PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 04/12

A Communication Middleware for Scalable Real-time Mobile Collaboration

Lincoln David Nery e Silva

Rafael Oliveira Vasconcelos

Lucas Alves

Rafael André

Gustavo Luiz Bastos Baptista

Markus Endler

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. 04/12 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena May, 2012

A Communication Middleware for Scalable Real-time Mobile Collaboration

Lincoln David Nery e Silva Rafael Oliveira Vasconcelos

Lucas Alves Rafael André Gustavo Luiz Bastos Baptista

Markus Endler

{lnsilva,rvasconcelos,lalves,randre,gbaptista,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 thus may require real-time tracking of the nodes, interaction with all participant nodes, as well as means of adaptability in a very dynamic scenario. In this paper we present a middleware communication service that supports real-time tracking of several thousands of mobile nodes, demand adaptability, as well as three modes of communication between the nodes: unicast, groupcast and broadcast. We then show Fleet Tracking and Management system and use it to evaluate our middleware.

Keywords: mobile collaboration, middleware, adaptability, dds, scalable communication

Resumo. Aplicações como gerenciamento de transporte e logística, resposta a emergência, monitoramento ambiental ou gerenciamento de equipes móveis, usam redes móveis como meio de possibilitar comunicação e coordenação entre possivelmente um grande conjunto de nós móveis. A maioria destes sistemas assim podem requerer rastreamento em tempo real dos nós móveis, interação com todos os nós participantes, como também meios de adaptabilidade em um dinâmico altamente dinâmico. Neste artigo nós apresentamos um serviço de comunicação em forma de middleware que suporta rastreamento em tempo real de milhares de nós móveis, demanda adaptabilidade, como também três modos de comunicação entre os nós: unicast, groupcast e broadcast. Em seguida mostramos o sistema de rastreamento e gerenciamento de frota e usamos o para testar nosso middleware.

Palavras-chave: colaboração móvel, middleware, adaptabilidade, dds, comunicação escalável

Table of Contents

1 Introduction

Advances in mobile communication, GPS positioning and sensor technology networks are some of the driving forces that push computing to mobile-networked systems, enabling new services and applications. Many current distributed systems such as transportation and logistics, emergency response, environmental monitoring, homeland security or mobile workforce management, employ mobile networks as means of enabling communication, collaboration and coordination among the mobile nodes - which might be people, vehicles or autonomous mobile robots [1]. With the rapid increase of embedded mobile devices many of such applications are faced with the challenge to support several hundred thousands of nodes, enabling both real-time tracking of their context/location information, and also efficient means of interaction among all nodes [8]. Moreover, in many of the systems the set of participating mobile nodes can vary a lot, as nodes may join and leave the system at any time, either due to application-specific circumstances or because of intermittent wireless connectivity. Such large-scale mobile systems thus require a scalable communication infrastructure that supports reliable and almost instantaneous data transmission between all mobile nodes, as well as monitoring and dynamic adaptation capabilities that enable automatic adjustment of the infrastructure to the very dynamic load demands caused by the mobile nodes. In this paper we present a real-time communication middleware that addresses most of these requirements. We also present a Fleet Tracking and Management application built upon our middleware and show performance results for a large number of vehicles. This work is part of a larger project called ContextNet, which aims at developing context distribution and reasoning services for mobile collaboration in large-scale systems with real-time monitoring, communication, and coordination of mobile node's movements and activities.

Paper outline: In the next two sections we present the goals of the Project ContextNet - of which this work is part - and the main characteristics of its communication layer. In section 4 we describe the Fleet Tracking and Management system, and in section 5 we present initial results on performance tests. Section 6 discusses related work on scalable middleware for such mobile systems, and in section 7 we argue about the benefits of our system. Finally, in section 8, we draw some conclusions and point to future work.

2 ContextNet Overview

Project ContextNet [2] is primarily focused on developing middleware APIs and services to address most of the challenges commented in the previous section. We believe that by finding feasible solutions for those issues, we can facilitate the development of large-scale collaborative mobile applications where mutual context and location-awareness is a requirement.

As an approach to meet these goals, we have designed the ContextNet software architecture (Figure 1), which consists of following software layers.

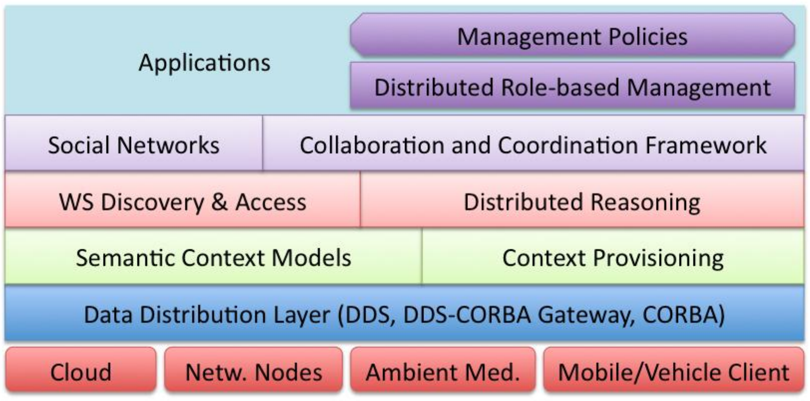


Figure 1. ContextNet general architecture.

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 SDDL

Scalable Data Distribution Layer (SDDL) employs two communication protocols: DDS's (Distribution Service for Real-Time Systems) RTPS [4] for the wired communication within the ContextNet core network, and the Reliable UDP protocol (RUDP) for the inbound and outbound communication between the core network and the mobile nodes. The DDS [4] 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.). The RUDP protocol will be explained in section 3.1. 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. The Gateway is thus responsible for managing a separate RUDP connection with each of these nodes, forwarding any application-specific message or context information into the core network, and in the opposite direction, converting DDS messages to RUDP messages and delivering them reliably to the corresponding node(s).

The PoA-Manager is responsible for two things: to periodically distribute PoA-List to the mobile nodes, and to eventually request some mobile nodes to switch to a new Gateway/PoA. The PoA-List is always a subset of the group 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. By having an updated PoA-List, a mobile node 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. To do so, the PoA-Manager subscribes to a DDS topic where Gateways periodically share their load reports. Hence, PoA-Manager's service is fundamental for the adaptive behavior of SDDL, since it can distribute load according to the current amount of mobile nodes and their attachment-distribution to the available Gateways.

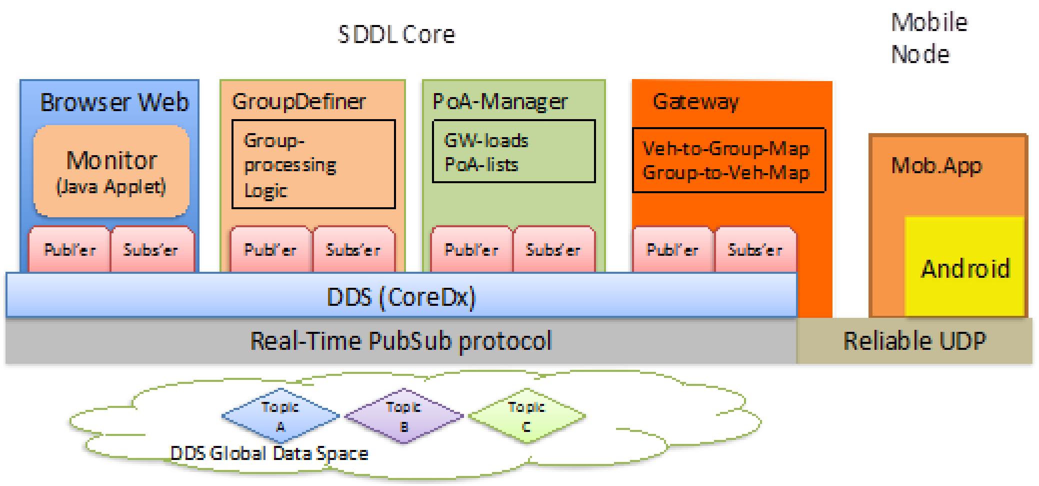


Figure 2. Nodes and protocols used in SDDL.

Apart from Gateways and the PoA-Manager(s), other nodes in the DDS domain may implement any application specific functions, such as group definition or group-cast messaging, etc. This is illustrated in Figure 2, and will be further explained in section IV, when we discuss the prototype application implemented using SDDL.

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 changes of IP addresses and 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 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 signal is 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 (if the time is shorter than a timeout).

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-initiated handover, i.e. when a mobile node is requested by the PoA-Manager to connect to a new GW, and a Mobile-initiated HO, i.e. when the mobile node spontaneously decides to connect to a new GW. In any case, it is the mobile node that actually chooses another PoA of its PoA-List, and reconnects to the corresponding GW.

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

The MTD Service is responsible to collect all messages that could not be delivery to the mobile node during its HO, but as soon the node is connected to the new GW the MTD Service will resend all the buffered messages through the DDS domain, so as to deliver them to the node through the new GW. Since not all applications require such reliable delivery, the MTD service is optional in SDDL.

3.3 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: UnicastMessages to a specific node, or BroadcastMessages, to all active nodes. In the SDDL core, sending a message is just as simple as writing into the DDS PrivateMessageTopic the message object and the ID of the mobile node or alternatively setting a broadcastFlag.

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. In out tests, so far, SDDL has proved to be a quite robust and high performance message exchange middleware even under high load of inbound location update messages, leaving us to believe that it is a good infrastructure for the development of large-scale collaborative mobile applications.

Applications developed using ContextNet can extend message delivery functionality by implementing a classifying service on a core network node. This service is responsible to classify all mobile nodes into groups, according to any criteria, such as current position, speed, or any other particular application specific context. Once the groups are defined, it is then also possible to send messages to all nodes of any group. In this case, only the Gateways that have some node in the group will subscribe to group-specific messages and will forward them to the locally served mobile nodes that are group members.

4 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 simple text messaging with drivers, to send them instructions or alerts, both individually as well as to subgroups of the vehicle's drivers. For the communication with the vehicles, the company uses several cellular operators, since in each region of the country there are significant differences of connectivity quality (e.g. 2G vs 3G), and coverage 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.

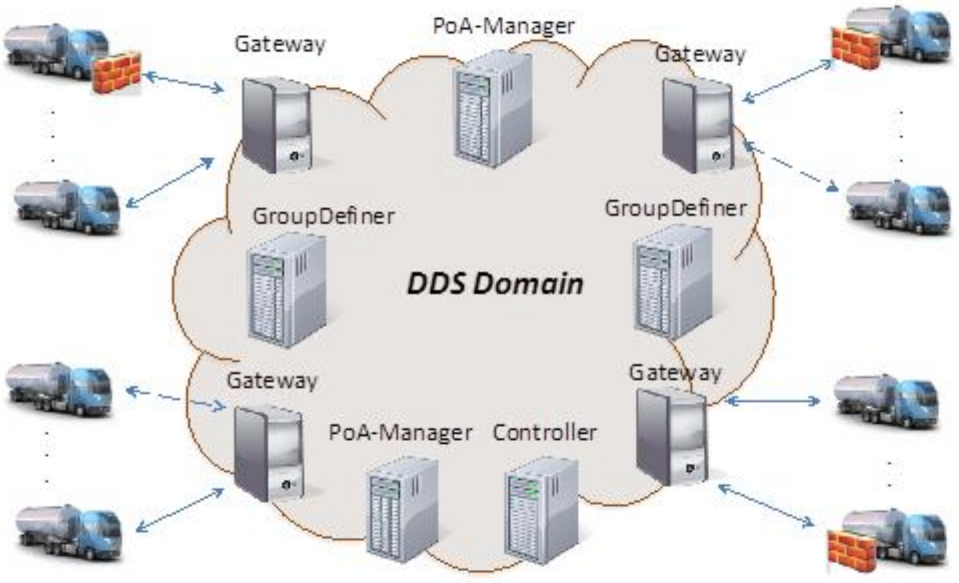


Figure 3. Fleet Management Application Architecture.

Figure 3 shows our application architecture, with all nodes in the ContextNet core network (DDS Domain) and our Fleet Tracking and Management application.

4.1 Implementation

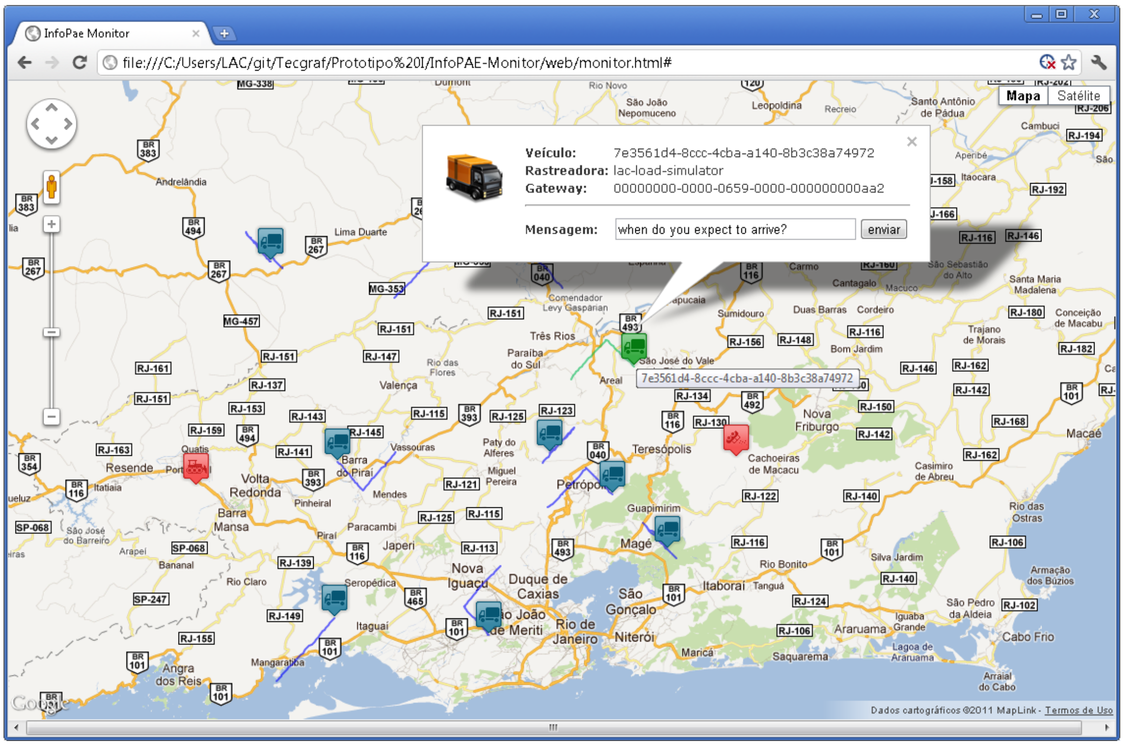
Using the SDDL as the middleware to implement this application, the mobile nodes are the company’s trucks. For this first version, we implemented this mobile client application as an Android app that runs on Android 2.3.

Once connected, the vehicle sends its location updates to the gateway every 30 seconds. This almost real-time tracking of all mobile nodes can enrich the quality of collaboration in the system. Since all participants (including others mobile nodes) are aware of the location or others’ context information about the mobile nodes, it is possible to react immediately to an abnormal situation, or start a communication with another mobile node that is close to the one´s location.

As part of the application specific behavior, our fleet management system implements two additional DDS nodes in the domain: the GroupDefiners and the Controller.

The GroupDefiners are responsible for defining the group-membership of all vehicles. To do so, they subscribe to the DDS messages published by any vehicle, and using any data contained in the message, classify the vehicle into one or more groups and share it with all Gateways in the core network, using a DDS topic. The Gateways subscribe to all the group topics so that they can forward the group-cast messages to the corresponding vehicles in the group. The current group of a vehicle can be determined, for example, by its current position (e.g. if it is inside some region).

The Controller is used by the Fleet Management Operations Center (FMOC) and is used to display, in real-time, the vehicles position on a map, as well as to send the unicast, broadcast and multicast messages to groups of vehicles. In the current version, the Controller is a Java applet that shows the vehicles´ positions on a map in a Web browser.


InfoPae Monitor
file:///C:/Users/LAC/git/Tecgraf/Prototipo%20I/InfoPAE-Monitor/web/monitor.html#
Mapa
Satélite
enviar
7e3561d4-8ccc-4cba-a140-8b3c38a74972
Dados cartográficos ©2011 MapLink - Termos de Uso

Veículo: 7e3561d4-8ccc-4cba-a140-8b3c38a74972
Rastreadora: lac-load-simulator
Gateway: 00000000-0000-0659-0000-000000000aa2

Mensagem: when do you expect to arrive?

Figure 4. Fleet Management Application Controller.

The Unicast, Groupcast and Broadcast message delivery modes supported by SDDL created means of mobile collaboration and coordination in the Fleet Tracking and Management system. By using any of these communication modes, the FMOC can send messages to vehicles to instruct them to change their itinerary and go to some specific point on the map. It also allows drivers to broadcast to the FMOC and all other vehicles, or only to the vehicles within the group defined by a region, some alerts about traffic jams, obstructions or accidents that have been witnessed at some position. Figure 4 shows a screenshot of the FMOC Controller, with some vehicles (blue icons) and informed road problems (red icons) on the map.

5 Initial Performance Tests and Results

So far, we have tested the Fleet Tracking and Management system in only some simple configurations. The goal was to get a first insight into the system's performance, in terms of communication latencies, both in the core network and on the Gateway-Vehicle links. So we tested the messages delivery delays for the unicast and broadcast types, with all the vehicles statically attached to its Gateways. This way, the PoA-Managers didn't have an important task in the test, as the handover or load balancing capabilities was not tested yet.

5.1 Configuration and Simulation Parameters

The performance tests were executed with following system configurations and simulation parameters: (a) 2,000, 4,000, 6,000, 8,000 and 10,000 vehicles per Gateway; (b) 1 and 2 Gateways; (c) LocationUpdates frequency of 2x/min, 4x/min and 10x/min; (d) 1 GroupDefiner.

5.2 Experimental Setup

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

Our hardware test setup was composed by 4 computers (virtual and real), 2 of them running Gateways and 2 others simulating the vehicles. The GroupDefiner was running on one of the simulations machine. The test took place on a 10/100 Mbps switch.

We have run experiments with most of the simulation parameters explained in the previous section. However, due to memory and processing limitations of the machines executing the vehicle simulations, we were able to simulate at most a total number of 12,000 vehicles performing 10 Location updates a minute.

5.3 Results

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

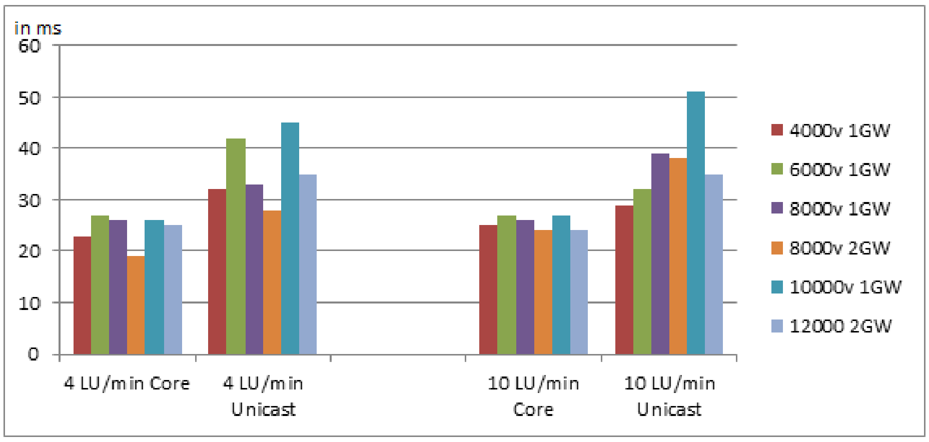


Figure 5. Core and Unicast round trip delays.

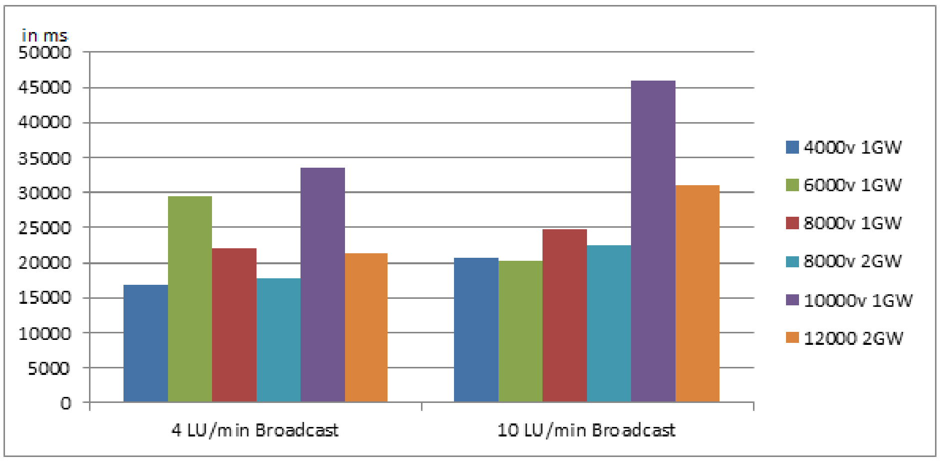


Figure 6. Broadcast round trip delays.

As Figure 5 shows, the unicast and core round trip delays are very stable for all test parameters (20-45 ms). This suggests that our system is not yet overloaded. Unicast messages to any vehicle are delivered quite fast (up to 50 ms), and the SDDL core network is yet far from saturation (< 20 ms), which means that it could handle a lot more messages. As shown by Figure 6, the broadcast delays are much higher (up to 45 sec), which is expected as all vehicles must be contacted individually and their response must be obtained until the total round trip is completed. As mentioned before, we couldn't send a broadcast message to more than 10,000 vehicles connected to a Gateway, because this caused a drop of connections during broadcast tests. We think that this is caused by the overload of the network interface, as all messages are sent almost instantaneously, probably causing a UDP buffer overflow.

6 Related Work

The *middleware* proposed in [5] also utilizes DDS for implementing real-time data distribution, and includes a specific architectural element which supports the adaptation to mobile networks. In their 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 devices. In addition to the conventional DDS software modules, the middleware of [5] also includes the *Proxy DDS Client*, responsible for managing all mobile subscriptions and forward data to and from mobile devices, *Home PDC* which stores all unsent messages while using reliable communication and device is disconnected. This middleware supports both reliable and unreliable data delivery among any node in the mobile network. Thus, the main difference to SDDL is that in our middleware the mobile devices only must execute the lightweight RUDP protocol, which is not platform dependent and is very resource-efficient. Moreover, since DDS was not designed to handle intermittent connectivity and Firewall/NAT traversal, their *Mobile DDS Clients* must run in a single network domain and in wireless networks with continuous connectivity guarantees.

SALES [6] is a middleware for data distribution, aimed at large-scale mobile systems. It was designed based on two central concepts: QoC (*Quality of Context*) and CDDLA (*Context Data Distribution Level Agreement*). In a nutshell, QoC is a *Quality of Service* related to context information distribution services, while CDDLA is a quality contract that is established between any data producer and consumer, and 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 internode communication, and hence does not take advantage of all real-time and QoS support of DDS.

Solar [7] 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 that are used, the more scalable the system is. This *middleware* uses two transport protocols, DHT Pastry (*Distributed Hash Table*), for Discovery and routing, and TCP, for "inter-planetary communication". Unlike SDDL, which uses the DDS-based core (and the Gateways), to ensure real-time and reliable delivery of data to and from the mobiles, Solar is based on DHT and TCP, which are not suited to mobile networks (TCP) and for low-latency message routing and delivery (DHT).

7 Discussion

The SDDL architecture was based on DDS in order to take advantage of its powerful data-centric approach, QoS support, and the highly optimized and scalable RTPS protocol. In particular, we implemented SDDL using CoreDX DDS1, which is a highly optimized implementation of the DDS standard specially aimed at embedded devices. Moreover, SDDL's design was driven by the desire to be simple, efficient, extensible and generic. This guiding principle can be identified by the following characteristics:

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

The extensibility is exemplified and tested by the use of the GroupDefiners. Using DDS is very simple to implement by attaching new services to the core network. For example, it is possible to implement a logging service, which captures all communications on the network and saves it to a database. This service would be totally independent of the rest of the system, making it possible to turn it off any time.

Only essential functions run on the vehicle and optimized, generic connectivity. Since the vehicle can be as simple as a J2ME or Android-powered device with limited resources, it is preferable to use an IP-based solution rather than expect a more sophisticated communication middleware on the device. Hence, we built a highly optimized UDP-based communication solution with a small footprint and tangible benefits in regard to communication reliability and Firewall/NAT traversal, which we think is the best way towards a general-purpose connectivity solution. And most important, making this protocol resilient to IP-changes and temporary disconnections without burdening the applications is a very interesting adaptability capability to mobile applications environment, where network connections are commonly not fully reliable.

1 CoreDX DDS is a trademark of TwinOaks Computing Inc.

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

Efficiency and scalability are also supported by the fact that vehicle communication issues are decoupled from the processing of their data. So far, we have limited ourselves to a rather simple context data classification and group definitions, but in the future we plan to experiment with more complex processing taking into account the data from all vehicles could be naturally incorporated into the system. Moreover, the number of Gateways can be increased with a larger volume of vehicles or inbound data, by the application.

8 Conclusion

In this paper we presented an inherently distributed communication middleware named Scalable Data Distribution Layer, aimed at supporting large numbers of data connections with mobile devices that send location updates many times a minute. Since at its core SDDL uses DDS, it directly inherits several of the OMG standard's benefits, such as data-centric data modeling, real-time and asynchronous, event-based communication through the RealTime-PublishSubscribe (RTPS) protocol, powerful data filtering and routing, as well as rich QoS configurability. The main contributions of SDDL, are thus: (i) its optimized extension of the real-time communication capabilities to mobile nodes without native DDS support, through the use of the highly optimized, and IP-address independent RUDP protocol, and (ii) an adaptive and extensible communication middleware supporting mobile node handovers, broadcast and groupcast communication modes, where the group defining logic is arbitrary.

Despite its single deployment for the Fleet Control and Management application described in section IV, with the results on our evaluation we are confident that SDDL is customizable to several other mobile collaboration applications with large groups of mobile nodes, and where high volume of context data must be efficiently processed and visualized, either in a centralized or distributed way.

References

[1] D. Stojanovic, B. Predic;, I. Antolovic et al., “**Web information system for transport telematics and fleet management,**” 9th International Conference on Telecommunication in Modern Satellite, Cable, and Broadcasting Services, (TELSIKS '09), pp. 314 - 317, October, 2009.

[2] M. Endler, G. Baptista, et al., “**ContextNet: Context Reasoning and Sharing Middleware for Large-scale Pervasive Collaboration and Social Networking,**” in Poster Session, ACM/USENIX Middleware Conference, Lisbon, December, 2011.

[3] M. Malcher, J. Aquino, H. Fonseca et al., “**A Middleware Supporting Adaptive and Location-aware Mobile Collaboration,**” in Mobile Context Workshop: Capabilities, Challenges and Applications, Adjunct Proceedings of UbiComp 2010, Copenhagen, September, 2010.

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

[5] 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.

[6] A. Corradi, M. Fanelli, and L. Foschini, "**Adaptive context data distribution with guaranteed quality for mobile environments,**" 2010 5th IEEE International Symposium on Wireless Pervasive Computing (ISWPC), pp. 8, 2010.

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