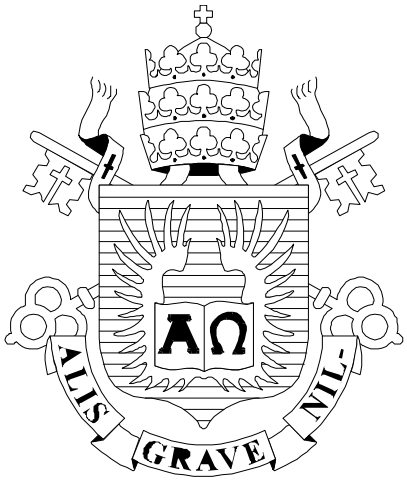

PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 05/12

Real-time Group Management and Communication for Large-scale Pervasive Applications

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

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900
RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 05/12 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena May, 2012

Real-time Group Management and Communication for Large-scale Pervasive Applications

Rafael Oliveira Vasconcelos Lincoln David Nery e Silva
Lucas Alves Rafael André Markus Endler
{rvasconcelos, lnsilva, lalves, randre, endler}@inf.puc-rio.br

Abstract. Applications such as transportation management and logistics, emergency response, environmental monitoring or mobile workforce management employ mobile networks as means of enabling communication and coordination among a possibly very large set of mobile nodes. The majority of those systems require real-time tracking of mobile nodes - be them vehicles, people or mobile robots-, group management and communication with the nodes, as well as adaptability in very dynamic scenarios, where it is not possible to predict how many nodes will be reachable and for how long. In this paper we present the Scalable Data Distribution Layer and explain its group management and communication features. We then present a prototype Fleet Tracking and Management system and performance results of the group communication and management middleware.

Keywords: group management, pervasive, large-scale, communication, middleware

Resumo. Aplicações como gerenciamento de transporte e logística, emergência, monitoramento ambiental ou gerenciamento de equipes móveis requerem redes móveis como meio de possibilitar comunicação e coordenação entre possivelmente um grande conjunto de nós. A maioria destes sistemas requer rastreamento em tempo real de nós móveis – sejam eles veículos, pessoas ou robôs móveis –, gerenciamento e comunicação de grupo com os nós, como também adaptabilidade em cenários bastante dinâmicos, onde não é possível prever quantos nós móveis serão alcançáveis e por quanto tempo. Neste artigo nós apresentamos a camada escalável de distribuição de dados e explicamos suas características de gerenciamento e comunicação de grupo. Nós depois apresentamos um sistema protótipo de rastreamento e gerência de frotas e resultados de desempenho do middleware de gerência e comunicação de grupos.

Palavras-chave: gerência de grupos, pervasivo, larga escala, comunicação, middleware

In charge of publications

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22451-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Table of Contents

1 Introduction

Applications such as vehicle fleet monitoring and dispatch systems, logistics, emergency response coordination, or mobile workforce management, employ mobile networks as means of enabling communication, data sharing and coordination among a possibly very large set of mobile nodes – that be vehicles, people or event mobile robots and UAVs. The majority of those applications thus requires the capacity of handling real-time dissemination of context/ location information, group communication and management for thousands of mobile nodes, adaptability in very dynamic scenarios, where mobile nodes experience intermittent connectivity, or change of their IP address, and dynamic grouping of nodes according to their current location or any other mobile context information.

As an example, consider the following hypothetic scenario: Door2Door (D2D) is a parcel delivery service that operates on the entire territory of Brazil, and has more than 5000 transport vehicles, each equipped with an on-board GPS and 2G/3G/Wi-Fi tablet. Through their Fleet Management system, D2D not only are able to track its vehicles in real-time, but also to send instant messages to drivers individually, to all drivers, or to subgroups of drivers, in order to give them instructions, ask them some questions, or send them specific alerts. In order to enable such tracking and communication capabilities the Fleet Management System relies on a communication middleware capable of explicitly defining groups of vehicles and of reliably sending messages to the groups. In addition, the parcel service also needs to group its vehicles according to their current locations (e.g. all the vehicles within a given metropolitan area, or within some county), according to the transported load (e.g. perishable goods), or the current operating conditions of the vehicles (e.g. vehicle tires with low pressure or overloaded), so as to be able to send specific instructions or alerts to the drivers of each of these vehicle subgroups. Unlike the explicit groups, these groups are dynamic, as vehicles may randomly enter or leave each group. Finally, D2D also uses vehicle-to-vehicle tracking and communication capabilities of the middleware, allowing each driver to receive updated information of all the other vehicles within a dynamic group, (e.g. all the vehicles in the same region), and thus enabling the drivers to help each other and to stand in when some problem with a nearby vehicle of the fleet is communicated.

The above scenario should have illustrated the benefits of such dynamic group management and communication capabilities for some Fleet Management Application. But we believe that several other communication and coordination applications for mobile nodes could also benefit from them.

In fact, in order develop such complex applications, it would be very helpful to use a communication middleware which: (a) handles the wireless connectivity problems with the mobile nodes, (b) is robust to short-lived disconnections and IP address changes of the nodes, e.g. when switching between mobile operators, or from one wireless technology to another, (c) supports the periodic context information/location update by large sets of nodes, (d) enables high-performance unicast, broadcast and groupcast communication to and among the mobile nodes; and (e) implements the management of dynamic, context-defined groups of mobile nodes.

To address all the above requirements, we have developed the *Scalable Data Delivery Layer* (SSDL), presented in [1]. SDDL uses a DDS [2] domain for communication among stationary nodes in its core, and employs a scalable gateway approach to bridge the gap between the high-performance communication within DDS and the potentially un-

reliable and disconnection-prone communication with thousands of mobile nodes. Unlike the previous paper [1], where we explained and showed performance data of only the unicast and broadcast communication to the mobile nodes, in this paper we will explain SDDLs group management and group communication capabilities.

We consider the main contributions of this paper to be the following:

- A scalable communication middleware which enables real-time processing of context information as well as unicasts, broadcasts and groupcasts to mobile nodes connected by a IP-based (wireless) connection;
- An approach to efficiently manage group membership updates of the mobile nodes, in a DDS domain, including management of context-defined groups, which may change by each context update message;
- Initial high performance results that show that even for 2,000 simulated nodes in a group, the groupcast delay is reasonable and that the group management overhead is negligible.

Paper Outline: In the next section we give an overview of the SDDL middleware and n section 3 we explain which kinds of mobile groups are supported, how they are dynamically updated and how groupcast messages are delivered in this middleware. In section 4 we explain the group subscription capabilities available also to the mobile nodes and how they are implemented. Then, in section 5 we present our prototype Fleet Tracking and Management System, and in section 6 we present results of performance tests our group management implementation. Section 7 discusses related work on scalable middleware for such mobile systems. Finally in section 8 we draw some conclusions and point to future work.

2 SDDL Overview

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 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 inbound and outbound communication between the core network and the mobile nodes. The DDS [2] is a standard from the OMG, which specifies a high performance, robust and scalable middleware architecture for real time data distribution, 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.).

As part of the core network based on DDS, two types of SDDL nodes have distinguished roles:

The Gateway (GW) defines a unique Point of Attachment (PoA), for connection with the mobile nodes (MN). The Gateway is thus responsible for managing a separate RUDP connection with each of these MN, forwarding any application-specific message or context information into the core network, and in the opposite direction, converting DDS messages to RUDP messages and delivering them reliably to the corresponding MN(s).

The PoA-Manager is responsible for two things: to periodically distribute PoA-List to the MNs, and to eventually request some of them to switch to a new Gateway/PoA. The PoA-List is always a subset of all available Gateways in the 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 or an existing Gateway is removed (failed) from the SDDL. Figure 1 shows the SDDL nodes and protocols.

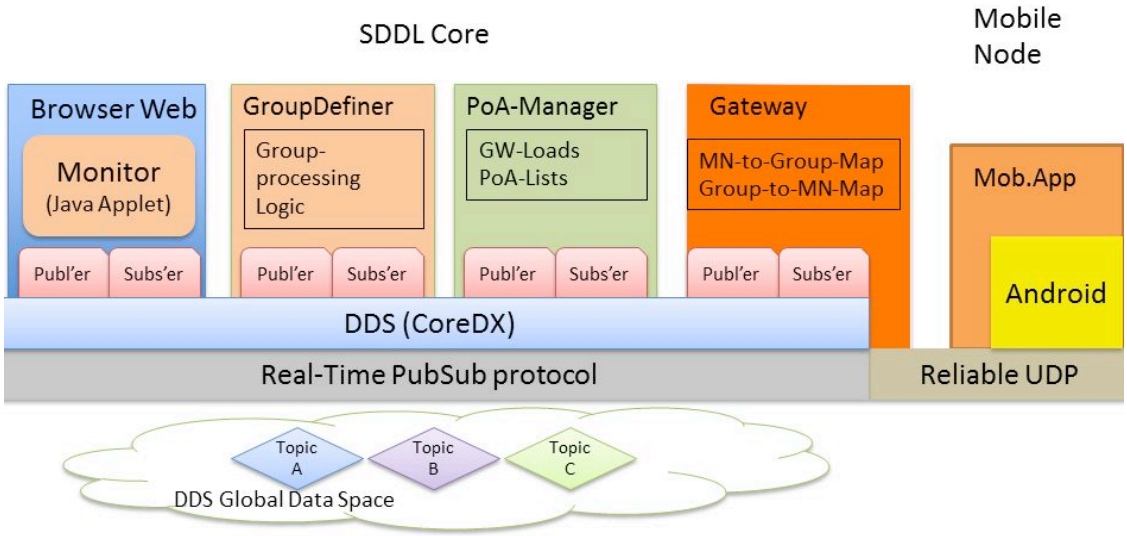

Figure 1. Nodes and protocols used in SDDL

2.1 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 this is not received, each transmission is retried several times, before the connection is considered 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.

2.2 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 has 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.

2.3 Node Ids and Message Delivery

In the SDDL, every mobile node and every Gateway have a unique identifier (ID). While RUDP 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 location update) entering the SDDL core network. By this, any corresponding node can learn which is the mobile's current GW, and most messages addressed to a mobile node will thus carry also a Gateway ID, allowing them to be directly routed to the corresponding Gateway. However, if the mobile node becomes suddenly unreachable/disconnected, its most recent Gateway, will notify this to all other nodes in the SDDL core network, by which the Gateway ID will be omitted in future messages to the mobile node. However, even in this case where the current Gateway of a mobile node is not specified, messages to the mobile node will be delivered, because they will be received by all Gateways.

As SDDL's basic functionality, it is possible to send two types of messages: unicast messages to a specific MN or SDDL core network node, and broadcast messages to all reachable MNs. Since many applications also require communication among subsets of mobile nodes, we incorporated into SDDL also a scalable group management and communication facility, described in the following.

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.

3 SDDL Group Management

Our concept of context-defined groups is similar to C. Julien's [3] context of a group, in which a mobile node (e.g. a vehicle on a highway) needs to track and communicate with other nodes that are nearby, for example to avoid a collision or sharing specific information related to its current visited region.

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 type of vehicle or company. In Context-defined groups, membership of a node is determined by its most recently produced context data. For example, if we consider the context "position", then all nodes located within a certain geographic region (e.g. a

metropolitan area or within the boundaries of a state), can form a context-defined group. But any other mobile-sensed & -produced context data may be used as well for group membership determination, such as: the node's local weather/temperature condition, its speed, residual energy (fuel or the user/driver's energy), or current mobile telephony company connected with. In any case, such context-defined groups have to be continuously updated according to the most recent context update sent by the nodes, and this will be done by specific nodes in the SDDL core, called *GroupDefiners*.

3.1 GroupDefiner

The GroupDefiners belong to the SDDL core and are responsible for defining the group-membership of all mobile nodes. To do so, they subscribe to the Context Update (CxtU) messages produced by any MN and published by the corresponding Gateway in the DDS domain. And based on some specific data field in this CxtU message, the GroupDefiners classify each MN into one or more groups. Then, using the DDS GroupAdvertisement topic, they disseminate any group membership change to all other elements in the SDDL core. The Gateways, on the other hand, subscribe to this topic so that they can update their group-to-node and node-to-group mapping of the corresponding node according to its new group membership.

The main task of a GroupDefiner is thus to continuously check and update the MN's group membership. For context-defined groups this is done by listening to the latest CxtU message by each MN being disseminated in the DDS domain and check if this context update causes some change in the node's group memberships. Thus, whenever a GroupDefiner detects some group membership change of a MN it will publish into the DDS domain a G*roup-membership-diff message* (shortly, G-diff) related to the MN, i.e. requesting to include MN into some groups, and to remove it from other groups. For example, consider a mobile node X moving along Highway 66. At one point in time it may be member of the position-related groups "Highway 66" and "California", but after moving on some miles and crossing the state border, it leaves group "California" and joins group "Nevada", while remaining in group "Highway 66". In this case, the G-diff message would conceptually consist of commands {add(X,"Nevada"), rem(X,"California")} All these G-diff messages will be used by the corresponding Gateway to update its Group-to-nodes mapping. For long-lived groups, the GroupDefiner will only publish a G-diff message if the user is explicitly removed from or added to a group.

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 mapping algorithm to determine the context-defined group/s that correspond to the received context data, e.g. if the CxtU data is within a certain range or region. This result is then handed back to the generic part of GroupDefiner, which will calculate the G-diff message and publish it to GroupAdvertisement topic in the DDS domain. Figure 2 illustrates the GroupDefiner's architecture and its interaction with the Gateways.

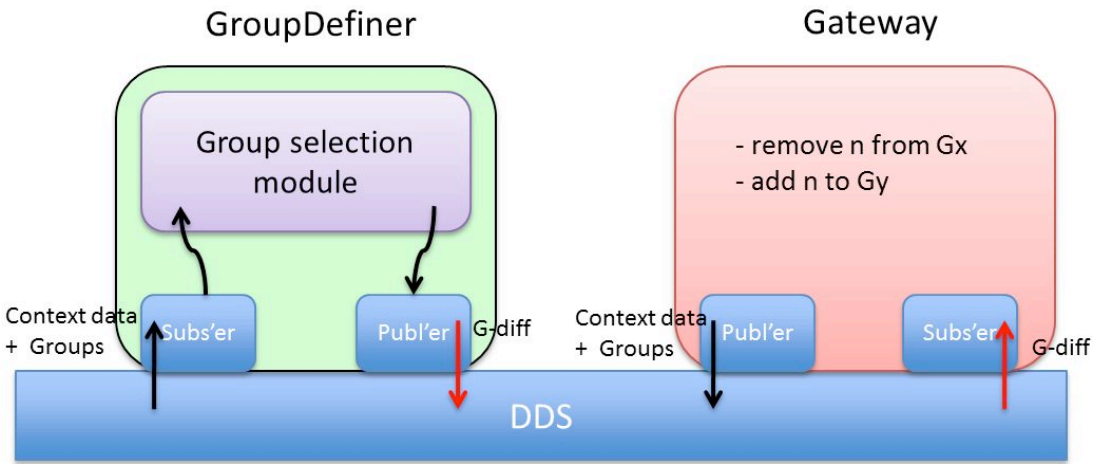

Figure 2. Interaction between GroupDefiners and Gateways

This split between 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.

3.2 Gateway

The Gateway has two mappings named "*MN-to-Groups*" and *Group-to-MNs* to support the Group Management and Communication. The first mapping, *MH-to-Groups*, is necessary for the *CxtU group tagging* by Gateways. The *CxtU group tagging* is the process of attaching to a CxtU messages all the groups that the specific MN belongs to. The *Group-to-MNs* mapping allows the Gateways to efficiently discover all the MNs that belong to a group, so as to forward group message efficiently to its MNs that are group members. Thus, the main task of Gateways in regard to Group Management and Communication is to keep its mappings up to date, allowing the Gateway to correctly forward all messages to/from MNs.

As aforementioned, the Gateways subscribe to the DDS GroupAdvertisement topic to receive and update the MNs´ group membership. After receiving a new G-diff message for MN *x* from a GroupDefiner, the Gateway updates its mappings by removing *x* from some groups, and adding it to other groups, according to the elements present in the G-diff message.

Another responsibility of the Gateway is to receive and manage MNs´ subscriptions, which could be application message subscriptions or context update subscriptions. The application message subscription informs the Gateway that the MN is interested in receiving all application messages sent to the specified group (that differs from the groups the MN is already member). When receiving such a subscription from a MN, the Gateway stores it, and from now on, all groupcast messages to the group informed in the subscription will be forwarded by the Gateway also to this MN. The message subscriptions are treated differently from the MN´s group membership. While MN´s group membership is determined by the Gateway's group tagging of CxtU messages, and by the G-diff messages disseminated by the GroupDefiners, the message subscriptions are stored locally in the Gateway to enable the correct forwarding of groupcast messages to MNs based on group membership and its subscriptions. In case of mobile node handover, the MN has always to re-submit its subscriptions to the new Gateway.

The MN is also capable of expressing interest of CxtU messages produced by others MNs. After receiving a context update subscription from one of its MNs, the Gateway forwards this CxtU-subscription to a *Global Subscription Manager (GSub)*, which is responsible for manage the CxtU-subscriptions and relaying the CxtU messages of a group to all the subscribed MNs, through the corresponding Gateways. Different from application message subscriptions, MNs do not need to re-submit their subscriptions after a handover.

Further explanations about asynchronous communication can be found in section 4.

3.3 Advantages Using Our Approach

In this subsection we point out the main advantages our group management approach, and explain how they are achieved: (i) enhanced and scalable performance, (ii) multiple/concurrent GroupDefiners, (iii) uniform treatment of explicit and context-defined groups, (iv) generic GroupDefiner architecture, and (v) stateless group management.

By disseminating a G-diff message that carries only the group membership changes of a MN, instead of sending the complete list of groups to which a MN belongs, we considerably reduce the size of data exchanged through the DDS in the GroupAdvertisement topic. Moreover, due to the G-diff structure, Gateways can more efficiently update their Group-to-MN mappings because they know exactly which operations (insertions or deletions) they have to apply in order to update their mappings.

The Gateway´s Group-to-MN mappings also support a scalable performance of group communication, as it only requires a single message to be disseminated in the SDDL core, while only the Gateways will replicate the message to all connected MNs that are group members and/or subscribers.

By implementing the GroupID through a pair (TypeID, InstanceID) we support multiple GroupDefiners, without the risk of a GroupID name conflict. Another advantage is the ability to have groupcast for specific TypeIDs, not only for the pair TypeID and InstanceID. When two GroupDefiners with different group selection modules are executing concurrently in the SDDL core, these group selections should have knew about the InstanceIDs of each module in order to avoid an ID conflict. Using our GroupID approach, each different group selection module has its own TypeID, which excludes the possibility of ID conflict in a scenario where group selection modules do not have known about others group selections modules.

Any group is managed in the same way - through the interplay of GroupDefiners and Gateways - regardless of its kind (explicit or context-defined group), or its membership- determining type of context.

The main advantage of the GroupDefiner's software architecture is that it completely hides all details related to DDS topic subscriptions and publications are hidden from the application developer, who can concentrate on programming the application logic of the Group selection module. Moreover, the general nature of the GroupDefiner's generic part allows it to be reused for different group definition logics. Moreover, it is even possible to swap implementations of Group Selection modules, (more accurate vs. faster group selection) according to the current execution requirements of the application.

Finally, apart from the group subscriptions by mobile nodes, which are either stored at the Gateways (as long as the MN is connected to it), or stored at the *Global Subscription Manager*, all other group management functions are stateless, which is very important to guarantee that the SDDL remains scalable, i.e. that new processing nodes and

Gateways can be added to the SDDL core, whenever this is required by increase of the number of mobile nodes or the volume of processed data.

4 Asynchronous Communication at the Mobiles

In order to satisfy the scalability requirement, SDDL extends to the mobile nodes only a restricted form of asynchronous communication, based on groups: i.e. mobile nodes may subscribe to data published in existing groups only (either explicit or context-defined), rather than be able to register an arbitrary filtering/selection expression with the subscription, as it is possible in content-based Publish/Subscribe systems. Moreover, delivery of application group-cast messages and Context Update messages are handled differently.

4.1 Application Message Subscription

When a mobile node subscribes to application messages of a certain group, a subscription is sent to the corresponding Gateway, which stores this subscription information locally and marks this mobile node as being interested in group-cast messages of the specific group. Thus, every DDS message tagged with the corresponding GroupID that this Gateway receives is forwarded to the subscribed mobile node. However, the Gateway will not tag inbound CxtU messages from this mobile node as being of this group, since it is not a producer of data for this group, but only a consumer. Neither will the Gateway update the mobile node's group membership upon reception of a G-diff message from a GroupDefiner.

4.2 Context Update Subscripton

Alternatively, a mobile node may also subscribe to CxtU messages generated by all other mobile nodes that are members of its group, for example, when a mobile node wishes to receive instantaneous updates of the other node's positions. In this case, however, this subscription is not handled by the Gateway, because this would cause too much overhead and compromise the Gateway's performance. Instead, the CxtU-subscriptions are forwarded to, and managed by, the Global Subscription Manager (GSub), a SDDL core node that is responsible for subscribing to all messages tagged with the corresponding group and relaying them all the mobile nodes that subscribed it. This has the advantage that a dedicated node – the GSM - will process all CxtU re-transmissions, and can be upgraded as required, and that the CxtU-message will only be processed by the Gateways that actually have the mobile subscribed nodes connected to them. Unlike, the application message subscriptions, in this case, the mobile node does not have to re-submit its subscriptions after a handover, since its subscriptions are not stored at the Gateways.

5 Fleet Tracking and Management System

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. Using this application, the company's Operations Center is able to track trajectories of its trucks in real-time, in order to optimize the trucks' itineraries, to detect and notify 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 supports a text messaging service for drivers, (e.g. to send instructions or alerts), both individually and to dynamic subgroups of vehicles/ drivers. For communication with the vehicles, several cellular operators are utilized, since in each region of the country has a particular wireless connectivity map (e.g. 2G vs. 3G), and coverage varies greatly among the operators. Thus, it is expected that vehicles may get several IPs during their journeys, may experience temporary data link disconnections, and that their links may cross cell operator-specific firewalls.

6 Fleet Tracking and Management System

In [1] we have already shown performance data of unicast and broadcast message delivery in SDDL. In our tests, so far, SDDL has proved to be a quite robust and high performance message exchange middleware even under high load of location update messages produced by the mobile nodes.

In this section, we will focus exclusively on performance tests related to the group management and communication capabilities discussed in this paper.

The performance tests were executed with following system configurations and simulation parameters: (a) 1,000, 2,000, 3,000, 4,000 MNs per Gateway; (b) 2 Gateways; (c) CxtU frequency by all MNs was twice per minute; (d) 1 GroupDefiner; (e) high and low probability of MNs group membership change (i.e. by dynamic grouping based on different parts of the generated position data by the simulated MNs). Furthermore, we evaluated two distinct implementations of GroupDefiner and Gateway: one using G-Diff message and other disseminating the whole MN´s group membership information.

6.1 Experimental Setup

To test the communication performance, in each test round we connected all simulated MNs to the Gateways and then sent unicast and groupcast "ping" messages to approximately 25% of all MNs. For each type of message we calculated the round trip delay (RTD), i.e. the difference between the instant of time the message was sent and the instant when the confirmation was received, from all the group members. We also measured: (i) the RTD between the instant the Gateway published the CxtUs in the DDS domain and when it updated its mapping after receiving the MNs´ group memberships and (ii) the group membership processing time, i.e. the Gateway internal processing time to update the mapping's data structures.

Our hardware test setup was composed by 5 desktop computers of our lab, interconnected through a 10/100/1000 Mbps switch: two of them running Gateways, one simulating all the MNs, one executing a GroupDefiner and the last one running our Control and Visualization Program, used to send the "ping" unicast and groupcast messages.

6.2 Results

The results are presented in Figure 3**,** Figure 4 and Figure 5. All times are shown in milliseconds. For the sake of better legibility, subtitles were abbreviated as follows: *RTD unicast/G-cast* means the RTD of a unicast/groupcast message; *RTD G-Diff GW-GD* means the RTD between the publication of a new CxtU and the corresponding recep-

tion of the G-Diff message from the GroupDefiner (measured at the Gateway); *RTD No G-Diff GW-GD* is also a RTD between the Gateway and GroupDefiner but now for transmitting the whole MN group membership information; *G-Diff Process time* means the group membership processing time by Gateway using G-Diff messages and No *G-Diff Process time* is the same as before but using the entire group membership information. All results are the mean value of 6 measurements. The RTD for unicast "ping" messages are presented here for the sake of better time comparison.

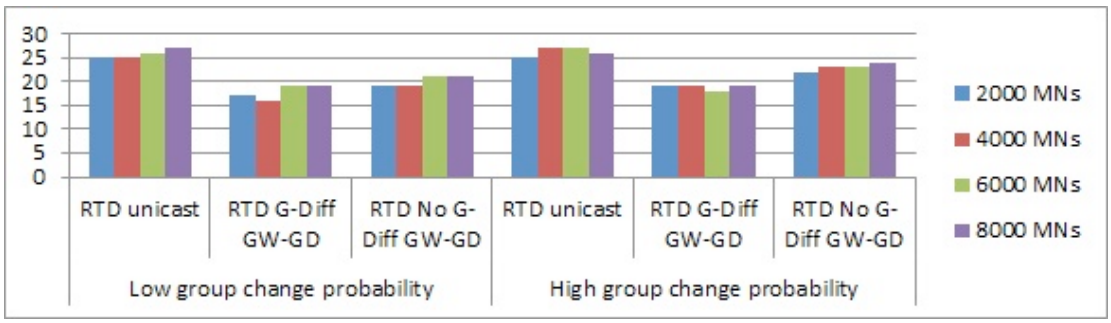

Figure 3. RTD between Gateways and GroupDefiner

As Figure 3 shows, the *RTD G-Diff GW-GD* times are lower than the other implementation that does not use G-Diff message. As expected, the network bandwidth saved when disseminating only the group membership changes caused better response times for the Gateways' update of the MNs´ group membership. In a scenario with higher group change probability, times are a little bit higher when confronted with a low group change probability. This is explained by the higher processing and network overhead that Gateways and GroupDefiner experience when MNs change groups more frequently.

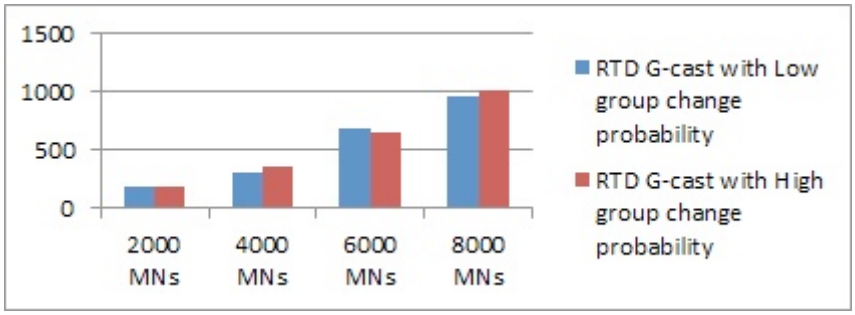

Figure 4. Groupcast RTD

Analyzing Figure 4 one can see how fast our group communication is: it allows us to send a groupcast message (and receive a reply) to approximately 2,000 MNs in approx. 1 second, when other 6,000 MNs are also connected and "flooding" the SDDL core with CxtU messages of course. This time would be larger in a real mobile network, since the wireless links of the MNs are much slower. However, this is a delay factor that is independent of our middleware. It should also be noted that the probability of group changes does not affect RTD G-cast, differently from the numbers of MNs connected.

Figure 5 shows that the G-Diff approach not only reduces the network overhead, but also enhances the Gateway performance. In the worst case, 8,000 MNs with high probability of group changes, Gateways processed a G-Diff message in 0,062641 ms. On the other hand, they spent 0,099352 ms using the whole group membership.

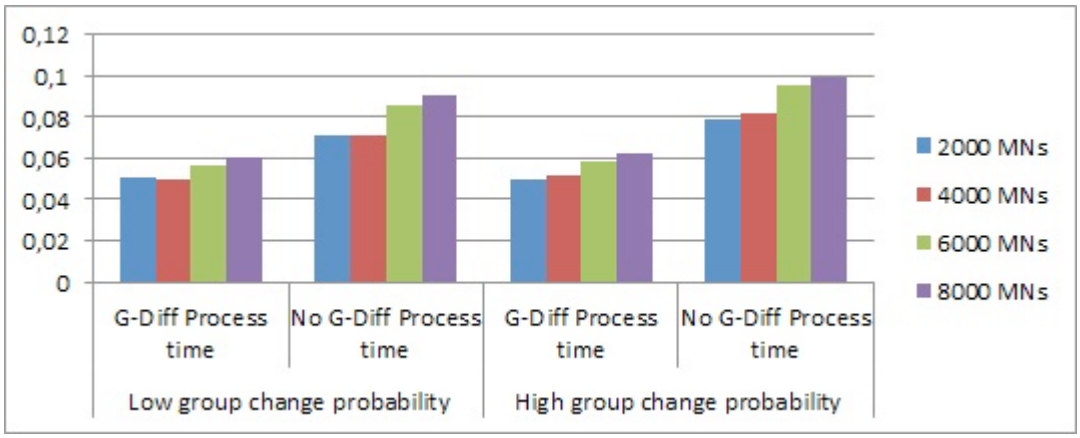

Figure 5. Group membership processing time

7 Related Work

A wide range of location tracking and mobile communication solutions can be found in the literature [4] [5] [6]. However, only few have focused on scalable group communication with the mobile nodes, and on context/location-based grouping of the nodes.

In [4], a DDS-based middleware is proposed for real-time data distribution, with a specific architectural element to support adaptation to mobile networks. This element, the Mobile DDS Client, is a lightweight version of DDS executed on all mobile devices. But due to DDS connectivity and Firewall/NAT traversal restrictions, all these Mobile DDS Clients must run in single network domain and rely on stable wireless connectivity. Moreover, in [4] there is no support for context-defined groups and groupcast communication.

REVENGE [7] is a DDS-compliant news dispatching infrastructure for mobile nodes and which is capable of transparently and autonomously balancing the data distribution load. 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 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. Therefore, 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 and is very resource-efficient. 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.

SALES [5] is a middleware for dada 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 which is enforced by the middleware. SALES defines a tree-based hierarchical architecture of nodes, so as 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 (in a ad hoc manner) with 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. Although SALES probably supports grouping of nodes and group-based communication, its hierarchical architecture suggests that efficient group communication is only possible if all MNs that are group members are within a branch of the tree.

Solar [8] is a middleware for ubiquitous computing, which was designed to be scalable in the set of communicating nodes, and is based on a self-organizing 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 "interplanetary 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 for low-latency message routing and delivery (DHT). There is also no support for dynamic group management and communication.

8 Conclusion

In this paper we have discussed the group management and communication capabilities of SDDL, our DDS-based communication middleware, and presented performance results hat confirm the feasibility of its utilization for real-world applications. Also, it was explained how the Pub/Sub capabilities can be extended to mobile nodes, while ensuring that the middleware remains with great performance and scalability. Our evaluation has shown that SDDL has a low group communication delay and the group management overhead is negligible. The encouraging results are motivating our current research work on SDDL extension and improvement.

As future work, we will investigate how it can be extended to enable also conjunctive groupcast (send to all MN $\in$ GA $\cap$ GB) and how to implement more complex subscriptions by the mobile nodes.

References

[1] L. David, R. Vasconcelos, L. Alves, R. André, G. Baptista, and M. Endler, "A Communication Middleware for Scalable Real-time Mobile Collaboration," in *IEEE 21st International WETICE, Track on Adaptive and Reconfigurable Service-oriented and component-based Applications and Architectures (AROSA)*, 2012.

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

[3] C. Julien, "The context of coordinating groups in dynamic mobile networks," *COORDINATION'11 Proceedings of the 13th international conference on Coordination models and languages*, pp. 49-64, 2011.

[4] K. Kwon and C. Park, "A proxy-based approach for mobility support in the DDS system," in *2008 6th IEEE International Conference on Industrial Informatics*, 2008, pp. 1200-1205.

[5] A. Corradi and M. Fanelli, "Adaptive context data distribution with guaranteed quality for mobile environments," *Computing (ISWPC), 2010*, pp. 373-380, May 2010.

[6] D. Stojanovic, B. Predic, I. Antolovic, and S. Dordevic-Kajan, "Web information system for transport telematics and fleet management," in *2009 9th International Conference on Telecommunication in Modern Satellite, Cable, and Broadcasting Services*, 2009, pp. 314-317.

[7] A. Corradi, L. Foschini, and L. Nardelli, "A DDS-compliant infrastructure for fault-tolerant and scalable data dissemination," in *The IEEE symposium on Computers and Communications*, 2010, pp. 489-495.

[8] G. Chen, M. Li, and D. Kotz, "Data-centric middleware for context-aware pervasive computing," *Pervasive and Mobile Computing*, vol. 4, no. 2, pp. 216-253, Apr. 2008.