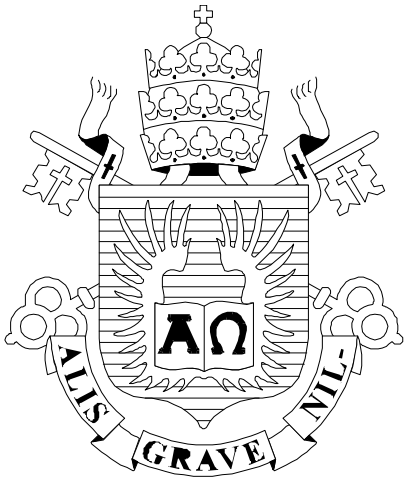

PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 06/12

A DDS-based middleware for scalable tracking and communication of wireless-connected mobile nodes in a WAN

Markus Endler
Rafael Oliveira Vasconcelos
Lincoln David Nery e Silva
Rafael André
Lucas Alves

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 06/12 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena June, 2012

A DDS-based middleware for scalable tracking and communication of wireless-connected mobile nodes in a WAN

Markus Endler Rafael Oliveira Vasconcelos

Lincoln David Nery e Silva Rafael André Lucas Alves

{endler, rvasconcelos, lnsilva, randre, lalves}@inf.puc-rio.br

Abstract. Applications such as vehicle fleet monitoring and logistic systems, emergency response coordination, environmental monitoring or mobile workforce management, employ mobile networks as means of communication, information sharing and coordination among a possibly very large set of mobile nodes interconnected by a Wide Area Network (WAN). The majority of those systems thus requires real-time tracking of the mobile nodes, interaction with all participant nodes, as well as means of adaptability in a very dynamic scenario, where it is not possible to predict when, where and for how long the nodes will remain connected. Several studies and real-world applications suggest that OMG's Data Distribution Service (DDS) standard - and corresponding middleware products - enable scalable decentralized solutions for real-time communication between large sets of networked nodes. However, our experiments show that DDS only works well when deployed on a LAN or a high-performance network, drastically loosing its performance when used in a wireless network. In this paper we present a DDS-based communication middleware that supports real-time tracking and communication with several thousands of mobile nodes, wirelessly connected over the WAN, as well as three modes of communication: unicast, groupcast and broadcast. We then show some performance results of our middleware, that demonstrate the viability of extending the real-time communication capacity of DDS also to wireless-connected mobile nodes in a WAN.

Keywords: DDS, mobile collaboration, mobile communication, middleware, scalable communication

Resumo. Aplicativos como o monitoramento de frota e sistemas de logística, coordenação de resposta a emergências, monitoramento ambiental ou gestão da força de trabalho móvel, empregam redes móveis como meio de comunicação, compartilhamento de informações e coordenação de um número possivelmente grande de nós móveis interligados por uma Wide Area Network (WAN). A maioria destes sistemas, portanto, requer monitoramento dos nós móveis em tempo real, interação com todos os nós participantes, e meios de adaptabilidade em um cenário muito dinâmico, onde não é possível prever quando, onde e por quanto tempo os nós permanecerão conectados. Vários estudos e aplicações reais sugerem que o padrão de Serviços de Distribuição de Dados da OMG (DDS) - e produtos correspondentes de middleware - habilitam soluções descentralizadas e escaláveis para comunicação em tempo real de grandes conjuntos de nós de rede. No entanto, nossos experimentos mostram que o DDS só funciona bem quando implantado em uma LAN ou uma rede de alto desempenho, perdendo drasticamente seu desempenho quando utilizado em uma rede sem fio. Neste artigo apresentamos um middleware de comunicação baseado em DDS que suporta monitoramento em tempo real e comunicação com vários milhares de nós móveis, conectados

sem fio por uma WAN, e três modos de comunicação: unicast, groupcast e broadcast. Depois, mostramos alguns resultados de desempenho do nosso middleware que demonstram a viabilidade de ampliar a capacidade de comunicação em tempo real do DDS para nós móveis conectados sem fio em uma WAN.

Palavras-chave: DDS, colaboração móvel, comunicação escalável, comunicação móvel, middleware

In charge for publications

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3114-1516 Fax: +55 21 3114-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Table of Contents

1 Introduction

As wireless connectivity through 2G/3G networks and Wi-Fi technology is becoming ubiquitous, and GPS-enabled portable devices with several embedded sensors are being massively sold (smart phones, tablets), new distributed mobile applications requiring instant communication and tracking of mobile nodes (e.g. humans, vehicles or autonomous robots/vehicles) are becoming feasible. In particular, application fields such as vehicle fleet tracking and management, logistics, remote monitoring of devices for maintenance, emergency response coordination, homeland security or mobile workforce management, are putting pressing demands for systems capable of (soft) real-time monitoring and communication among large sets of mobile nodes, connected through any wireless technology. For example, transport and energy companies operating trucks on a large territory are demanding systems for tracking and instant communication between their transport teams, enabling them to share information about delivery-tasks, road conditions, vehicle conditions, or any event that may have occurred during their routes.

As wireless connectivity through 2G/3G networks and Wi-Fi technology is becoming ubiquitous, and GPS-enabled portable devices with several embedded sensors are being massively sold (smart phones, tablets), new distributed mobile applications requiring instant communication and tracking of mobile nodes (e.g. humans, vehicles or autonomous robots/vehicles) are becoming feasible. In particular, application fields such as vehicle fleet tracking and management, logistics, remote monitoring of devices for maintenance, emergency response coordination, homeland security or mobile workforce management, are putting pressing demands for systems capable of (soft) real-time monitoring and communication among large sets of mobile nodes, connected through any wireless technology. For example, transport and energy companies operating trucks on a large territory are demanding systems for tracking and instant communication between their transport teams, enabling them to share information about delivery-tasks, road conditions, vehicle conditions, or any event that may have occurred during their routes.

A common characteristics of all the applications considered in our work is the fact that the mobile nodes periodically produce some data about them (e.g. probing sensors), which we call context information, as for example, their position, speed or other data, and publish this data so to be processed or visualized by other nodes – either stationary or mobile. Hence, we also assume that each mobile node has some wireless interface and is capable of communicating with other stationary machines through the IP protocol. In these applications, the main requirement is that, if the mobile node is connected and is producing its context data, this context update (CxtU) should be delivered to all the other interested nodes almost instantaneously. Moreover, messages should be deliverable to each mobile node that is connected, with the smallest possible delay.

In the past, much research has been done in Publish/Subscribe (Pub/Sub), but only few support large-scale mobile networks with wireless links and at the same time offering QoS communication guarantees. On the other hand, OMG's Data Distribution Service (DDS) standard [11] offers high performance communication capabilities, currently used for several distributed mission-critical applications, and which should be useful also for large-scale mobile networks. DDS specifies a Peer-to-Peer, scalable middleware architecture for data distribution in (soft) real time, with Quality of Service (QoS) contracts between producers and consumers of data (e.g. best effort or reliable communication, data persistency and several other message delivery optimizations, etc.). Unlike other communication middleware, DDS can explicitly control the latency and efficient

use of network resources through fine-tuning of its network services that are critical in soft real-time applications (e.g. its QoS policies deadline, latency budget or transport priority, etc.). However, so far we have not seen any deployment of DDS for large-scale and wireless-connected mobile distributed applications, such as the ones described above, which suggests that DDS cannot be efficiently deployed directly at mobile nodes in the wireless networks. This motivated us to design and implement a middleware that extends DDS' high-performance communication capabilities to wireless-connected mobile devices. As the main requirement of our middleware, we put scalability, simplicity and high communication performance.

The main contributions of this paper are the following:

1. We discuss the problems of deploying DDS on mobile networks, and give evidence of the incurred performance penalties when using DSS' Reliability and Durability QoS policies in settings with intermittent connectivity;

2. Describe our DDS-based communication middleware and show that it supports re-liable unicast and groupcast message delivery to mobile nodes in spite of IP address changes, temporary disconnections, and Firewall/NAT traversal;

3. Give evidence of the scalability of our middleware, even for a deployment in a WAN, and show that it is capable of handling frequent handovers of mobile nodes;

4. Show a mechanism by which the processing and communication workload can be balanced among the Gateway nodes;

5. Describe support for two sorts of groups of nodes: explicit and context-defined groups, and show how the latter are efficiently maintained/updated by our middleware.

6. Present results of several performance tests made in several WAN settings, showing the apparent suitability of our middleware for communications in large-scale mobile applications.

This work is part of a larger project called ContextNet, aimed at developing middleware for (soft) real-time communication, coordination and collaboration in large-scale distributed mobile applications. In the scope of this project, the middleware presented in this paper, is the basic layer for communication and context information sharing. This middleware, called Scalable Data Distribution Layer (SSDL), is available for download at http://www.lac.inf.puc-rio.br/sddl

Paper outline: In the next section we discuss why DDS apparently is not well suited for wireless communications, and justify our claim by showing data of a performance test done with commercial DDS products. In section 3 we overview the architecture and main components of SDDL, and in section 4, explain SDDL's support for group communication and management. Section 5 then shortly explains our middle-ware's main use case, and section 6 presents results of several performance tests done for different numbers of simulated mobile nodes accessing SDDL from a WAN, and with diverse handover behavior. In section 7 we compare our work with related work, and in section 8 we make some concluding remarks, respectively.

2 Why DDS Should not be Used for Mobile Networks

Due to several successful deployments, DDS has proved to be a good solution for a large number of real-world mission critical applications that have soft real-time communication requirements. However, most of these DDS-based applications seem to have a specific network set-up which ensures highly efficient communication and ability to support most of the required QoS parameters. However, this common net-work

setting for DDS is not the same as that of large-scale distributed mobile applications discussed in the previous section. Trying to use DDS for such applications raises several interesting challenges, which we will discuss in the following.

It does not take long to realize that deploying an application over the WAN with DDS is not a simple task. DDS' optimized data communication facilities are heavily based on IP Multicast messages, that are quite restricted in the Internet. In fact, as discussed in [7], Internet is characterized by many interconnected "scattered islands where IP Multicast is available".

But not only the lack of Internet-wide IP Multicast is the problem. Since DDS QoS policies are implemented by properly setting the DDS Network Services, there is no guarantee that most DDS optimizations and QoS policies will work in a WAN where the messages are routed through several networks which are beyond reach to the user of a DDS-based application.

In a wireless mobile setting we face additional challenges, since mobile nodes usually have scarce resources and small bandwidth network connections, therefore making them not well suited to act as regular DDS nodes. Moreover, wireless mobile DDS nodes can be expected to frequently experience network disconnections, especially in cases where the node is connected via 3G or EDGE technology. On the other hand, DDS offers two QoS policies that could be used to address this problem: (i) Reliability and (ii) Durability. The Reliability policy can be used to guarantee that every message will be delivered to all active/connected nodes, while the Durability policy persists all DSS messages for a pre-defined amount of time. So, with both policies DDS would be able to ensure that intermittent nodes and late joiners will receive all messages sent. However, there is very little information in literature on how those QoS policies behave under different frequencies and durations of intermittent connectivity. In order to answer these questions we decided to run an experiment: to deploy a simulated set of mobile data-producing DDS nodes on two servers, and test the system's data delivery performance using the aforementioned DDS QoS policies with intermittent connectivity. For our experiments we chose two well-established commercial DDS implementations available. In this experiment we would artificially cause short periods (< 1 second) of wireless disconnection for a random number of the simulated mobile DDS nodes and measured the Roundtrip delay (RTD) of unicasts with acknowledgement to all simulated nodes.

Initially, we wanted to run the experiment with machines in different network domains. But, as in [7], we faced a serious limitation of our DDS products: their inability/complexity of usage in a WAN. Each node would have to establish a VPN connection in order to participate in the DDS domain – which is a non-scalable solution. To proceed with our tests, we thus switched to a LAN network set-up and obtained results for a simultaneous disconnection of 10, 100 and 1000 simulated mobile DDS nodes, shown in Table 1.

Total nr of simulated DDS nodes	10	100	1.000
Average RTD (in milliseconds)	2900	2800	3050

Table 1. Test of DDS implementations with intermittent connectivity

The obtained result show that even for short-lived disconnections, and independent of the number of disconnected nodes DDS suffers from significant loss of communication performance. These results and the problems mentioned before suggest that DDS should not be used for high-scalable mobile applications deployed over the WAN, at least not until there's a major upgrade of the networking quality by ISPs or improvements of the available DDS implementations. As a means to address these problems, in

this paper we present SDDL, a middleware solution that uses DDS only on stationary nodes in a LAN (it could also be a cloud infrastructure) and employs a scalable gateway approach to bridge the gap between the high performance DDS and the weak wireless connectivity to the mobile nodes. SDDL has a particularly interesting feature: it's a lightweight implementation that can run on almost any mobile device with IP connectivity, and does not require a platform-specific DDS implementation on the mobile nodes.

In order to compare SDDL with one of the commercial DDS implementations, we did some performance tests for three different scenarios in a wired LAN setting. The tests measured the amount of time needed to distribute one hundred 1K messages among different sets of nodes. The results are shown below:

Total number of MN	100	500	1000
Our DDS Implementation	1020ms	223ms	114ms
SDDL	990ms	199ms	102ms

Table 1. SDDL vs. commercial DDS performance

The results indicate that our middleware is good enough as a DDS replacement for mobile nodes in the contexts that DDS does not perform well or does not perform at all. Through our middleware we expand some advantages of DDS's real-time communication capabilities to mobile nodes, and solve at least some of the problems that prevent DDS from being used in a mobile WAN setting. The most basic layer of the ContextNet architecture is its communication middleware, named Scalable Data Distribution Layer (SDDL), which connects stationary nodes of a wired "core" network with all the mobile nodes. This layer is the focus of this paper, and will be further described in more detail. The other ContextNet middleware layers and services have been presented in [2]. The collaborative applications are to be built within Social Networks or by using our Mobile Collaboration and Coordination Framework Mobilis [3].

3 Overview of the Scalable Data Distribution Layer (SDDL)

Scalable Data Distribution Layer (SDDL) is a communication middleware that connects stationary DDS nodes in wired "core" network to mobile nodes with an IP-based wireless data connection. Some of the stationary nodes are information and context data processing nodes, others are gateways for communication with the mobile nodes, and yet others are monitoring and control nodes operated by humans, and capable of displaying the mobile node's current position (or any other context information), managing groups, and sending unicast, broadcast, and groupcast message to the mobile nodes (MN).

SDDL employs two communication protocols: DDS's (Distribution Service for Real-Time Systems) RTPS [2] for the wired communication within the SDDL core, and the Reliable UDP protocol (RUDP) for the communication at the edges, between the core network and the mobile nodes. As part of the core network - based on DDS - three sorts of SDDL nodes have distinguished roles:

The Gateway (GW) defines a unique Point of Attachment (PoA), for connection with the mobile nodes (MN), which are an IP address and a port. The Gateway is responsible for managing a separate RUDP connection with each of these MN, for-warding any application-specific message or context information to the SDDL core network, and in the opposite direction, converting DDS messages to RUDP messages and delivering them to the corresponding MNs.

The PoA-Manager is responsible for two things: to periodically distribute PoA-List to the MNs, and to eventually request some MNs to 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, a 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-membership/s of all mobile nodes. To do so, they subscribe to the DDS topic where any message or context up-date is disseminated, e.g. the ones sent by mobiles and forwarded by the corresponding Gateway, and map each node to one or more groups, according to some application-specific group logic. This group membership information is then shared with all Gateways in the SDDL core network, using another specific DDS topic, to which all Gateways subscribe so that they can update their cached node's membership information. Whenever a new message is published to a group, each Gateway queries its group-to-MN mapping, to learn to which of the attached MNs it should send the message. The current groups of a node can be determined, for example, by its current position (e.g. if it is inside some region), by the node ID, or by any other attribute/field of its context information (e.g. a vehicle's remaining fuel).

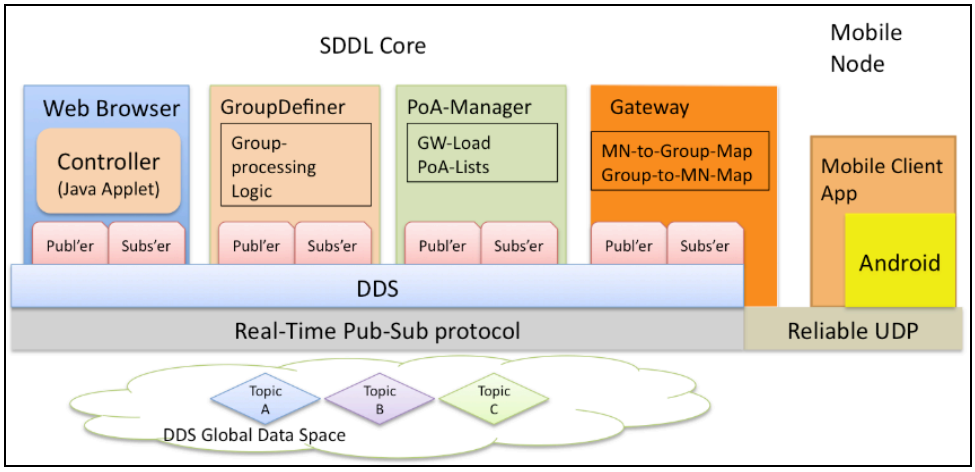

Fig. 1. Nodes and protocols used in SDDL

Figure 1 shows these three types of SDDL nodes, and the communication protocols they use. At the mobile side, currently we have implemented a simple Android-based client application, which displays the user's current location on a map and is capable of sending and receiving text messages. For testing purposes, we also have a program (a.k.a. the Vehicle simulator) which simulates any number of mobile nodes (each in a separate thread) periodically sending their position and also capable of performing a switching between Gateways. This program will be further explained in section 7.1. The Controller is a JavaScript and Applet, which runs in a web browser and is used to visualize the mobile nodes and their (random) trajectories on a map and to send unicast, groupcast and broadcast messages to the mobile nodes. Figure 2 shows a screenshot of the Controller, where the mobile nodes are represented by blue icons, leaving a blue trail behind, and the green icon is the mobile node to which a message is been sent. In addition to this main map-view, there is also another view (not depicted here) with options to select a group for groupcast communication.

3.1 RUDP

The Reliable UDP (RUDP) protocol is the basis for the Gateway-mobile node interaction. It implements some TCP functionality on the top of UDP, and has been customized to handle intermittent connectivity, Firewall/NAT traversal and robustness to switching of IP addresses and/or network interfaces. Each message, in either direction, requires an acknowledgement, and if one is not received each transmission is retried several times before considering the connection broken. In addition, RUDP has been optimized in the following ways: reduced number of connection-check packets; transparent continuation of a RUDP connection in spite of IP address changes; small number of connection maintenance packets for Firewall/NAT traversal. These optimizations are very important, since cellular wireless networks are not fully reliable everywhere. 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 wireless signal comes back, the device will probably get a new IP address. In our RUDP implementation, the previous connection to the mobile node will be maintained, and all buffered UDP messages will be delivered in the original order, provided that the disconnection time is shorter than a threshold.

3.2 Handling Mobile Node Handover

A mobile node Handover (HO) happens when a node connected to a Gateway drops or loses its connection and connects itself to a different Gateway. SDDL supports both core-requested, mandatory HO, i.e. when the PoA-Manager request the mobile node to re-connect to a new GW, and a mobile-initiated, spontaneous HO, i.e. when the MN spontaneously decides to re-connect to a new GW. In any case, it is the MN that actually chooses the new PoA from its PoA-List, and reconnects to the corresponding GW.

While performing a handover between Gateways, i.e. during the period of time when the MN is temporary disconnected, it could happen that some messages for the node are lost. In order to prevent this from occurring and achieving reliable delivery of messages during handovers (a.k.a. smooth handovers), SDDL includes the Mobile Temporary Disconnection Service (MTD).

The MTD Service, which may run on any node(s) of the SDDL core network, is responsible for buffering all messages that have not been delivered to the MN during its HO. But as soon the MN is connected to the new GW the MTD Service will resend all buffered messages into the DDS domain, so that they can be delivered to the MN through the new GW. Since not all applications require reliable message delivery across handovers (smooth handover), the MTD service is optional in SDDL.

The interested reader may refer to [3] for more information about the SDDL and about performance tests done in a Local Area Network.

3.3 Load Balancing Support

As aforementioned, SDDL has a load balancing support based on MN handovers. This approach allows the SDDL to balance the workload among Gateways by sending mandatory HOs to MNs. To enable load balancing, Gateways periodically publish through SDDL domain a LoadReportTopic that contains information about their CPU usage, free memory, number of vehicles connected and public IP address and port. Others information such as network usage, for instance, could be easily added to this topic.

By receiving the load reports sent from Gateways the PoA-Manager has knowledge about the Gateways' workloads, thus it is able to redistribute the MNs to reconnect through a different Gateway. As a first implementation, PoA-Manager applies a simple algorithm that distributes the Gateways' workload based only on number of vehicles connected. However, we argue that this algorithm, which is a module a part from the PoA-Manager itself, can be changed by another more sophisticated one. In fact, we are in an initial stage to develop a new algorithm that utilizes information about CPU and network usage, free memory and number of MNs connected.

PoA-Manager has a second module that is in charge of choosing which MNs should execute a HO to which new Gateways. The first module only estimates how many MNs should stay connected in each Gateway. This second module, for instance, could decide to keep on the same Gateway MNs that belong to the same geographical proximity. Both modules may be dynamically changed without impacting PoA-Manager execution.

4 Group Communication and Management

SDDL has been deployed in a real-world Fleet Tracking and Management application of a major gas distribution company, which operates throughout the entire country in Brazil. In SDDL, a mobile node n may be member of one or more groups. 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 a same company or administrative domain, or nodes of a 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, and this is done by the GroupDefiners in tandem with the Gateways: for each CxtU the GroupDefiners check if some membership changed, and if this is the case, disseminate this node's group change to all Gateways, which update their mappings accordingly.

Based on this group management, groupcast message delivery is then implemented, in two steps: first, the sender disseminates a single message in the DDS domain and tags it for the corresponding groups; secondly, each Gateway identifies which of its attached MNs are in the groups, and sends to them a copy for the message through RUDP. It should be mentioned that this groupcast message may not be delivered exactly to all members of a group, since the group membership of a node may not be reflected in the Gateway mapping until its group determining CxtU is produced by the node (e.g. every 30s), received and processed by the corresponding GroupDefiner. However, we believe that for most applications such groupcast delivery imprecision is perfectly tolerable. In fact, there is this tradeoff between groupcast precision and the generated traffic load caused by of more frequent CxtU.

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 member of. The separation of the generic and the specific group membership processing parts has some advantages: (i) it is possible to deploy several GroupDefiners in the SDDL core, each of which executing a Group selection module that examines a certain type of the CxtU object independently of the other modules, and (ii) Group selection modules may be easily exchanged in the GroupDefiner, without compromising the remaining function of the SDDL group management and communication capabilities.

4.1 Extending Asynchronous Pub/Sub Alike Communication to the Mobile Clients

Our middleware also enables asynchronous communication from the mobile nodes, but only in a restricted form. Instead of allowing them to register an arbitrary filtering/selection expression with the subscription (as it is possible in content-based Publish/Subscribe), for the sake of performance, it is group-based: i.e. a mobile node may subscribe to data published in any of the groups (either explicit or context-defined), registered by through the Controller.

Moreover, we expect the mobile client app also to store on the MN all its active subscriptions. When a mobile node subscribes to a certain group, its subscription is sent to the corresponding Gateway, which stores it locally and marks this mobile node as being interested in groupcast messages of the corresponding group. Thus, whenever this Gateway receives a message for this group from the DDS domain, it will also select the subscribed MN as a destination of the message and hand it the message through RUDP. But whenever a handover occurs between two Gateways, it is necessary for the MN to re-issue its subscriptions. Of course, this form of Pub/Sub is quite different from the asynchronous communication support of DDS, including the lack of QoS policies. Until now, however, this is the best compromise that we could find for a scalable Pub/Sub mobile communication. Further details about our implementation of group management and asynchronous communication in SDDL can be found in [12].

5 Use Case: Fleet Tracking and Management

In the Mobile InfoPAE project, SDDL is being integrated into a real-world fleet tracking and management application of a major gas distribution company, which operates in the entire country. Using this application, the company's Operations Center is able to track and analyze the mobility pattern of all its trucks in real-time, so as to optimize the fleet's journeys, detect obstructions or jams on roads, and monitor the vehicle's itinerary, detecting abnormal events or actions (e.g. too long elapsed time at a client stop, too slow or to fast driving, etc.). Moreover, it supports simple text messaging with drivers, allowing the Operations Center to send them instructions or alerts, individually or to groups of them. For the communication with the vehicles, the company uses all four Brazilian major cellular network operators, since in each region of the country has its main operators. Moreover, in each region, there are significant differences of 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, temporary data link disconnections (due to weak coverage, and caused by handover latency). Finally, in most cases their 2G/3G connections will be behind firewalls of the cell operators.

6 Performance Tests

In order to evaluate the performance of the SDDL middleware in an environment with high latency connections (such as in WANs) 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 did the following experiments: we ran several performance tests involving three Gateways and one PoA-Manager executing 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 mobile nodes. Some experiments also included frequent handovers, both initiated by the mobiles and/or by the PoA-Manager, the latter aiming load balancing among the Gateways. In a previous publication, [3] we showed the middleware's performance for unicast and broadcast message delivery when deployed in a local area network.

6.1 General Test Set-up

The experimental set-up was as follows: In our lab, the three Gateways executed on separate machines: a Dell PowerEdge server (3.0 GHz, 2x Dual Core), a PowerMac G5 (2.5GHz Quad-core with 8GB RAM), and a PC (CPU Core i5 with 8GB RAM), while 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 lab. 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 in the range from 0.25 to 0.9 Mbps) and downlink capacity (in the range from 1.01 Mbps to 9.56 Mbps). We chose not to use Wi-Fi wireless network, as this would create a less realistic simulation scenario, since all simulated remote vehicles would be competing with each other for a single wireless connection, using a wireless protocol (802.11), which is not collision free, contrary 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 vehicles randomly close RUDP connections and reconnect to a new Gateway. Also, there was very little interference at the usage of the Internet connection by other applications on the remote machines.

Our vehicle simulation program uses a thread pool of size 30 to indefinitely exe-cute an arbitrary number vehicles, where each vehicle is scheduled to send 20 simulated coordinates (latitude, longitude) in an serialized java object (a.k.a. its context update - CxtU) to one of the gateways every 30 seconds. Thus, the total size of the CxtU message is approximately 1 KB . In addition to sending CxtUs each 30 seconds, each vehicle also receives sporadic ping messages from the Control node executing in the SDDL core, and immediately replied with a pong message.

6.2 Testing Unicast and Broadcast Without Handovers

In these experiments, the vehicle simulation program (say, VS-i for program launched at remote_machine i) initially connected all the simulated vehicles to a single Gateway, but right after each of them established the RUDP connection with this Gate-way, it received (only once) a PoA-List of size 3, containing the IP addresses and ports of all the three gateways running in our lab (which was used for the handover tests - see section 6.3). In this experiment, we turned off the load-balancing function of the PoA-

Manager, since we wanted to evaluate exclusively 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 3 shows the round trip delays (RTD) of the unicast messages for three total amounts of simulated vehicles executing at the remote machines. Since the Internet connectivity and the remote machine's capacities were so different, we measured the mean RTD time for each vehicle simulation programs separately.

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

Table 2. Round Trip Delays of unicasts to vehicles of each home machine (in ms)

The lower value for unicast RTD for 7174 simulated vehicles was probably 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 vehicles is not a multiple of 4 because during the parallel launch and connection of >1000 simulated vehicles to a Gateway some of the RUDP 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 RTD of a broadcast to - and reply from- all 1000 simulated vehicles, executing on the four home machines, which was only 47,1 seconds. Since the broadcast incurs in too much instantaneous communication load at the vehicle simulation program and their Internet connections, we were only able to execute it for 250 vehicles per machine.

6.3 Tests With Mobile-initiated Handovers

In order 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, the Handover Test messages (HT), and modified the PoA-Manager and the vehicle simulation program, accordingly to do also the following:

- Every 3 minutes, each vehicle decides if it will disconnect from the current Gate-way 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 a vehicle starts an handover, it first closes the current RUDP connection, and then requests a new RUDP connection to the newly chosen Gate-way, i.e. for some short period of time - a few ms - the simulated vehicle is entirely disconnected from any Gateway. Each handover is printed at the terminal console.

- Each vehicle 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 reliable delivery of messages to the vehicles during a handover/disconnection. It is sent by the PoA-Manager immediately after it receives a "connection closed" message from the corresponding Gateway. Since

the mobile node is disconnected, the non-delivered messages are received by the MTD service, and later forwarded to the new Gateway where the node/vehicle connects. Thus, we wanted to check, at each vehicle simulation program, if the total number of received HT messages equals the total number of performed handovers by the simulated vehicles, i.e. if the MTD service had replayed all the non-delivered unicast messages, or if some unicast message had been loss during the handover.

Table 4 shows the mean values of round trip delays (RTD) of unicast messages for four combinations of total number of vehicles and handover probability (HO_P), again, presented separately for each home machine.

Total nr of vehicles/HO_P	VS-1	VS-2	VS-3	VS-4	VS-5	Global mean
1800/15%	103.6	72	65	61.2	70.2	74.4
3979/5%	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. Round trip delays of unicast messages (to each home machine) under different handover probabilities (in ms)

From this data, we can see two things: (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 a same handover probability, e.g. 5%, larger number of total mobile nodes does slightly impact on the increase of the overall message RTD.

When comparing the data of Tables 3 and 4 (for approximately 4000 simulated vehicles), it is interesting to notice that the unicast RTD are similar, and even decreased a little bit in the experiments with low-probability mobile-initiated handovers. But, again, this could be due to a lucky choice of the pinged vehicles, or a sudden enhancement of the link quality of the remote Internet connections.

Concerning the delivery of HT messages, there is a natural delay due to the fact that the MTD service only resends non-delivered messages to the mobile nodes, after the connection establishment is announced by the new Gateway. And since we did not implement the vehicle simulation program to stop doing handovers after some time, at the end of the simulation, there is always a gap between the last announced handover, and the corresponding delivery of the HT message, and this gap obviously increases with the number of vehicles, and their probability of doing handovers. Table 5 shows the percentage of "missing" HT messages at the end of the simulation for the tests with 1800 and 3979 simulated vehicles. However, when examining the output logs of the vehicle simulation program, apparently almost all the HT messages (of past handovers) were delivered. This raises our confidence that SDDL supports reliable delivery of messages in the presence of handovers between Gateways.

Total # nodes	VS-1	VS-2	VS-3	VS-4	VS-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

Table 4. Percentage of "missing" HT messages after stopping the vehicle simulation programs

6.4 Tests With Groupcast Messages

The purpose of this 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 vehicle simulation programs (VS-i) executing on the remote machines served by the different ISPs. Since we did this experiment on a different day and from other remote machines, we named them VS-6 to VS-11 to make clear that the RTD times of this and previous experiments cannot be compared. In this experiment, common ping delay was around 25 ms (except for VS-11, that was 444ms), and down- and up-links varied between 1.59 – 1.2 Mbps and 0.93 – 0.33 Mbps, respectively. It should be noted that VS-11 was a machine connected in Europe, and therefore its RTD is so much higher than the other vehicles executing on Brazilian machines. For this experiment we turned off the induced mobile-initiated handover behavior of the simulated vehicles (HO_P=0), i.e. they would only switch to another Gateway if their RUDP connection in fact failed.

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

Vehicles/Mode	Group size	VS-6	VS-7	VS-8	VS-9	VS-10	VS-11	Gr-cast RTD
5794/Unicast	0	100	59.6	58.4	59.2	50	289.4	
5795/Gr-10%	579							19720.60
5430/Gr-25%	1358							66437.80

Table 5. Round trip delays of unicast and groupcast messages (in ms)

Table 6 shows the mean RTD times of 5 measurements, both for the unicast and the groupcast communication modes. The color of the field indicates which remote machines actually executed vehicles that participated in the groupcast (red means: not used). The numbers reveal that the mean RTD time for the estimated 579 and 1358 group members is only 19.7 and 66.4 seconds. This suggests that a one-way groupcast message is probably delivered to all the group members is 40-70%-fraction of this time. Moreover, although we don't know how many group members were actually executed by VS-11, its longer ping delay certainly contributed to the total increase of the RTD in the Gr-10% experiment. As mentioned in section 6.2, we also tested and measured the RTD of a broadcast to 1000 simulated vehicles, and the obtained results for 1000 and 1358 deliveries and replies seem to be consistent. We had planned to test the groupcast performance also in conjunction with induced mobile-initiated handovers (e.g. P_HO > 0), which certainly would considerable increase the RTD of the groupcast, but due to limited human resource and time, we were not able to do the experiments for presentation in this paper.

7 Related Work

Apparently, so far there is only few research and development on DDS-based middleware systems for mobile distributed applications in arbitrary wireless networks – most of DDS studies present comparisons between and benchmarks of different DDS vendors´ implementations, such as [8] [9] [10], but none of them mentions wireless networks or mobile DDS deployments. Among the few works that focus on mobile de-

vices, we found the DDS-based middleware proposed in [4], named DDSS. It includes a specific architectural element that supports mobile nodes and ensures reliable data delivery even for mobile subscribers that switch their wireless access point 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 single network domain and rely on stable wireless connectivity. Moreover, the authors present no data about the communication performance over wireless networks, and apparently there is no support for context-defined groups and groupcast communication.

Another DDS-based system targeted at mobile networks is presented in [5]. REVENGE is a DDS-compliant infrastructure for news dispatching among mobile nodes and which is capable of transparently and autonomously balancing the data distribution load in the DDS network. It implements a P2P routing substrate - deployed on a LAN - that is fault tolerant and self-organizing. More specifically, it is able to detect crashed nodes, and to re-organize the routing paths from any source node to any mobile sink nodes. Since all nodes run DDS (mobile nodes have the DDS minimum profile), it has full DDS QoS Policy support. REVENGE has been tested in a wireless network (on an University Campus wireless LAN), but the authors have not shown performance data in situations where the mobile nodes had intermittent wire-less connections and suffered IP address changes. Concerning 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, REVENVE's asynchronous communication depends of mobile nodes´ initiative to become a group publisher/subscriber. SDDL asynchronous communication support, instead, supports, in a uniform way, both MN-initiated group participation and external, MN-agnostic grouping determined by the GroupDefiners, and also context-defined groups.

It seems that the main distinguishing feature of SDDL, when compared to the above systems is that the mobile nodes of SDDL only need to execute the lightweight RUDP protocol, which is platform independent (since it requires only the TCP/IP-protocol stack) and is very resource-efficient. Moreover, since 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 execute in a single network domain and in wireless networks with strong connectivity guarantees. Table 7 summarizes the main difference among the systems.

Aspect	**REVENGE [5]**	**DDSS [4]**	**SDDL**
Application	News dissemination	Generic middleware	Generic middleware
Communication modes	Pub/Sub	Pub/Sub	Unicast, Groupcast and Broadcast, Limited form of Pub/Sub on MNs
Fault-tolerance	Active Replication on fixed nodes, and node failure detection allowing data re-routing	No	Gateway failure through MH handovers, and RUDP resilience to node´s short disconnections and IP Addr changes
Reliable data delivery to mobile nodes	Yes, but no handover support	Yes	Yes, MTD service caches non-deliverd messages, and RUDP

			has internal aks
Software on the mobile node	DDS node with minimum profile	*Lightweight(?)* DDS node	Just the RUDP java Library
DDS compliance and QoS support	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, in the routing substrate	N/A	Yes, of the mobile Gateways' load
Wireless deployment/ test	Deployment in campus Wi-Fi network	Not mentioned	In a WAN, but simulated disconnection and IP Address changes
Communication Scale	10 source nodes, 10 sink nodes	N/A	7500+ CxtU producers and several sinks; and 1 gr-cast source (Controller) and 1500+ sinks
Context Updates by each mobile	N/A	N/A	Yes, ≈1KB sent every 30 seconds
Publishing frequency per source	10–100 news/s	N/A	3 times/min
Total traffic load	1000 news/s	N/A	>250 1KB-objects/s

Table 6. Comparison of DDS-based systems for mobile communication

8 Conclusion and Next Steps

In this paper we have presented a middleware for efficient communication and dissemination of context data sent from mobile nodes connected through arbitrary IP-based wireless links. Through the use of DDS in its core network, the implementation of Gateways handling all connections with the mobiles, a simple but efficient mechanism for handling handovers between Gateways, and its lightweight and robust RUDP its architecture appears to be scalable to large sets of mobile clients. We also think that the group management and groupcast capability, and the processing of context-defined groups are interesting and useful feature which, to the best of our knowledge, have not been tried out for DDS.

So far, our tests with several thousands of simulated vehicles have given us satisfactory performance results, where a group/broadcast communication to more than a 1000 nodes happens in less than 1 minute. But of course, it is early to say how well this middleware will behave when used in a real-world node tracking and communication application, which will be tested soon. As future steps, we intend to work on several fronts: complete and enhance the mobile client API, turn the middleware more generic (so far, the DDS data model defining the topics is very specific and has to be customized for every new application), implement new functionality at the Controller for creating/destroying mobile groups, persisting them in a groups directory that can be queried by subscribers, augment the resource utilization and load data that PoA-Manager gets from the Gateways, and experiment with smarter load balancing algorithms.

References

[1] Shruti P. Mahambre, Madhu Kumar S.D., and Umesh Bellur, "**A Taxonomy of QoS-Aware, Adaptive Event-Dissemination Middleware**", IEEE Internet Computing 11, July, 2007, pp. 35-44.

[2] U. Farooq, S. Majumdar, and E.W. Parsons, "**High Performance Publish/Subcribe Middleware for Mobile Wireless Networks**", Mobile Information Systems, vol. 3, No. 2, IOS Press, 2007.

[3] L. David, R, Vasconcelos, L. Alves, R. Andre, G. Baptista, M. Endler, "**A Communication Middleware Supporting Large scale Real-time Mobile Collaboration**", IEEE 21st International WETICE, Track on Adaptive and Reconfigurable Service-oriented and component-based Applications and Architectures (AROSA), Toulouse, June, 2012.

[4] K.-J. Kwon, C.-B. Park, and H. Choi, "**A proxy-based approach for mobility support in the DDS system**," 6th IEEE International Conference on Industrial Informatics, 2008. INDIN 2008, 2008.

[5] A. Corradi, L. Foschini, and L. Nardelli, "**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, 2010, pp. 489-495.

[6] A. Corradi, and C. Julien, "**The Context of Coordinating Groups in Dynamic Mobile Environments**", in Proceedings of the 13th International Conference on Coordination Models and Languages (Coordination), June, 2011, pp. 49-64.

[7] Christian Esposito, "**Data Distribution Service (DDS) Limitations for Data Dissemination w.r.t. Large-scale Complex Critical Infrastructures**", 2011.

[8] T. Pongthawornkamol, K. Nahrstedt, and G. Wang, "**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), 2007, pp. 1-8.

[9] C. Esposito, S. Russo, and D. Di Crescenzo, "**Performance assessment of OMG compliant data distribution middleware**," in 2008 IEEE International Symposium on Parallel and Distributed Processing, 2008, pp. 1-8.

[10] M. Xiong, J. Parsons, and J. Edmondson, "**Evaluating the Performance of Publish/Subscribe Platforms for Information Management in Distributed Real-time and Embedded Systems**," omgwiki.org/dds, 2010.

[11] OMG, "**Data Distribution Service for Real-time Systems.**", 2007.

[12] R.O. Vasconcelos, L.D. Silva, L. Alves, R. André, and M.Endler, "**Real-time Group Management and Communication for Large-scale Pervasive Applications**", Technical Report: Monografias em Ciência da Computação - MCC 05/2012, Dep. de Informática, PUC-Rio, ISSN 0103-9741, May, 2012.