected through a 10/100 Mbps switch.

We ran experiments with most of the simulation parameters explained in the previous section. However, due to the memory and processing limitations of the machines executing the MN-Simulator, we were able to simulate at most a total of 12,000 MNs performing 10 Location updates per minute.

4.1.3 Testing unicast and broadcast

The results are presented in Figures 5 and 6. All round-trip times are shown in milliseconds. For the sake of better legibility, the subtitles were abbreviated, e.g., LU means Location Updates and 8,000v 2GW means a total of 8,000 MNs connected at 2 Gateways (4,000 at each GW). In all experiments, we started to measure the delays only after all MNs were sending their LUs. All results are the mean value of 5 measurements.

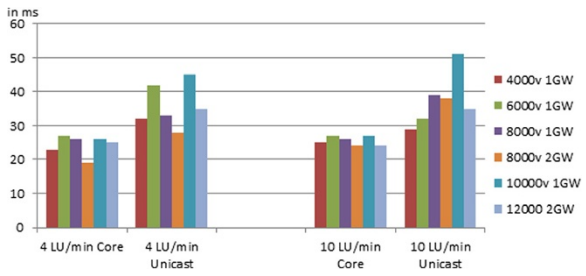

Figure 5 Core and unicast round trip delays.

As Figure 5 shows, the unicast and core round trip delays are very stable for all test parameters (20–45 ms), which suggests that our system is not yet overloaded. Unicast messages to any MN are delivered quite fast (up to 50 ms), and yet the SDDL core network is still far from saturation (< 20 ms), which means it could handle far more messages. As shown in Figure 6, the broadcast delays are much higher (up to 45 sec), which is expected because all MNs must be contacted individually and their response must be obtained until the total round trip is completed. As mentioned before, we could not send a broadcast message to more than 10,000 MNs connected to a single Gateway because this caused a drop of connections during the broadcast tests. This problem is due to the large bulk of messages being sent out - and the corresponding replies received - nearly simultaneously through a single UDP port. This results in an overload peak on the operating system's UDP buffer and causes several datagrams to be lost, many of them being MR-UDP's Acknowledge and connection control segments. Hence, MR-UDP drops the connections as if the node had lost its network connection.

4.2 WAN tests

To evaluate the performance of the SDDL middleware in a WAN environment with high-latency connections

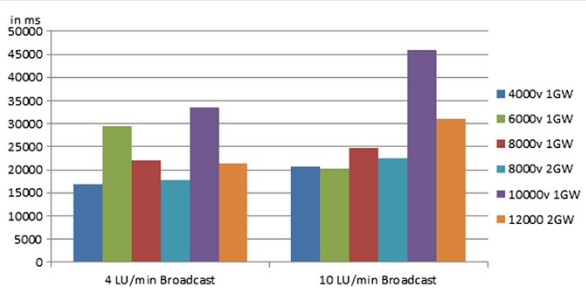

Figure 6 Broadcast round trip delays.

and subject to intermittent connectivity and/or the occurrence of IP address changes (such as those experienced by mobile nodes connected by mobile network providers), we performed the following experiments: we ran several performance tests involving three Gateways and 1 PoA-Manager in our lab and several thousand simulated mobile nodes/vehicles launched in parallel on 4 to 5 remote machines served by different broadband ISP internet connections. We measured the Round-Trip Delay of both unicast and groupcast messages to the MNs. Some experiments also included frequent handovers, both initiated by the mobiles and/or by the PoA-Manager, the latter aiming at load balancing among the Gateways. For all these tests, the LU frequency was every 30 seconds (2 LU/min).

4.2.1 Experimental setup

The experimental setup was as follows. In our laboratory, the three Gateways were executed on separate machines: a Dell PowerEdge server (3.0 GHz, 2× Dual Core), a PowerMac G5 (2.5GHz Quad-core with 8GB RAM), and a PC (CPU Core i5 with 8GB RAM); the PoA-Manager and a GroupDefiner executed on a separate PC. All these machines were connected to a 10/100/1000-Mbps switch. This switch, in turn, was connected to a 10/100-Mbps switch at the router serving the Internet connection of our laboratory. At the remote side, the machines were diverse, but all were connected via wired Ethernet to the ISP modem or router. Before executing the experiment, all home testers measured their effective uplink capacity (which was always within a range of 0.25 to 0.9 Mbps) and downlink capacity (within a range of 1.01 to 9.56 Mbps). We chose not to use a Wi-Fi wireless network, as this would create a less realistic simulation scenario because all simulated MNs would be competing with each other for a single wireless IEEE 802.11 connection, which is not collision free, as opposed to what happens with real-world Edge or 3G connections. However, we emulated the intermittent connectivity of real-world wireless connections by making the simulated MNs randomly change their MR-UDP connections and reconnect to a new Gateway. Additionally, there was very little interference from Internet connection usage by other applications on the remote machines.

4.2.2 Testing unicast and broadcast without handovers

In these experiments, the MN-Simulator program (here denoted by *MS-i*, the MN_simulator program launched at remote/home machine i) initially connected all the simulated MNs to a single Gateway, but immediately after each of them established the MR-UDP connection with this Gateway, it received (only once) a PoA-List of size 3, containing the IP addresses and ports of all three gateways running in our laboratory (which was used for the handover tests - see section 4.2.4). In this experiment, we turned off the load-balancing function of the PoA-Manager because we wanted to exclusively evaluate the SDDL's performance with mobile-initiated, spontaneous handover, i.e., without any interference/overload caused by mandatory handover requests by the PoA-Manager.

Table 1 shows the round trip delays (RTDs) of the unicast messages for three total amounts of simulated MNs executed at the remote machines. Because the Internet connectivity and the remote machine's capacities were so different, we measured the mean RTD time for each vehicle simulation program separately.

The lower value for the unicast RTDs for 7174 simulated MNSs was most likely caused by a sudden performance boost in the throughput of the ISP up/downlinks at one or more of the remote Internet connections. It also indicates that the increased number of clients does not yet affect the SDDL communication performance. The total number of MNSs is not a multiple of 4 because during the parallel launch and connection of >1000 MNSs to a Gateway some of the MR-UDP connections failed to be established, and our vehicle simulation program was not conceived to retry all failed connections several times.

In this same test run, we also measured the RTDs of a broadcast to, and reply from, all 1000 MNs executed on the 4 home machines, which took only 47.1 seconds. Because the broadcast incurs too great an instantaneous communication load in the MN-Simulator program and their Internet connections, we were only able to execute it for 250 MNs per machine.

4.2.3 Tests with mobile-initiated handovers

To test the performance of SDDL with spontaneous mobile-initiated handovers and with intermittent connectivity of the mobile nodes, we added - just for this experiment - a new message to the system, namely, the Handover Test (HT) message. We also modified the PoA-Manager and the MN-Simulator program accordingly to also perform the following.

Every 3 minutes, each MN decides if it will disconnect from the current Gateway and reconnect to another Gateway, chosen randomly from its PoA-list. This decision is controlled by a handover probability (HO_P), which we varied from 0% to 15%. Whenever an MN starts a handover, it first closes the current MR-UDP connection

Table 1 Round Trip Delays of Unicast to MNs of each home machine (in ms)

Total nr of MNS	MS-1	MS-2	MS-3	MS-4	MS-5	Global mean
1000	108	67.80	70	67.2	N/A	78.25
4123	115.8	86.20	84.4	87	N/A	93.35
7174	98.8	68.60	77.4	67.4	N/A	78.05

and then requests a new MR-UDP connection to the newly chosen Gateway, i.e., for some short period of time (a few ms), the simulated MN is entirely disconnected from any Gateway. Each handover is printed at the terminal console. Each MN also accepts the HT message and increments a global counter, which is also printed at the console.

The purpose of the HT message is to test the reliability of message delivery to the MNs during a handover/disconnection. The message is sent by the PoA-Manager immediately after it receives a "connection closed" message from the corresponding Gateway. Because the mobile node is disconnected, the non-delivered messages are received by the MTD service and later forwarded to the new Gateway where the MN reconnected. Thus, we wanted to check, at each MN-Simulator program, whether the total number of received HT messages equals the total number of performed handovers by the MNs, i.e., whether the MTD service had replayed all the non-delivered unicast messages or if some unicast message had been lost during the handover.

Table 2 shows the mean values of round trip delays (RTDs) of unicast messages for four combinations of total numbers of MNs and handover probability (HO_P), again, presented separately for each home machine.

From this data, we can make two observations: (i) a higher handover probability does not necessarily increase the overall RTD of unicast messages, showing that the retransmissions by the MTD and the disconnection management by the Gateways apparently only affect the message delivery times of the migrating mobile nodes; (ii) for the same handover probability, e.g., 5%, a larger number of total mobile nodes does slightly impact the increase of the overall message RTD.

When comparing the data of Tables 1 and 2 (for approximately 4000 MNs), it is interesting to note that the unicast RTDs are similar and even decreased slightly in the experiments with low-probability mobile-initiated handovers. Again, however, this result could be due to a lucky choice of the "pinged" MNs or a sudden enhancement of the link quality of the remote Internet connections.

There is a natural delay in the delivery of HT messages because the MTD service only resends non-delivered messages to the mobile nodes after the connection establishment is announced by the new Gateway. Because we did not implement the MN-Simulator to stop performing handovers after some time, at the end of the simulation there was always a gap between the last announced handover and the corresponding delivery of the HT message. This gap obviously increases with the number of MNs and their probability of performing handovers. Table 3 shows the percentage of "missing" HT messages at the end of the simulation for the tests with 1800 and 3979 MNs. However, when examining the output logs of the MN-Simulator, almost all the HT messages (of past handovers) appear to have been delivered. This result raises our confidence that SDDL supports reliable delivery of messages in the presence of handovers between Gateways.

4.2.4 Tests with groupcast messages

The purpose of the groupcast message test was to measure the RTD of groupcast messages (including the corresponding acknowledgements by all group members), for different sizes of groups, where the group members were simulated by MN-Simulator (MS-i) programs executed on the remote machines served by the different ISPs. Because we performed this experiment on a different day and from other remote machines, we named these programs MS-6 to MS-11 to clarify that the RTD times of this and previous experiments cannot be compared. In this experiment, the common ping delay was approximately 25 ms (except for MS-11, which was 444 ms). The down- and uplinks varied between 1.59 and 1.2 Mbps and 0.93 and 0.33 Mbps, respectively. It should be noted that MS-11 was a machine connected in Europe, and therefore, its RTD is much higher than those of the other vehicles executed on the Brazilian machines. For this experiment, we turned off the induced mobile-initiated handover behaviour of the simulated MNs (HO_P=0), i.e., they would only switch to another Gateway if their MR-UDP connection in fact failed.

The group size is approximate, as it was determined by the GroupDefiner using a mod operation (e.g., ×%100) over the least significant byte of the MN-identifier, which is a randomly generated UUID. Thus, in the Gr-10%, the group had approximately 10% of 5795 MNs, and so on. Recall that in all test runs, the SDDL core nodes were also busy processing the LU messages sent every 30 seconds by each MN.

Table 4 shows the mean RTD times of 5 measurements for both the unicast and groupcast communication modes. It also indicates which of the remote machines simulating the mobile nodes actually participated (Yes) in the groupcast experiment. The numbers reveal that the

Table 2 Round trip delays of unicast messages (to each home machine) under different handover probabilities (in MS)

Total nr ot MNs/HO_P	MS-1	MS-2	MS-3	MS-4	MS-5	Global mean
1800/15%	103.6	72	65	61.2	70.2	74.4
3979/15%	93.2	68.2	84	63	73.4	76.36
5812/5%	112.6	79.2	102	70	92.2	91.2
7815/10%	79	58.8	59.6	50.4	334.8	116.52

Table 3 Percentage of "missing" HT messages after stopping the MN-Simulator programs

Total # MNs/HO_P	MS-1	MS-2	MS-3	MS-4	MS-5
1800/15%	2.4	1.7	4.9	3.1	1.5
3979/5%	4.9	5.9	2.5	3.0	6.2

mean RTD time for the estimated 579 and 1358 group members is only 19.7 and 66.4 seconds, respectively. This result suggests that a one-way groupcast message is most likely delivered to all the group members 40-70% of this time. Moreover, although we do not know how many group members were actually executed by MS-11, its longer ping delay certainly contributed to the total increase of the RTD in the Gr-10% experiment. As mentioned in section 4.2.3, we also tested and measured the RTD of a broadcast to 1000 MNs, and the obtained results for 1000 and 1358 deliveries and replies seem to be consistent.

5 Related work

SALES [24] is a middleware for data distribution aimed at large-scale mobile systems. It was designed based on two central concepts: QoC (*Quality of Context*) and CDDLA (*Context Data Distribution Level Agreement*). In a nutshell, QoC is a *Quality of Service* related to context information distribution services, while CDDLA is a quality contract that is established between any data producer and consumer and that is enforced by the middleware. SALES defines a tree-based hierarchical architecture of nodes to balance communication cost, performance and load balancing among the four types of nodes: the *Central Node* (at the root of the tree); the *Base Node*, a stationary node responsible for a network domain; the *Coordinator User Node*, which is responsible for discovering and connecting to (in an ad hoc manner) the *Simple User* nodes. Unlike our work, SALES relies solely on pure UDP for inter-node communication and, hence, does not take advantage of all real-time and QoS support of DDS.

Solar [25] is a middleware for ubiquitous computing that was designed to be scalable in the set of communicating nodes and is based on a self-organising P2P (*Peer-to-Peer*) overlay network. Solar employs a specific programming model called *filter-and-pipe*, where each component (filter) has a set of entry and exit ports and there may be data producers (sources) and consumers (sinks). In the Solar architecture, each node is considered a planet (that may have a number of "satellite nodes"), and the more nodes are used, the more scalable the system is. This *middleware* uses two transport protocols, DHT Pastry (*Distributed Hash Table*), for Discovery and routing, and TCP, for "inter-planetary communication". Unlike SDDL, which uses the DDS-based core (and the Gateways), to ensure real-time and reliable delivery of data to and from the mobiles, Solar is based on DHT and TCP, which are not suited to mobile networks (TCP) and to low-latency message routing and delivery (DHT).

Apparently, there has been little research and development on DDS-based middleware systems for mobile distributed applications in arbitrary wireless networks. Most DDS studies present comparisons between and benchmarks of different DDS vendors' implementations, such as [26-28], but none of them mention wireless networks or mobile DDS deployments. Among the few works that focus on mobile devices, we found the DDS-based middleware proposed in [4], named DDSS. DDSS includes a specific architectural element that supports mobile nodes and ensures reliable data delivery even for mobile subscribers that switch their wireless access points during system operation, similar to the handovers supported in SDDL. In the proposed architecture, all mobile devices are required to execute a lightweight version of DDS, the Mobile DDS Client, whereas stationary nodes on the fixed communication network run full-fledged DDS nodes and are responsible for the routing and delivery of data to all nodes. Due to DDS connectivity and Firewall/NAT traversal restrictions (unless a VPN is created), all these Mobile DDS Clients must run in a single network domain and rely on stable wireless connectivity. Moreover, the authors present no data about the communication performance over wireless networks, and there is apparently no support for context-defined groups and groupcast communication.

Another DDS-based system targeted at mobile networks is presented in [29]. REVENGE is a DDS-compliant infrastructure for news dispatching among mobile nodes and that is capable of transparently and autonomously balancing the data distribution load in the DDS network. REVENGE implements a P2P routing substrate - deployed on a LAN - that is fault tolerant and self-organising. More specifically, it is able to detect crashed nodes and to reorganise the routing paths from any source node to any mobile sink node. Because all nodes run DDS (mobile nodes have the DDS minimum profile), it has full DDS QoS Policy support. REVENGE has been tested in a

Table 4 Round trip delays of unicast and groupcast messages (in ms)

Vehicles/Mode	Group size	MS-6	MS-7	MS-8	MS-9	MS-10	MS-11	Gr-cast RTD
5794/Unicast	0	100	59.6	58.4	59.2	50	289.4	
5795/Gr-10%	579	Yes	Yes	Yes	Yes	Yes	Yes	19720.60
5430.Gr-25%	1358	Yes	Yes	Yes	Yes	Yes	No	66437.80

wireless network (on a University Campus wireless LAN), but the authors have not shown performance data in situations where the mobile nodes had intermittent wireless connections and suffered IP address changes. For asynchronous communication capabilities at the mobile nodes, this system provides full DDS-based Pub/Sub support, while SDDL implements only a restricted form of group subscription, but which has the advantage of high performance and scalability. Moreover, REVENGE's asynchronous communication depends on mobile nodes' initiative to become a group publisher/subscriber. SDDL asynchronous communication instead supports, in a uniform way, MN-initiated group participation, external MN-agnostic grouping determined by the GroupDefiners, and context-defined groups.

It seems that the main distinguishing feature of SDDL, when compared with the above systems, is that its mobile nodes only need to execute the lightweight MR-UDP protocol, which is platform independent (because it requires only the TCP/IP-protocol stack) and is very resource-efficient. Moreover, because DDS does not perform well with intermittent connectivity and does not natively support Firewall/NAT traversal, the mobile clients of REVENGE and DDSS have to be executed in a single network domain and in wireless networks with strong connectivity guarantees. Table 5 summarises the main differences among the middleware systems.

6 Discussion

The SDDL architecture takes advantage of DDS' powerful data-centric approach, its "broker-free" Peer-to-Peer architecture, rich QoS support, and the highly optimised and scalable RTPS wire protocol to boost its performance. In particular, it is possible to add new DDS nodes without much degrading the overall system's communication performance. SDDL's design was also driven by the desire to be simple, efficient, extensible and generic. This guiding principle can be identified by the following characteristics:

Each type of node has a very specific and simple function, and the overall processing is achieved by the interaction among these simple building blocks. For example, while Gateways are concerned with the reliable communication with MNs, the PoA-Manager handles the mobile-to-Gateway assignment and Gateway load balancing, while GroupDefiners are responsible for tagging vehicles with group information.

SDDL's extensibility is also inherited from DDS, which makes it quite simple to add new nodes to the SDDL core network for inclusion of new processing services.

Table 5 Comparison of middleware systems for mobile communication

Aspect	Sales	Solar	DDSS	Revenge	SDDL
Application	Generic middleware	Generic middleware	Generic middleware	News dissemination	Generic middleware
Communication modes	Pure UDP	Pasty and TCP	Pub/Sub	Pub/Sub	Unicast, Groupcast and Broadcast, Limited form of Pub/Sub on MNs
Fault-tolerance	No	No	No	Active Replication on fixed nodes, and node failure detection allowing data re-routing	Gateway failure through MH handovers, and MR-UDP resilience to node's short disconnections and IP Addr changes
Reliable data delivery to mobile nodes	There is a contact between data producers and consumers	TCP reliability only (not well suited for wireless connection)	Yes	Yes, but no handover support	Yes, MTD service caches non-deliverd messages, and RUDP has internal asks
Software on the mobile node	Just UDP	TCP stack and Pastry protocol	Lightweight DDS node	DDS node with minimum profile	Just the MR-UDP java Library
DDS compliance and QoS support	No	No	Yes, also at the mobile nodes	Yes, also at the mobile nodes	Only in the SDDL core but not on the MNs
Load Balancing	Yes, using a hierarchical (tree)	N/A	N/A	Yes, in the routing substrate	Yes, of the mobile Gateways'load
Wireless deployment/test	Yes, without wireless disconnection tests	Yes, without wireless disconnection tests	Not mentioned	Deployment in campus Wi-Fi network	In a WAN, but simulated disconnection and IP Address changes
Number of MNs	N/A	N/A	N/A	10 source nodes, 10 sink nodes	Several thousands MNs
Context Updates by each MN	N/A	N/A	N/A	N/A	Yes, ≈1KB sent every 30 seconds
Total traffic	N/A	N/A	N/A	1000 news/s	>250 1KB-object/s

For example, it is possible to add a logging service that captures all communications on the network and saves them to a database. This service would be completely independent of the rest of the nodes.

Only essential communication support is expected to be running on the mobile nodes. Because a mobile node can be as simple as an embedded processor with limited resources, it is preferable to use an IP-based solution rather than expect more sophisticated communication protocols or middleware to be available on the device. Hence, we built a highly optimised UDP-based communication protocol (MR-UDP) with a small footprint and tangible benefits in regard to communication reliability and Firewall/NAT traversal, which we think is the best way towards a general-purpose connectivity solution. Making this protocol resilient to IP changes and temporary disconnections is a valuable adaptability capability in a mobile applications environment, where network connections are commonly not fully reliable.

No SDDL core node is required to maintain any state about any MN, whether it is its IP address, or its association with groups. This not only simplifies the handovers between Gateways but also facilitates the definition of new sorts of groups that are entirely customisable to the application.

Efficiency and scalability are also supported by the fact that mobile communication issues are decoupled from the processing of their data. Thus far, we have limited ourselves to rather simple context data classification and group definitions, but in the future we plan to experiment with more complex processing of the application messages and context updates. Moreover, the number of Gateways can easily be raised, to handle an increase in the number of nodes or in the generated traffic by the application.

7 Conclusion

In this paper, we presented an inherently distributed communication middleware named the Scalable Data Distribution Layer, which aims to support large numbers of data connections with mobile devices that send location updates many times a minute. Because at its core SDDL uses DDS, it directly inherits several of the OMG standard's benefits, such as data-centric data modelling, real-time and asynchronous event-based communication through the RealTime-PublishSubscribe (RTPS) protocol, powerful subscription filtering and data routing, and a rich set of QoS policies. Thus, the main contributions of SDDL are the following: (i) its optimised extension of real-time communication capabilities with mobile nodes without native DDS support through the use of the highly optimised and IP-address-independent MR-UDP protocol; and (ii) an adaptive and extensible communication middleware supporting mobile node handovers and broadcast and groupcast communication modes, where the group-defining logic is arbitrary. SDDL is free and can be downloaded from URL www.lac-rio.com/sddl/.

In addition to the Fleet Tracking and Management application described in section 3, more recently we have also used SDDL to develop a second mobile application aimed at supporting vehicle inspection by traffic police. In both cases, tests with several thousands of simulated mobile nodes (sending location updates every 6 seconds) have shown satisfactory performance results, where a group/broadcast communication to more than 1000 nodes happens in less than 1 minute. Of course, it is too early to tell how well this middleware will function when deployed in a real-world setting, which we plan to do soon. As future steps, we intend to work along several lines, such as the following: extend the CNClib with asynchronous communication capabilities (i.e., a content-based Pub/Sub), implement mechanisms and policies for enabling dynamic load balancing among the nodes in the SDDL core network, integrate a secure communications layer, and incorporate autonomic capability into the middleware system, allowing it to become completely adaptive and to support deployment of the middleware as a scalable connectivity service in the cloud.

Endnotes

[a]http://www.lac-rio.com/mr-udp.

[b]CoreDX DDS is a trademark of TwinOaks Computing Inc.

[c]RTI Connext is a trademark of Real Time Innovations (RTI).

[d]Because our current MR-UDP implementation carries 256 Bytes on each UDP packet, each LU is split into at least three UDP packets.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
All authors read and approved the final manuscript.

Received: 6 February 2013 Accepted: 1 July 2013
Published: 15 July 2013

References

1. Stojanovic D, Predic B, Antolovic I, et al. (2009) Web information system for transport telematics and fleet management. In: 9th International Conference on Telecommunication in Modern Satellite, Cable, and Broadcasting Services, (TELSIKS '09), pp 314–317
2. Rybicki J, Scheuermann B, Kiess W, Lochert C, Fallahi P, Mauve M (2007) Challenge: Peers onWheels – A Road to New Traffic Information Systems. In: Proceedings of the 13th annual ACM international conference on Mobile computing and networking - MobiCom '07
3. Sibley GT, Rahimi MH, Sukhatme GS (2002) Robomote: a tiny mobile robot platform for large-scale ad-hoc sensor networks. IEEE International Conference on Robotics and Automation (ICRA '02) 2:1143–1148
4. Herms A, Schulze M, Kaiser J, Nett E (2008) Exploiting publish/subscribe communication in wireless mesh networks for industrial scenarios. In: IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2008, pp 648–655
5. Grossmann M, Bauer M, Honle N, Kappeler U-P, Nicklas D, Schwarz T (2009) Efficiently Managing Context Information for Large-Scale Scenarios. In: Third

IEEE International Conference on Pervasive Computing and Communications (PERCOM '05), pp 331–340
6. Huang Y, Garcia-Molina H (2004) Publish/Subscribe in a Mobile Environment. Wireless Networks 10(6):643–652
7. Castro M, Druschel P, Kermarrec A-M, Rowstron AIT (2002) Scribe: a large-scale and decentralized application-level multicast infrastructure. IEEE J Selected Areas in Communications 20(8):1489–1499
8. Carzaniga A, Rosenblum DS, Wolf AL (2001) Design and evaluation of a wide-area event notification service. ACM Trans Comp Syst 19(3):332–383
9. Terpstra WW, Behnel S, Fiege L, Zeidler A, Buchmann AP (2003) A peer-to-peer approach to content-based publish/subscribe. In: Proceeding of the Second DEBS
10. Pietzuch PR, Bacon JM (2002) Hermes: A distributed event-based middleware architecture. In: Proceedings IEEE 22nd International Conference on IEEE Distributed Computing Systems Workshops, pp 611–618
11. Mahambre SP, Kumar M, Bellur U (2007) A taxonomy of QoS-aware, adaptive event-dissemination middleware. Internet Computing, IEEE 11(4):35–44
12. Corsaro A, Querzoni L, Scipioni S, Piergiovanni ST, Virgillito A (2006) Quality of service in publish/subscribe middleware. Global Data Manag 19:20
13. Esposito C, Cotroneo D, Gokhale A (2009) Reliable publish/subscribe middleware for time-sensitive internet-scale applications. In: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems. ACM
14. OMG (2012) Data Distribution Service for Real-time Systems Specifications. www.omg.org/spec/ *(visited on Sept. 28, 2012)*
15. Wang N, Schmidt DC, van't Hag H, Corsaro A (2008) Toward an adaptive data distribution service for dynamic large-scale network-centric operation and warfare (NCOW) systems. In: IEEE Military Communications Conference - MILCOM, pp 1–7
16. Pardo-Castellote G, Farabaugh B, Warren R (2005) An Introduction to DDS and Data-Centric Communications. In: Real-Time Innovations
17. Sanchez-Monedero J, Povedano-Molina J, Lopez-Vega JM, Lopez-Soler JM (2011) Bloom filter-based discovery protocol for DDS middleware. J Parallel Distributed Comp 71(10):1305–1317
18. Esposito C (2011) Data Distribution Service (DDS) Limitations for Data Dissemination wrt Large-scale Complex Critical Infrastructures (LCCI). Mobilab Technical Report 100. www.mobilab.unina.it
19. Xu B, Xu B, Linderman M, Madria S, Wolfson O (2010) A Tactical Information Management Middleware for Resource-constrained Mobile P2P Networks. In: Reliable Distributed Systems, 2010 29th IEEE Symposium on. IEEE
20. Baldoni R, Bonomi S, Lodi G, Platania M, Querzoni L (2011) Data dissemination supporting complex event pattern detection. Int J Next Gen Comp 24
21. Endler M, Baptista G, et al. (2011) ContextNet: Context Reasoning and Sharing Middleware for Large-scale Pervasive Collaboration and Social Networking. In: Poster Session, ACM/USENIX Middleware Conference. , Lisbon
22. Malcher M, Aquino J, Fonseca H, et al. (2010) A Middleware Supporting Adaptive and Location-aware Mobile Collaboration. In: Mobile Context Workshop: Capabilities, Challenges and Applications, Adjunct Proceedings of UbiComp 2010. Copenhagen
23. RTI (2011) RTI Data Distribution Service - Comprehensive Summary of QoS Policies. http://community.rti.com/rti-doc/45e/ndds.4.5e/doc/pdf/RTI_DDS_QoS_Reference_Guide.pdf (visited in September 2012)
24. Corradi A, Fanelli M, Foschini L (2010) Adaptive context data distribution with guaranteed quality for mobile environments. In: 2010 5th IEEE International Symposium on Wireless Pervasive Computing (ISWPC), p 8
25. Chen G, Li M, Kotz D (2008) Data-centric middleware for context-aware pervasive computing. Elsevier Pervasive Mobile Comp 4(2):216–253
26. Pongthawornkamol T, Nahrstedt K, Wang G (2007) The Analysis of Publish/Subscribe Systems over Mobile Wireless Ad Hoc Networks. In: 2007 Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services (MobiQuitous), pp 1–8
27. Esposito C, Russo S, Di Crescenzo D (2008) Performance assessment of OMG compliant data distribution middleware. In: 2008 IEEE International Symposium on Parallel and Distributed Processing, pp 1–8
28. Xiong M, Parsons J, Edmondson J (2010) Evaluating the Performance of Publish/Subscribe Platforms for Information Management in Distributed Real-time and Embedded Systems. http://portals.omg.org/dds/sites/default/files/Evaluating_Performance_Publish_Subscribe_Platforms.pdf
29. Corradi A, Foschini L, Nardelli L (2010) A DDS-compliant infrastructure for fault-tolerant and scalable data dissemination. In: Proceedings of the The IEEE symposium on Computers and Communications (ISCC '10). IEEE Computer Society, Washington, DC, USA, pp 489–495

doi:10.1186/1869-0238-4-16
Cite this article as: David *et al.*: **A DDS-based middleware for scalable tracking, communication and collaboration of mobile nodes.** *Journal of Internet Services and Applications* 2013 **4**:16.