ICDT 2014

The Ninth International Conference on Digital Telecommunications

ISBN: 978-1-61208-316-2

February 23 - 27, 2014

Nice, France

ICDT 2014 Editors

Constantin Paleologu, University Politehnica of Bucharest, Romania
Andy Snow, Ohio University, USA

ICDT 2014

Foreword

The Ninth International Conference on Digital Telecommunications (ICDT 2014), held between February 23rd-27th, 2014 in Nice, France, continued a series of special events focusing on telecommunications aspects in multimedia environments. The scope of the conference was to focus on the lower layers of systems interaction and identify the technical challenges and the most recent achievements.

High quality software is not an accident; it is constructed via a systematic plan that demands familiarity with analytical techniques, architectural design methodologies, implementation polices, and testing techniques. Software architecture plays an important role in the development of today's complex software systems. Furthermore, our ability to model and reason about the architectural properties of a system built from existing components is of great concern to modern system developers.

Performance, scalability and suitability to specific domains raise the challenging efforts for gathering special requirements, capture temporal constraints, and implement service-oriented requirements. The complexity of the systems requires an early stage adoption of advanced paradigms for adaptive and self-adaptive features.

Online monitoring applications, in which continuous queries operate in near real-time over rapid and unbounded "streams" of data such as telephone call records, sensor readings, web usage logs, network packet traces, are fundamentally different from traditional data management. The difference is induced by the fact that in applications such as network monitoring, telecommunications data management, manufacturing, sensor networks, and others, data takes the form of continuous data streams rather than finite stored data sets. As a result, clients require long-running continuous queries as opposed to one-time queries. These requirements lead to reconsider data management and processing of complex and numerous continuous queries over data streams, as current database systems and data processing methods are not suitable.

We take here the opportunity to warmly thank all the members of the ICDT 2014 Technical Program Committee, as well as the numerous reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to ICDT 2014. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the ICDT 2014 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that ICDT 2014 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the field of digital communications.

We are convinced that the participants found the event useful and communications very open. We also hope the attendees enjoyed the charm of Nice, France.

ICDT Advisory Committee:

Constantin Paleologu, University Politehnica of Bucharest, Romania
Tomohiko Taniguchi, Fujitsu Laboratories Limited, Japan

Jaime Lloret Mauri, Polytechnic University of Valencia, Spain
Abdulrahman Yarali, Murray State University, USA
Michael Grottke, University of Erlangen-Nuremberg, Germany
Javier Del Ser Lorente, TECNALIA RESEARCH & INNOVATION - Zamudio, Spain
Saied Abedi, Fujitsu Laboratories of Europe Ltd. (FLE), UK
Gerard Damm, Alcatel-Lucent, USA
Dan Romascanu, Avaya, Israel
Klaus Drechsler, Fraunhofer Institute for Computer Graphics Research IGD - Darmstadt, Germany

ICDT 2014

Committee

ICDT Advisory Chairs

Constantin Paleologu, University Politehnica of Bucharest, Romania
Tomohiko Taniguchi, Fujitsu Laboratories Limited, Japan
Jaime Lloret Mauri, Polytechnic University of Valencia, Spain
Abdulrahman Yarali, Murray State University, USA
Michael Grottke, University of Erlangen-Nuremberg, Germany
Javier Del Ser Lorente, TECNALIA RESEARCH & INNOVATION - Zamudio, Spain
Saied Abedi, Fujitsu Laboratories of Europe Ltd. (FLE), UK
Gerard Damm, Alcatel-Lucent, USA
Dan Romascanu, Avaya, Israel
Klaus Drechsler, Fraunhofer Institute for Computer Graphics Research IGD - Darmstadt, Germany

ICDT 2014 Technical Program Committee

Bilal Al Momani, Cisco Systems, Inc., Ireland
Antonio Marcos Alberti, INATEL - Instituto Nacional de Telecomunicações, Brazil
Abdullah M. Alnajim, Qassim University, Saudi Arabia
Maria Teresa Andrade, FEUP / INESC Porto, Portugal
Iosif Androulidakis, MPS Jozef Stefan, Slovenia
Regina B. Araujo, Federal University of São Carlos, Brazil
Khaled Assaleh, American University of Sharjah, United Arab Emirates
Anteneh Ayanso, Brock University, Canada
Francisco Barcelo-Arroyo, Technical University of Catalonia, Spain
Ilija Basicevic, University of Novi Sad, Serbia
Carlos Becker Westphall, Federal University of Santa Catarina, Brazil
Abdelouahab Bentrcia, King Saud University – Riyadh, Kingdom of Saudi Arabia
Andrzej (Andrew) Borys, University of Technology and Life Sciences (UTP) – Bydgoszcz, Poland
Andi Buzo, University Politehnica of Bucharest, Romania
Lee-Ming Cheng, City University of Hong Kong, Hong Kong
Alberto Coen-Porisini, Università degli Studi dell'Insubria – Varese, Italy
Doru Constantin, University of Pitesti, Romania
Gerard Damm, Alcatel-Lucent, USA
Klaus Drechsler, Fraunhofer-Institut für Graphische Datenverarbeitung IGD - Darmstadt, Germany
Roger Pierre Fabris Hoeffel, Federal University of Rio Grande do Sul, Brazil
Peter Farkas, FEI STU – Bratislava, Slovakia
Christophe Feltus, Public Research Centre Henri Tudor, Luxembourg
Gerardo Fernández-Escribano, University of Castilla-La Mancha, Spain
Mário Ferreira, University of Aveiro, Portugal
Pierfrancesco Foglia, University of Pisa, Italy
Alex Galis, University College London, UK

Table of Contents

Modeling War as a Business Process with the Assistance of Service Oriented Architecture

Major Tapio Saarelainen, PhD
Research and Development Division
Army Academy
Lappeenranta, Finland
tapio.saarelainen@mil.fi

***Abstract*— The pace of war is increasing since militaries are adopting the ideas of Network Centric Warfare (NCW). Therefore, the process of war has to be modeled into the Business Process in order to benefit from available resources in real-time. There is an increasing need to automate command and control tools utilized in military operations because of the versatility and increased tempo of operations. Operations can be commanded and orchestrated with the assistance of Service Oriented Architecture (SOA). SOA is currently seen as a technology that can satisfy these needs of network centric operations (NCO). The Business Processes are chains of logic that request SOA services. This paper argues that in the case of a military setting, in order to achieve maximum impact with minimal effort (cf. downsizing), military operations need to be modeled as Business Processes (BP) (e.g., a dismounted company attack). This asks for using a Resource Manager (RM), a Scheduler and a Battle Secure Scheduler (BSS) in allocating the requested services (e.g. processing a fire support order). In the future, a single Future Force Warrior (FFW), an essential performer in military operations, can benefit from the Business Processes approach via enhanced performance, improved Situational Awareness (SA) and with decreased instances of fratricide.**

Keywords - Business Process; Service Oriented Architecture; Future Force Warrior

I. INTRODUCTION

The asymmetric nature of war requires improved capabilities in allocating available resources. This sets increasing demands for commanders executing operations in the battlespace. The requirement of precise data in location information, current data in performance capabilities of own troops and current operation status are only a few critical pieces of information commanders are depending on in military operations. The pace of war constantly increases and the need for accurate Situational Awareness is imminent and vital from the perspective of successive operations.

This paper examines how to model the phenomenon of a war as a business process assisted by means of Service Oriented Architecture. The contribution of this paper introduces a possible method to improve the overall performance of military operations by sequencing the combat services available in real-time. Typically these types of combat service sequencing systems are based on classified data. Therefore also this paper can only draw from sources available for non-restricted use.

Military commanders are depending on automatic data accruing processes and the tools to simplify complicated military maneuvers. A military operation has to be simplified into a form of an untestable Business Process (BP). This can be achieved with the successful use of Service Oriented Architecture (SOA). When an attack as an operation can be simplified in a form of a Business Process, Service Oriented Architecture can be used. The element named Resource Manager is a tool to be used in managing and allocating the existing resources.

The possibilities to compare various approaches for modelling a war as a business process assisted by means of Service Oriented Architecture are difficult to find. This is because by default value models of this kind fall in to the category of classified data. Thus the model introduced here is one of a kind. Comparing different models is out of the scope of this paper. Furthermore, the issues of network topology and energy remain outside the scope of this paper.

This paper is organized as follows: Section II discusses about the Military Operation as a process, Section III introduces Service Oriented Architecture, Section IV introduces the Business Process. Section V concentrates on combining the Business Process and the Resource Manager. Section VI explains the functions inside the Resource Manager and Section VII concentrates on security and the Scheduler. Section VIII examines benefits and drawbacks of this solution and Section IX concludes the paper.

II. MILITARY OPERATION AS A PROCESS

Military operations use real resources available. In the Business Process approach to Service Oriented Architecture, the services available correspond to existing real resources. Availability of real resources in a given place time is limited and needs to be carefully scheduled. Usually, SOA services are assumed to be independent of each other but this assumption is no longer valid if SOA services represent real resources. The Resource Manager is a necessary element in SOA architecture. An example of a demanding process, in which the timing and optimal use of resources is critical, is a dismounted company attack. Successful performance requires that the requested services (e.g., processing a fire support order) are allocated timely and accurately. This sets demands for enhanced Situational Awareness. In the utilization process of SOA, challenges of real-time SOA must be solved [1]. To successfully execute BPs, the Business Process Execution Language (BPEL) is required, as argued in [2].

III. SERVICE ORIENTED ARCHITECTURE (SOA)

Service Oriented Architecture promises to enable utilizing and operating complicated systems. SOA enables organizations and entities to enhance interoperability, collaboration, see [1], and foster the reusing of components and interfaces. SOA can be used in service collaboration. With the correct framework SOA allows publishing services in a service registry and exchanging data through the Simple Object Access Protocol (SOAP) [1]. SOA offers a flexible solution for systems integration, applications, protocols, data sources and processes to form a cohesive system that supports the execution of critical BPs [2]. SOA can be used as a collaboration tool in crises management and industrial environments if the challenges of real-time SOA [2] are solved.

In order to successfully execute BPs, the Business Process Execution Language (BPEL) is required, as argued in [3]. In military systems, the adoption of SOA principles can beneficially result in the overall improvement of system flexibility and maintenance. SOA provides the user with richer information sets via the ability of Web Services to reach out through the networks, see [4]. In the process of achieving greater interoperability, SOA can be used by utilizing service oriented migration and reuse technique, described in [5].

In Network Centric Warfare (NCW) contexts, SOA has been recognized to act as an enabler of services. SOA is an architecture style that encourages loose coupling between services to facilitate interoperability and the reuse of existing resources as described in [6]. SOA is seen as a tool in enabling agility to handle the changing dynamic evolution needed in network enabled capability, see [7]. The concept NCW can be viewed as an integration of assets to fulfil a mission objective, as discussed in [8]. NCW fosters SOA to achieve flexible forces, which are constantly ready and deployable, capable of dynamic changes and evolution to achieve realizable effects. To benefit from SOA in an optimal way, organizations require a comprehensive and applicable SOA governance framework to implement the management and control mechanisms in the system, as argued in [9].

It has been pointed out [10] that Shared SA is in central role for network-enabled capabilities, as described in [10]. In NEC, SOA is most commonly realized through Web Services GUIs, as discussed in [11], using Extensible Markup Language (XML) formatted documents, see [10]. As evident, XML WS have been recently used to implement SOA enabling the building of BPs by dynamically calling services from the World Wide Web.

SOA is an open concept and supports plug-and-play capabilities of heterogeneous software and hardware components, with the implementation of Web Services, which is probably so far the most popular implementation of SOA, as discussed in [12]. For this reason, SOA has been selected as the architectural solution for the C4I2SR systems for the Finnish Defence Forces [13]. SOA is seen as an enabler in crises management organizations for delivering data and services across political, organizational and cultural boundaries as well as addressing the issues of information sharing regardless of where required data is stored, as concluded in [14]. The global information grid is an essential vehicle in the execution of SOA and for the transformation of data.

In tactical operations, the significance of the real-time location data plays an important role. The tools available include different types of Tactical Battle Management Systems for the dismounted combat to produce the location information of own troops and precise target designation. Obviously, the tools for target designation and air-land coordination are necessary requirements for success in operations as described in [15]. Furthermore, air-to-ground communications are described in [16].

Requirements related to improved Situational Awareness, communications and networks are described in [17], grid computing in the battlespace plays an important role as described in [18], and enhancing squad communications with the assistance of smart phones is described in [19]. Lastly, multiplication of various technologies is introduced in [20]. Their overall purpose is to increase the performance of a Future Force Warrior. The inputs of all these tools and networks can be processed with the assistance of tools used in Business Processes and with the assistance of Service Oriented Architecture.

IV. BUSINESS PROCESS

Because of the central role of the business process in SOA, the main ideas concerning the BP are described in this and the following chapter. In a military environment, an example of utilizing a Business Process approach embedded to SOA is a military operation consisting of sequenced phases, for instance, in an operation labeled as a dismounted company attack, as illustrated in Fig. 1.

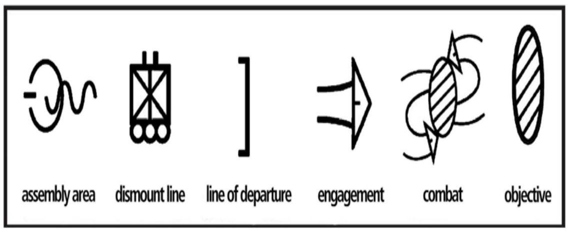


Figure 1. Dismounted company attack as a Business Process [21].

The variety of services used in BPs may be in operational use of a single unit or several units at the same time. This requires an efficient orchestration of services to maintain service control. SOA can be seen as an enabler in the process of executing military operations as BPs.

A planned dismounted attack usually starts from the assembly area, moves on to the dismount line, via a line of departure, advances to engagement, results in close combat and ends when the set objective is reached. The SOA BP approach can increase the probability of success of an attack by empowering the human-based decision-making process by computers. This can enable an optimal use of resources, and thereby improve overall performance in operations.

The offered services during an advancing dismounted attack are listed in Table 1. Most of these services can be pre-programmed to concern the wanted product-line FFW level. The company commander utilizes various services (fire support orders, location services, medical care, resupply, evacuation, geographical information system -map-service, Blue Force Tracking) while executing the commanded attack from the assembly area to the objective. Table 1 illustrates possible services available for a dismounted company attack.

TABLE I. LIST OF PRE-PROGRAMMED AND ADDITIONAL SERVICES IN A DISMOUNTED COMPANY ATTACK.

Area of dismounted attack		*Basic services for the Warriors*	*Advanced services (platoon leaders and above)*
1	Assembly area	Location data, terminal guidance to the dismount line	Blue Force data, evacuation
2	Dismount line	Blue Force data, evacuation, resupply	Blue Force data, evacuation
3	Line of departure	Blue Force data, evacuation, resupply	Precision location data, fire support
4	Engagement	Precision location data, fire support	Air-strike
5	Combat	Reinforcement, evacuation, resupply	Air-strike, preparing instructions to the following mission
6	Objective	Evacuation, resupply, reinforcement, precise location data	Air-strike, next mission objective and its time-frame

Fulfilling a requested service asks for the requested service to be available and within range. When dealing with Fire Support Orders (FSO), the range limitations of artillery units are critical. An artillery unit has to be located within an appropriate range, and it has to be ready to intake Fire Support Orders and execute them in the required time frame, precisely as ordered.

V. THE BUSINESS PROCESS AND THE RESOURCE MANAGER

The orchestration of Business Processes requires a tool for allocating resources, the Resource Manager (RM). The tool has been described in [21]. The RM sorts out and lines up the requested services. As militaries implement the framework of network centric warfare with a continuing need to automate the command and control (C2) tools utilized in military, the tempo of operations must be taken into consideration. The collected data need to be processed, analyzed, verified, transmitted, and finally stored. SOA can be identified as a technology that can satisfy these needs of network centric operations. The starting point in the BP approach to SOA is that the main business operations of the organization are described by SOA BPs. The Business Processes are chains of logic that request SOA services. In case of a military setting, the Business Processes represent military operations as depicted in Fig. 2.

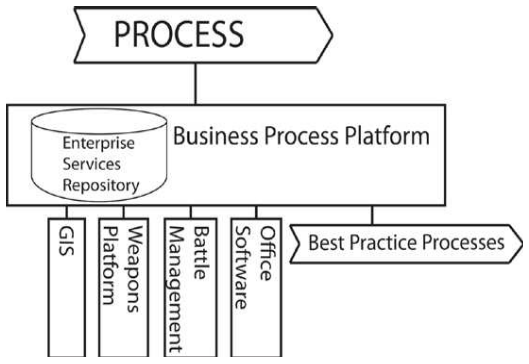


Figure 2. Business Process Platform as a service enabler.

SOA -technology involves assisting processes performed in military operations. Business processes are executed in a specific business process platform. Services and platforms, such as geographical information-services, weapons platforms, and battle management systems, are linked to the Business Process Platform to produce the best results to the ongoing processes. When an FFW can benefit from the possibilities offered by a successful adoption of BP and SOA, the result can be improved overall performance in military operations.

Fig. 3 describes how the Business Process approach can improve the performance of a Future Force Warrior (FFW). Several battlespace sensors gather data from the battlespace. The collected data are then automatically transmitted to be analyzed in a command post. Various battlespace sensors transmit data to a context-aware reasoning layer. In this layer, data are converted to context and an inference engine transmits the data to a ubiquitous main layer for analyzing purposes. The data are verified, analyzed and transmitted as information for the execution of the operation.

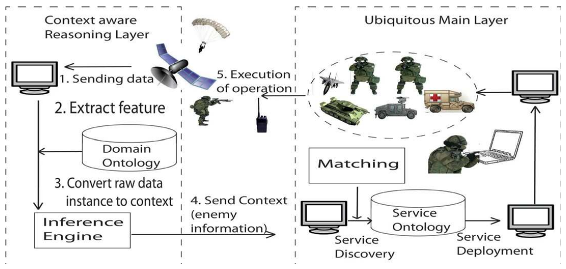


Figure 3. Increased FFW Performance can be gained via successful data utilization and analyzing process.

VI. INSIDE THE RESOURCE MANAGER

Several of the needed services require real-time resources. These services can be identified, for example, as collecting SA data and issuing fire support orders. Thus the services and their use must be scheduled and sequenced to sustain the processes. The RM sorts out and lines up simultaneous requests concerning the requested service. The

RM serves as the element, which provides the needed services for User Groups (UGs). Services can be either pre-programmed on demand or be available on request basis. The RM as a tool is located at the battalion level. The user groups send a request for the demanded service. The UGs are then authenticated, and their privileges are verified, and then the request is transmitted to the RM. The key functions of the RM are: 1) to receive the request of a required service, 2) to organize the line of user groups in the correct order depending on the UGs' privileges and battle-situation, 3) to check whether the service is available and within range, 4) to provide the User Groups with the answer, which is either the requested service or a rejection of the service. Fig. 4 illustrates the process.

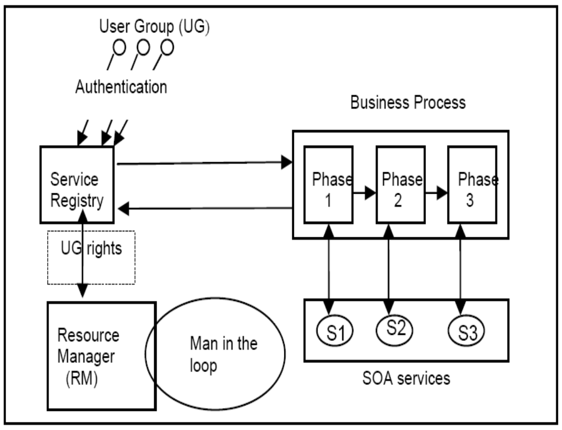


Figure 4. The main idea of the Business Process approach to Service Oriented Architecture.

The RM functions as a fully automated chain of functions in certain processes [21]. The key function of the person in the loop is to monitor the flow of events and to interfere to the chain of events if an unpredicted anomaly occurs in the process. As the RM is a critical resource, it must be physically protected against enemy actions.

The role of the RM is central in the allocating of resources in the BP process. The RM communicates with four intermodules. The RM graphical user interface provides the core interface between all the presented modules and the Local Area Network (LAN), as shown in Fig. 5. The LAN is utilized as a battlespace network or a community network as it can be used on wide area of networks. Yet, the sharing of networking environment and its resources remains challenging. Searching for information and asking for resources become challenging when lacking proper search mechanisms. Each module has pre-defined and precise functions. First, the file and resources sharing module communicates with the RM GUI in conjunction with the sharing and the download module. The file and resource transfer and download module supports and enables the transfer or download of the searched file or resource from the other node connected to the network. The shared files and resources are listed on the RM GUI, where the listed and downloaded files can be examined. It is obvious that the same identified services are requested simultaneously. Therefore, the composition of the RM needs to be stable and reliable. Fig. 5 illustrates the composition and function of the RM.

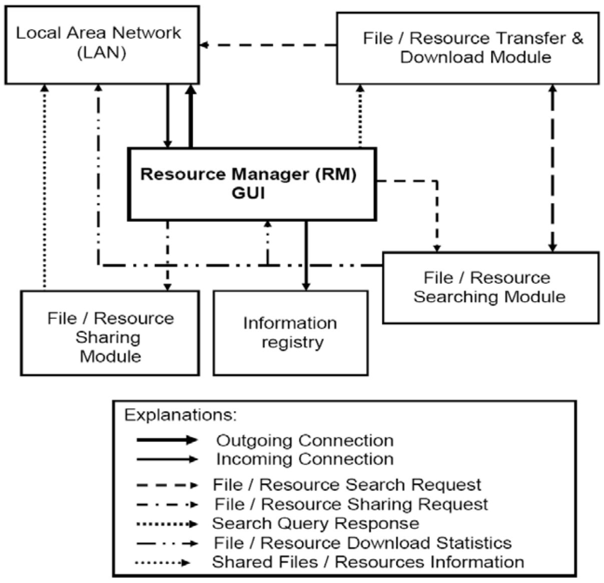

Local Area Network (LAN)
File / Resource Transfer & Download Module
Resource Manager (RM) GUI
File / Resource Searching Module
File / Resource Sharing Module
Information registry
Explanations:
Outgoing Connection
Incoming Connection
File / Resource Search Request
File / Resource Sharing Request
Search Query Response
File / Resource Download Statistics
Shared Files / Resources Information

Figure 5. The composition and function of the RM.

The example below in Fig. 6 depicts the processing of fire support order requests inside the RM as an informal Specification and Description Language (SDL) diagram. This action performed by the RM is essential to proceed in the process of offering requested service/s [21].

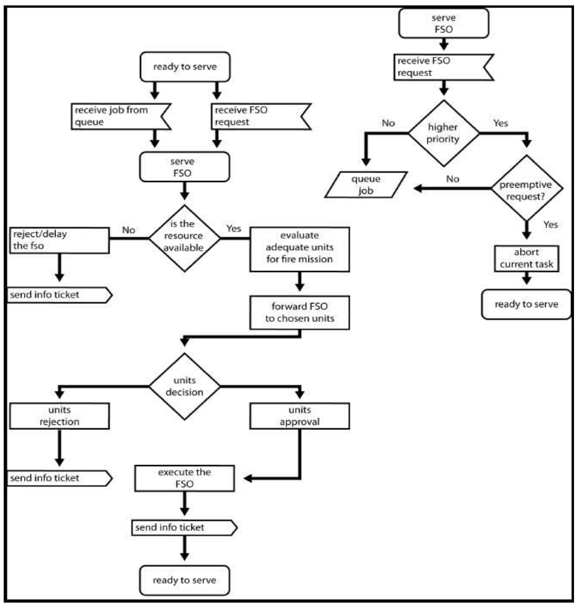


Figure 6. The processing of Fire Support Order requests in RM in an informal SDL diagram.

Each request has a time-stamp and own identification and the request also contains route data and is traceable whenever tracking data are required. Each request is categorized according to an urgency class and its execution process is monitored and evaluated continuously. Once the request has been executed, it will be filed as a completed task in the common database. The tracking data of the completed request can be retrieved for analyzing purposes at any time by the system operator.

VII. Security and Scheduler

To account for operational security, there are protocols to identify the credentials of the requester entity by applying a security, authentication and agreement tool embedded in the RM. Before any tasks are given to be executed or resources are allocated for use, the task or resource request goes via the described system, as presented in Fig. 7. An incoming task passes through a preliminary phase, in which it is checked and identified. Once the task has been verified and approved and sent from a trusted and secure cooperation entity, it will be processed via a series of approval and authorization policies.

Security issues remain vital also when dealing with unmanned aerial vehicles utilized in Network Centric Warfare at tactical level as described in [22]. The accrued data have to be secured to be intact and coherent when passing different interfaces from the sensor to the shooter.

The described process ends with a phase in which a common language and tools are selected and then the given request moves forwards inside the RM. The overall description of the whole concept consists of three major parts and functions: 1) SA comprehending the existing solutions and tools, 2) command and control tools, and, lastly, 3) information repository. These three together enable the command and control process and saving of logdata for further analyses. These functions presuppose the RM and the Scheduler to share and distribute the tasks and resources.

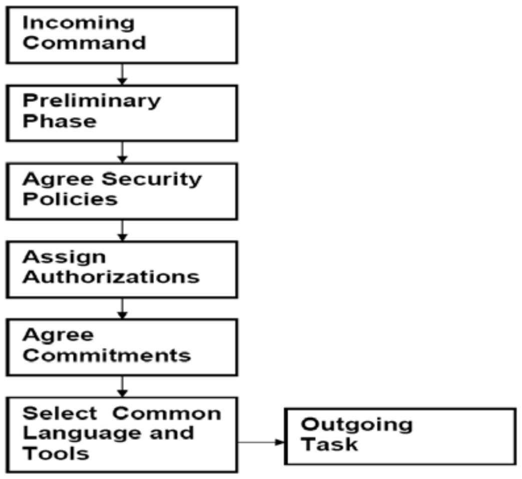


Figure 7. Security, authentication and agreement system.

To provide for the requested service, the RM requires one more component [21]. This critical component for the military use of SOA which relies on the utilization of the RM is called the Scheduler. The role of the Scheduler is to coordinate processes to maximize the performance of resources and to reduce fratricide and collateral damage. The Scheduler enables militaries to execute various operations simultaneously but still under a strict command and control. The issue of simultaneously operations is solved by the element named Battlefield Secure Scheduler (BSS). This component uses two different methods of sharing calendar, Pre Shared Scheduler (PSS) and Dynamic Schedule Update (DSU). The Scheduler functions together with the RM and utilizes SOA as a process. These elements can be recognized in Fig. 8, which introduces the process from an incoming command/task to an outgoing command/task.

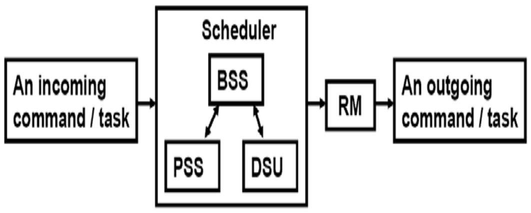


Figure 8. The elements inside the scheduler and the permeable command and control -process.

VIII. Benefits and Drawbacks of This Solution

The delicate system introduced can malfunction for various reasons. Challenges related to energy have to be solved to enable the function of different processes. The orchestration of the system can also fail because of intentional enemy action (jamming, a virus, a worm). The system needs to be equipped with an analyzing program, which indicates when the system functions properly. If the system malfunctions and retrieving the capabilities becomes impossible, the system becomes useless for an FFW. This asks for an easily replaceable and fault-tolerant system with inbuilt check-in routines. Otherwise, traditional methods in orchestrating services need to be adopted.

By adopting SOA and embedding business processes into the existing command and control -system, the overall performance of military operations can be improved. With the assistance of the RM, limited military resources can be allocated more efficiently to the users requiring for services. When the new invented tool, the Scheduler is implemented together with the RM into the BPs, the performance of the system can be significantly increased. The allocated resources available can be used optimally. This means shorter execution times, and a higher amount of data for improved decision making. The overall system performance can be optimized with the assistance of these tools.

Offering a service of ubiquitous computing to battlespace commanders increases the possibility to utilize the resources available. This fosters a rapid decision-making process especially, when SOA can be embedded in the decision-making systems. As described in [14], SOA must deliver a solution that crosses existing boundaries as well as address the issues of information sharing regardless of where that information is stored. A BP -like orchestration of systems and services can improve the overall performance of military operations executed. This can result in improved overall performance capabilities while executing missions in the battlespace benefitting from SOA, the RM and the Scheduler.

By adopting these introduced new elements into BMS together with SOA, it may be possible to gain improved capability to execute operations. This can also mean reduction in time related to allocation of resources. This can result improved overall performance and minimal execution times of operations. Besides, with the improved level of SA, fratricide and collateral damage can be reduced.

The system presented here is free to be adopted and tested. To create a functioning, automated business process applicable for future battlespace purposes requires future work. This is briefly tackled in the ensuing section.

IX. Further Work

So far, this system has not been field tested for considerable funding is needed to execute the tests. One would hope such funding became widely available since the need for automated systems and allocation of ever limiting resources force militaries to discover the performance offered via SOA. When the process of war has been modeled to resemble a Business Process, the performance of FFWs can be optimized with the assistance of processes assisted with the SOA. The result of this can be seen as an agile and modular military performer with improved capabilities and improved Situational Awareness, and the capability to utilize ever diminishing resources more optimally with decreased instances of fratricide.

Further work related to modeling a war as a business process must concentrate on security issues of software. Worms and viruses pose an increasing threat in digitized battlespace. Issues such as adequate level of constant energy flow and protection against violations caused by electronic warfare must be studied, tested and finally solved before adoption of the system in operative use.

Funding and human resources are required to run the validity tests of the introduced system. Firstly, tens of thousands of simulation laps in each scenario type are required before implementing the introduced system/prototype into any real-time military exercise performed. Secondly, once resources have been invested in implementing the system introduced here, the follow-up paper cannot any longer be accessed in any public domain data sources.

References

[1] M. Panahi, W. Nie, K-J. Lin, The Design and Implementation of Service Reservations in Real-Time SOA, in Proceedings of IEEE International Conference on e-Business Engineering (ICEBE2009), 21-23 Oct. 2009, Macau, pp. 129 – 136, doi 10.1109/ICEBE.2009.26.

[2] A. Gravel, X. Fu, and J. Su, An Analysis Tool for Execution of BPEL Services, in Proceedings of the 9th IEEE International Conference on E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC-EEE2007), Tokyo, pp. 429 – 432, doi 10.1109/CECEEE.2007.19.

[3] W. T. Tsai, Q. Huang, J. Xu, Y. Chen, and R. Paul, Ontology-based Dynamic Process Collaboration in Service-Oriented Architecture, in Proceedings of IEEE International Conference on Service-Oriented Computing and Applications (SOCA'07), 19-20 June 2007, Newport Beach, CA, pp. 39 – 46, doi10.1109/SOCA.2007.35.

[4] M. Jiang, and A. Willey, Service-Oriented Architecture for Deploying and Integrating Enterprise Applications, in Proceedings of 5th Working Conference on Software Architecture (WICSA 2005), Pittsburg, PA, pp. 272 - 273, doi 10.1109/WICSA.2005.60.

[5] G. Lewis, E. Morris, and D. Smith, The Service-Oriented Migration and Reuse Technique (SMART), Technical report CMU/SEI-2005-TN, Software Engineering Institute, in Proceedings of 13th International Workshop on Software Technology and Engineering Practice, September 2005, Budapest, pp. 222 – 229, doi 10.1109/STEP.2005.24.

[6] M. Medlow, Extending Service-Oriented Architectures to the Deployed Land Environment, Journal of Battlefield Technology, vol. 13, No. 1, March, 2010, pp. 27 – 33.

[7] L. Liu, D. Russell, N. Looker, D. Webster, and J. Xu, Delivering Sustainable Evolutionary Service-Oriented Architecture for Network Enabled Capability, in Proceedings of International Workshop on Verification and Evaluation of Computer and Communication Systems (VECoS2008), pp. 1 – 11.

[8] D. Russell, N. Looker, L. Lu, and J. Xu, Service-Oriented Integration of Systems for Military Capability, in Proceedings of 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), 2008, pp. 33 – 41, doi 10.1109/ISORC.2008.45.

[9] F. Hojaji, M. Reza, and A. Shirazi, Developing a More Comprehensive and Expressive SOA Governance Framework, in Proceedings of the 2nd IEEE International Conference on Information Management and Engineering (ICIME), 16-18 April 2010, Chengdu, pp. 563 – 567, doi 10.1109/ICIME.2010.5478046.

[10] K. Lund, A. Eggen, D. Hadzic, T. Hafsoe, and F. T. Johnsen, Using web services to realize service oriented architecture in military communication networks, Communications Magazine, vol 45, Issue 10, pp. 47 – 53, doi 10.1109/MCOM.2007.4342822.

[11] J. He, I-L. Yen, T. Peng, J. Dong, and F. Bastani, An Adaptive User Interface Generation Framework for Web Services, in Proceedings of IEEE Conference on Congress on Services Part II, 2008. SERVICES-2, Beijing, 23-26 Sept. 2008, pp. 175 – 182, doi 10.1109/SERVICES-2.2008.23.

[12] E. Zeeb, A. Bobek, H. Bohn, and F. Golatowski, Service-Oriented Architectures for Embedded Systems Using Devices Profile for Web Services, in Proceedings of IEEE 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW '07), 21-23 May 2007, Niagara Falls, Ont., pp. 956 – 963, doi 10.1109/AINAW.2007.330.

[13] J. Jormakka, and J. Lucenius, Possibilities for improving dependability of C4I2SR systems based on Service Oriented Architecture, in IEEE Transctions 2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, pp. 315 – 324, doi

10.1109/ComputationWorld.2009.74., ISBN: 978-1-4244-5166-1.

[14] B. Farroha, and D. Farroha, SOA as a catalyst to empower the Warfighter through improved enterprise data access over the GIG, 3rd Annual IEEE Systems Conference (SYSTEMS2003), 23-26 March 2009, Vancouver, BC, pp. 48 – 53, doi 10.1109/SYSTEMS.2009.4815770.

[15] Sagem Showcasing Systems for Demanding Infantry Forces Mission, www.soldiermod.com, Vol 11 Summer Autumn 2013, pp. 42 – 43.

[16] M. Phillips, Air-to-Ground and Ground-to-Air Communications, Military Technology, Vol XXXVII, Issue 6/2013, pp. 66 – 67.

[17] P. Donaldson, D-P. Merklinghaus, and S. Nitschke, Technology Enablers for Global Special Forces Capability, Military Technology, Vol XXXVII, Issue 5/2013, pp. 65 – 74.

[18] P. Cordwainer, Grid Computing on the Battlefield, Military Technology, Vol XXXVII, Issue 4/2013, pp. 74 – 75.

[19] J. Antal, Enhancing Squad Communications – Seven Considerations for the Militarisation of COTS Smartphones. Military Technology, Vol XXXVII, Issue 5/2013, pp. 58 – 61.

[20] D. Alexander, Toward NATO Forces 2020: Combat Multiplication Technologies for Smart Defence, Military Technology, Vol XXXVII, Issue 10/2012, pp. 84 –86.

[21] T. Saarelainen, Improving the Performance of a Dismounted Future Force Warrior by Means of C4I2SR, Doctoral Dissertation, ISBN 978-951-25-2457-0.

[22] G. Ebbutt, Network Centric Warfare at the Tactical Level, Military Technology 2/2011, pp. 106 – 110.

Link-Aware NICE Application Level Multicast Protocol

Amr Naser, Mohamed Rehan
Intel Labs
Cairo, Egypt
{amrx.naser, mohamed.m.rehan}@intel.com

Dina Helal, Ayman El Naggar
German University in Cairo
Cairo, Egypt
{dina.helal, ayman.elnaggar}@guc.edu.eg

***Abstract*—Multicast is one of the most efficient ways to distribute data to multiple users. There are different types of Multicast such as IP Multicast, Overlay Multicast and Application Layer Multicast (ALM). In this study, we focus on Application Layer Multicast where the multicast functionality is implemented at the end user. We introduce a new ALM protocol, Link Aware-NICE (LA-NICE), which is an enhanced version of the NICE protocol [1]. NICE is a recursive acronym which stands for NICE is the Internet Cooperative Environment. LA-NICE protocol takes into account the fact that different links can have different bandwidths and this fact can be used to improve multicast message delivery and minimize end-to-end delay. OMNeT++ simulation framework [2] was used to evaluate LA-NICE. The evaluation is done through a comparison between LA-NICE and NICE. The simulation results showed that LA-NICE produces an increase in the percentage of successful message delivery ranging from 2% to 10% compared to NICE. Also, LA-NICE has less average delay and less average message hop count than NICE which reduces the overall latency of message delivery.**

***Keywords*—Application Level Multicast, Multicast tree, Overlay networks , Link Aware, Hop count, NICE, Scribe, OMNeT.**

I. INTRODUCTION

As the number of Internet users increases, data delivery over the Internet becomes more challenging as networks get more overloaded and congested. Currently, data exchange through the Internet is mainly based on unicast (point-to-point between two computers). So, if millions of users try to stream an important broadcast event like the world soccer cup, instead of broadcasting the data to all users, the data source sends a copy of the data to each of the users so the source keeps transmitting the same packet a million times. This leads to redundant traffic in the network in addition to overloading the data source resulting in inefficient data delivery and an increase in packet loss. Multicast was introduced as an alternative to unicast in such cases. In multicast, the source sends contents to a sub-server set and each one of those sub-server set forward the content to a different group of users. There are several types of multicast such as IP, overlay, and application level. In IP Multicast, the multicast process is implemented at the IP level during packet transmission. IP multicast provides an efficient multicast technique. However, it was never widely deployed in the Internet due to multiple reasons including the fact that it requires changes at the infrastructural level which slows down the pace of deployment. Also IP multicasting introduces high complexity and serious scaling constraints at the IP layer in order to maintain a state for each multicast group. As a result of the non acceptance of IP Multicast, the Application layer multicast (ALM) approach was proposed. ALM, also called End-System Multicast, was proposed as an alternative implementation of the multicast technique to the IP Multicast implementation. ALM builds a virtual topology on top of the physical Internet to form an overlay network. Each link in the virtual topology is a unicast link in the physical network [3]. Therefore, the IP layer provides a unicast datagram service, while the overlay network implements all the multicast functionality such as dynamic membership maintenance, packet duplication and multicast routing [4].

NICE [1] is an ALM protocol that arranges the set of members in a multicast group into a hierarchical control topology. As new members join and existing members leave the group, the basic operation of the protocol is to create and maintain the multicast tree hierarchy. The NICE hierarchy is created by assigning members to different levels (or layers) as illustrated in layer 0, figure 1. Layers are numbered sequentially with the lowest layer of the hierarchy being layer zero (L_0). Members in each layer are partitioned into a set of clusters [5]. Each cluster is of size between k and 3k-1 members, where k is a constant (usually k=3), and consists of a set of members that are close to each other. Further, each cluster has a cluster leader. The protocol chooses the center of the cluster to be its leader, i.e., the cluster leader has the minimum distance to all other members in the cluster. This choice of the cluster leader ensures that a new joining member is quickly able to find its appropriate position in the hierarchy using a very small number of queries to other members. The leaders in level i are the members of level i+1 in the tree, so all the leaders in L_0 belong to L_1 and their leaders belong to L_2 and so on until there is only 1 leader which is the Rendezvous Point (RP) in the highest level of the tree. Since each cluster in the hierarchy has between k and 3k - 1 members, a host that belongs only to L_0 layer peers with O(k) other hosts for exchange of control messages. In general, a host that belongs to layer L_i and no other higher layer, peers with O(k) other hosts in each of the layers L_0....L_i, which results in control overhead O(k*i) for this member. Hence, the cluster-leader of the highest layer cluster peers with a total of O(k*logN) neighbors, which is the worst case control overhead at a member. NICE mainly focuses on minimizing end-to-end delay. This is done by computing the distance between the nodes and constructing the tree such that nodes close to each other get assigned to the same cluster. This technique minimizes end-to-end delays as the cluster leader is always centered in the middle of the cluster where the distance between it and the rest of the cluster members is minimum.

The rest of the paper is organized as follows. Section 2 describes Link-Aware NICE protocol. Section 3 presents the simulation design, the implementation and the evaluation and test results. Finally Section 4 presents the conclusion based on the simulation results.

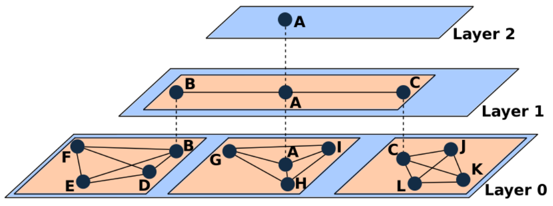


Fig. 1: Hierarchical arrangement of hosts in NICE [6]

II. Link-Aware NICE

LA-NICE takes into account the fact that different links can have different bandwidths and uses this fact to improve multicast message delivery and minimize end-to-end delay. As shown in figure 2, LA-NICE doesn't change the cluster structure or how the cluster splits or merges, it focuses on two phases in the tree management which are the member join and the tree maintenance phases.

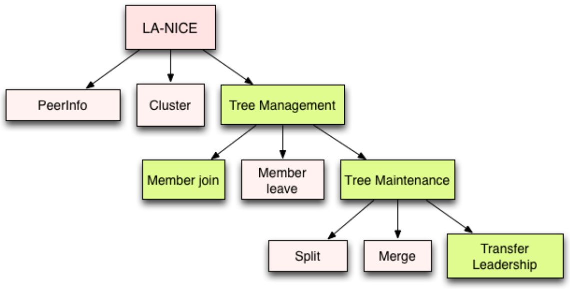


Fig. 2: LA-NICE code structure

A. Multicast Tree Member Join Procedure

When a new node wants to join the multicast tree, in the original NICE algorithm, it contacts the RP with its join query, The RP responds with the hosts that are present in the current highest level of the tree L_i. The joining host then contacts all members in L_i to locate the member closest to it. This member then informs the joining host of its other members in L_{i-1} layer and so on recursively until the joining host finds its position in the lowest level of the tree. Meanwhile, the joining host gets peered temporarily with the RP as soon as it starts communicating with it to get the muticast messages sent until it finds its appropriate position in the tree. This ensures that the joining node gets connected to the tree as soon as it requests to join. Right before joining the tree in the lowest level and after knowing where exactly it will join, the member node requests to be disconnected from the RP and requests to join its appropriate parent in the tree.

LA-NICE modifies the member join procedure of NICE. NICE members join the closest clusters based on round trip time (RTT) measurements. The RTT of sending a message is the time it takes for the message to be sent plus the time it takes for an acknowledgment of that message to be received. The RP gets a list of the clusters leaders ordered by distance and assigns the new host to the closest cluster to it. In LA-NICE, in a tree of i levels, the RP gets a set of potential clusters (PC) that the new node can join using (1) where PC is a set of clusters of size i. Ln is the leader of cluster n. Then the RP checks the bandwidth of the leaders of the potential clusters and assigns the new node to the cluster with the highest ratio of leader bandwidth/number of cluster members as shown in (2).

$$PC = min_{RTT}(New\ node, Ln) \tag{1}$$

$$Selected\ cluster = max_{bandwidth/cluster\ size}(PC\ leaders) \tag{2}$$

B. Multicast Group Member Leave Procedure

When nodes leave the tree, they can leave either gracefully or ungracefully. A graceful leave, which is when a host leaves the multicast group after notifying all the clusters it has previously joined that it's leaving. On the other hand, an ungraceful leave occurs when member fails without being able to send out a leave notification to its clusters, the cluster members then manage to detect this departure when they don't receive HeartBeat message from that member. A HeartBeat message is a periodic message sent by every member to the rest of the multicast group informing them that it still exists in the group. If a cluster leader left the group, this could lead to a partition in the tree so a new leader needs to be chosen faster. A new leader of the cluster is chosen depending on who is estimated to be closest to the center among these members.

C. Multicast Tree Maintenance

A cluster leader periodically checks the size of its cluster, and appropriately splits or merges the cluster when it detects a size bound violation. If a cluster exceeds the cluster size upper bound 3k - 1, it gets split into two equal-sized clusters. Given a set of hosts and the distances between them, the cluster split operation partitions them into subsets that meet the size bounds, such that the maximum radius of the new set of clusters is minimized. The centers of the two partitions are chosen to be the leaders of the new clusters and transfers leadership to the new leaders. If these new clusters still violate the size upper bound, they are split by the new leaders using identical operations.

To maintain the tree structure and detect any unexpected partitioning in the tree, each member of a cluster sends periodic HeartBeat message to each of its cluster members. The message contains the distance estimate from the sender to each other member of the cluster. The cluster leader includes the complete updated cluster membership in its HeartBeat messages to all other members. This allows existing members to set up appropriate peer relationships with new cluster members on the control path. The cluster leaders also periodically send their higher layer cluster membership their cluster.

LA-NICE takes the link load into account when maintaining the tree. So instead of only checking if the cluster leaders need to be modified based on the leader's position with respect to the cluster members, LA-NICE checks the bandwidth of the closest 3 nodes to the center of the cluster (given that the cluster has more than 3 members) and assigns the one with the highest ratio of bandwidth/number of cluster members to be the cluster leader. This modification is selecting the cluster leader based on the fact that there is always higher load on the cluster leaders than the other nodes in the cluster since the cluster leaders send the multicast messages to all the cluster members leading to a bottleneck of O(k log N). So, ensuring that the leader has a relatively high bandwidth in addition to being close to all the members reduces the delay and improves multicast message deliver

III. Results and Analysis

A. Experiment Setup

To evaluate LA-NICE, we compared it to NICE protocol and Scribe[7] which is another ALM protocol. The evaluation was done using four different test groups of users. Each group had different number of users. The number of users in those groups were 20, 45, 70, and 100 respectively. The simulation was done using OMNeT++ as shown in figure 3 and it runs for 300 seconds where nodes exchange multicast messages where some users would drop out while other join the group randomly.

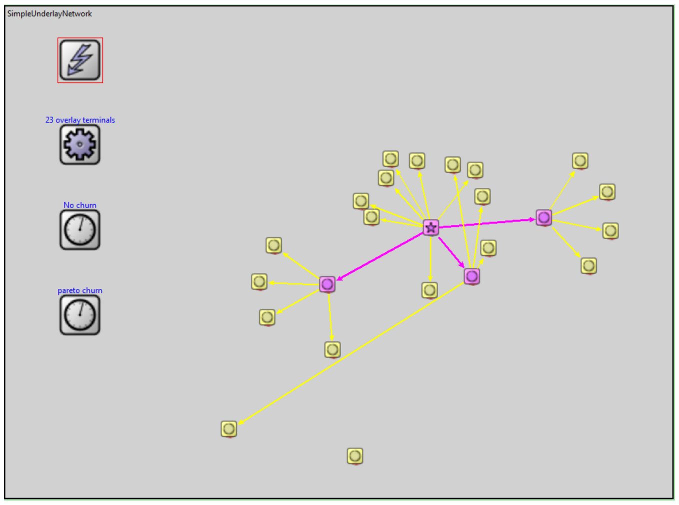


Fig. 3: LA-NICE multicast tree simulation using OMNeT++

B. OMNeT++

OMNeT++ [8] is an object-oriented open-source network simulator that is highly flexible and easy to use. The model's structure is described in OMNeT++ NED language. The Network Description (NED) language facilitates the modular description of a network, which consists of a number of component descriptions (channels, simple/compound module types, gates, etc.). In addition, OMNeT++ provides various statistic collections and visualization tools for results analysis. The simulator supports parallel and distributed simulation with the multiple instances communicating via Message Passing Interface(MPI), as well as support for network emulation through interfaces with real networks and the ability to use real networking code inside the simulator. OMNeT++ simulation models are composed of modules and connections. Connections may have associated channel objects. Channel objects encapsulate channel behavior: propagation and transmission time modeling, error modeling, and possibly others. Channels are also programmable in C++ by the user. Modules and channels are called components. Components are represented with the C++ class cComponent.

C. Implementation

The implementation was done using OMNeT++ framework with oversim [9]. Oversim includes a basic implementation of NICE protocol. They were both used in evaluating LA-NICE performance. The implementation of LA-NICE was built using OMNeT++ oversim framework. In LA-NICE, both the member join and the maintenance methods were modified. The code is implemented in C++ to write the logic of the protocol and OMNeT's .Net language to represent the user interface of the network. Datarate channels were used with different datarates to test various network conditions and bandwidth variations.

D. Results

Message Delivery*:* The first evaluation criteria for LA-NICE is the percentage of multicast messages delivered, failure in delivery indicates inefficient tree maintenance due to not detecting node departures in a reasonable amount of time or bad bandwidth utilization, which results in bottlenecks that lead to late delivery and sometimes failure in delivery. As seen in table I and figure 4, scribe has lower message delivery percentage. The reason behind the less performance in Scribe is due to the fact that the nodes are assigned random generated IDs. This randomness could lead to a situation where two hosts can be close to each other and yet sending a message from one of them to the other passes by other nodes that are far from them which takes longer time than it should. This is due to the routing table which sends messages to hosts that have the same prefix in their ID (which doesn't always mean that this is the closest node in the table). In addition, messages have a certain time to live (TTL) and then they timeout, so as the delay increases the amount of messages that timeout increase. NICE, on the other hand doesn't have this problem and it sends the messages to the closest nodes graphically without any regard to bandwidth conditions. LA-NICE combines both distance and bandwidth factors.

TABLE I: MESSAGE DELIVERY PERCENTAGE RESULTS

Number of users	NICE	Scribe	LA-NICE		
				% Improvement	
				over NICE	over Scribe
20	90.93	88.57	99.34	8.4%	10.7%
45	96.01	89.65	98.69	2.7%	9.04%
70	90.48	88.71	96.53	6.05%	7.8%
100	93.76	89.62	95.68	1.92%	6.06%

On the other hand, NICE takes the node's proximity to each other into account when constructing the delivery tree. for example, if two nodes are close to each other, the time taken to send exchange messages between them should be less than the time taken to exchange message between nodes that are further away from each other. Our main focus is to build on NICE and enhance its performance by making it link-aware. Link-Aware NICE handles the proximity factor in NICE in addition to the bandwidth factor when a new node joins the tree and when maintaining the tree as well. This is done through measuring the cluster leaders' bandwidth and selecting the cluster with the highest leader bandwidth that is relatively close. This approach produced the highest message delivery percentage as seen in figure 4 compared to NICE and Scribe.

Message Delay and Hop count: Another evaluation criteria is the average hop count of the multicast messages. The hop count of a packet is defined as the number of routers traversed by a packet between its source and destination [10], which is in this case the number of hosts that a message passes from the source to the destination [11]. As seen in figures 5 and 6 and tables II and III when the number of users is small, the average hop count of LA-NICE and NICE, as well as the maximum hop count is almost the same, however as the number of users increase LA-NICE has less number of hops leading to less delay in delivering the messages.

The delay factor is related to the hop count. Figures 7 and 8 and tables IV and V show a comparison between LA-NICE, NICE in terms of delay. The delay and message hops are usually proportional, therefore, with the increase of the hops along the tree, the delay increases. It is clear that LA-NICE outperforms NICE in terms of delay and hop count after taking the links' load factor into account.

TABLE II: AVERAGE HOP COUNT RESULTS

Number of Users	LA-NICE	NICE
20	1.55	1.48
45	1.73	1.77
70	1.79	2.1
100	1.78	2

TABLE III: MAXIMUM HOP COUNT RESULTS

Number of Users	LA-NICE	NICE
20	2	2
45	4	4
70	4	7
100	3	6

TABLE IV: AVERAGE DELAY RESULTS (S)

Number of Users	LA-NICE	NICE
20	0.923	0.91
45	0.83	0.9
70	1.54	1.58
100	1.21	1.19

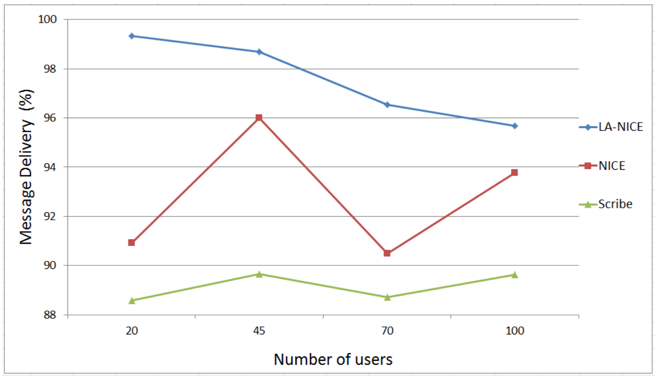


Fig. 4: Percentage of Message Delivery of LA-NICE, NICE and Scribe against different number of users

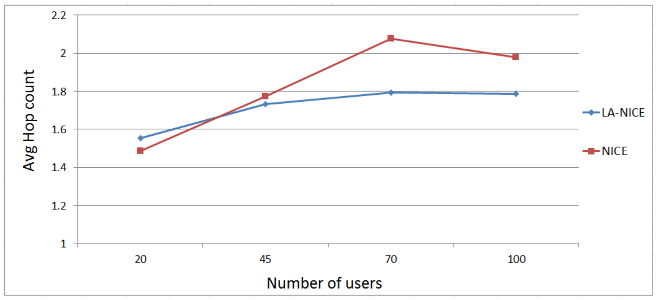


Fig. 5: Average hop count in LA-NICE and NICE against different number of users

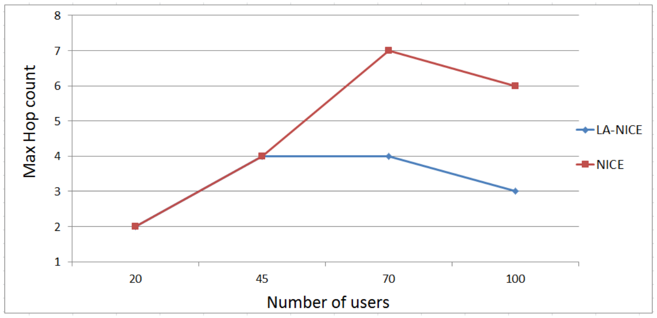


Fig. 6: Maximum hop count in LA-NICE and NICE against different number of users

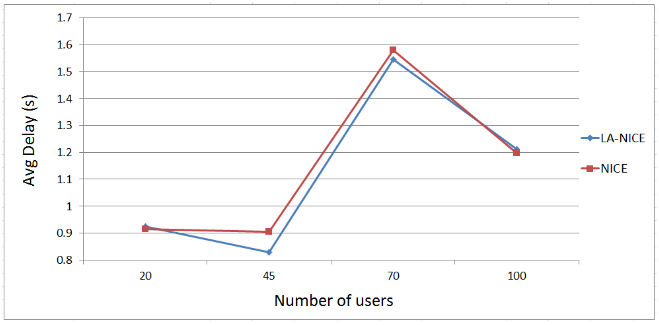


Fig. 7: Average Delay in LA-NICE and NICE against different number of users

TABLE V: MAXIMUM DELAY RESULTS (S)

Number of Users	LA-NICE	NICE
20	1.67	1.72
45	1.76	1.57
70	2.48	2.48
100	2.01	2.05

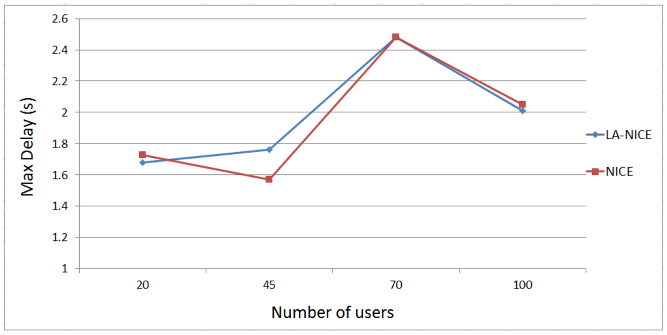


Fig. 8: Maximum Delay in LA-NICE and NICE against different number of users

IV. Conclusion

We presented LA-NICE which is an enhancement version of the ALM based NICE protocol. The added enhancement resulted in improved multicast message delivery and lower end-to-end delay. LA-NICE takes into account the fact that different links can have different bandwidths. The original member join, member leave and the maintenance functions in NICE algorithm were modified to include the link information. OMNeT++ simulator was used to evaluate LA-NICE and compare it against NICE as well as Scribe protocols. Simulation results showed that LA-NICE produced higher percentage of successful message delivery and less delays in data forwarding multicast messages compared to NICE and Scribe protocols. The study of the behavior of the three algorithms has shown that as the number of users increased and the network became more congested, the successful message delivery decreased and the delay increased.

References

[1] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," vol. 32, no. 4. New York, NY, USA: ACM, Aug. 2002, pp. 205–217.

[2] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, ser. Simutools '08. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 60:1–60:10.

[3] K. Ke and C. Huang, "Performance evaluation of multisource application layer multicast (alm): Theoretical and simulative aspects," vol. 57, no. 6. New York, NY, USA: Elsevier North-Holland, Inc., Apr. 2013, pp. 1408–1424.

[4] L. Lao, J.-H. Cui, M. Gerla, and D. Maggiorini, "A comparative study of multicast protocols: top, bottom, or in the middle?" in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 4, 2005, pp. 2809–2814 vol. 4.

[5] X. Li, X. Zhang, W. Luo, and B. Yan, "A clustering scheme in application layer multicast." vol. 30, no. 2, 2011, pp. 335–355.

[6] W. Xijuan, J. Ruisheng, L. Guang, and Y. Xianghong, "Research on p2p-based application layer multicast technology for streaming media," vol. 1. Los Alamitos, CA, USA: IEEE Computer Society, 2010, pp. 341–345.

[7] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "Scribe: A large-scale and decentralized application-level multicast infrastructure," vol. 20, 2002, pp. 1489 – 1499.

[8] A. Varga, "Omnet++ discrete event simulation system user manual," 2011.

[9] I. Baumgart, B. Heep, and S. Krause, "Oversim: A scalable and flexible overlay framework for simulation and real network applications," in *Peer-to-Peer Computing, 2009. P2P '09. IEEE Ninth International Conference on*, 2009, pp. 87–88.

[10] C. Hübsch, C. P. Mayer, and O. P. Waldhorst, "User-perceived performance of the nice application layer multicast protocol in large and highly dynamic groups," in *Proceedings of the 15th international GI/ITG conference on Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 62–77.

[11] S. Rizvi, M. Khan, and A. Riasat, "Deterministic formulization of bandwidth efficiency for multicast systems," in *Computer, Control and Communication, 2009. IC4 2009. 2nd International Conference on*, 2009, pp. 1–6.

Challenges when Realizing a Fully Distributed Internet-of-Things – How we Created the SensibleThings Platform

Stefan Forsström, Victor Kardeby, Patrik Österberg, and Ulf Jennehag
Department of Information and Communication Systems
Mid Sweden University
Sundsvall, Sweden
Email: {stefan.forsstrom, victor.kardeby, patrik.osterberg, ulf.jennehag}@miun.se

***Abstract*—The SensibleThings platform is an open source architecture for enabling Internet-of-Things based applications. During its development, multiple problems have been faced and solved, for example issues related to networking, information dissemination, sensors, and application access. This paper describes these problems and the technical solutions that are implemented in the platform. We also present the current progress and a series of demonstrator applications, which show the wide range of possibilities enabled by the platform. Finally, we present future work and how it will be used in future research endeavors and commercial interests.**

Keywords-overlay;sensors;actuators;internet-of-things.

I. Introduction

Applications that utilize information from sensors attached to different things in order to provide more personalized, automatized, or even intelligent behavior are commonly referred to as Internet-of-Things (IoT) applications [1] or Machine-to-Machine (M2M) applications [2]. The prediction is that these kinds of applications will be able to interact with an IoT, a worldwide network of interconnected everyday objects, and thereby be able to display context-aware behavior [3]. These applications may address a variety of areas, such as environmental monitoring (pollution, earth quake, flooding, forest fire), energy conservation (optimization), security (traffic, fire, surveillance), safety (health care, elderly care), and enhancement of social experiences. IoT applications will probably have a big impact on how we interact with people, things, and the entire world in the future.

There is also an interesting relationship between the IoT and big data [4], since all of the connected things will produce and consume large amounts of data. Current estimations are in the order of 50 billion connected devices year 2020 [5]. In order to enable a widespread proliferation of IoT services there must be a common platform for dissemination of sensor and actuator information on a global scale. This is however a very difficult goal to achieve, because there is a large number of practical difficulties that must be solved. We state that applications on the IoT require the following from an underlying platform.

1) The platform must be able to quickly disseminate information to end points. The communication should be done with low overhead and there should be no unnecessary proxying of data. This due to that there exists many scenarios with real-time constraints for IoT applications.
2) The platform must be stable and handle devices joining and leaving the system with high churn rates. There should be no central points of failure and it should be possible for the system to heal itself, even when a large number of devices leave at the same time. For example in IoT scenarios with high mobility.
3) The platform must be lightweight enough to run on devices with limited hardware resources, such as mobile devices, small computers, and sensor motes. Hence, computational heavy algorithms and large amounts of data storage is not possible in such end devices. It is reasonable to expect that the IoT will include a wide range of different devices with varying computational and storage resources.
4) The platform must be extensive and adaptive to conform with a wide range of possible applications and devices. Applications should be able to create new functionality on top of the platform, suiting their own needs. This due to that IoT applications spreads across a wide range of scenarios and there might be unknown possible future scenarios.
5) The platform must be easy to adopt and free to use in commercial products. Since the goal is global proliferation, there should be no restrictions in terms of software licenses and fees related to the code for those companies or enterprises wishing to utilize the platform. Also, IoT applications must be viable for pure commercial interests, not just as research implementations.

This paper addresses the practical difficulties of facing the above mentioned requirements. We first present a short survey of related work in terms of the different IoT platforms currently available, including their relation to the requirements. We then present our solution called the SensibleThings platform, describing how it is designed and implemented to meet all of the requirements in a satisfactory manner.

The paper is outlined as follow. Section II presents the survey over currently available platforms. Section III describes the SensibleThings platform, whereas Section IV focuses on the problems faced during the processes. Section V discusses current results and possible applications. Finally, Section VI presents the conclusion and future research.

II. Related Work

There currently exists a vast amount of platforms which claims to enable an IoT, far more than can be listed in this paper. However, they can generally be categorized into three categories, centralized (or cloud distributed), semi-distributed, and fully distributed systems.

A. Centralized Systems

Most of the systems being released today seem to focus on distributing the data on some form of cloud-based IoT architecture. The cloud is a concept that rise in popularity, but it is in many cases simply a new word for traditional web services. Very few of the cloud based systems explain how the distribution and synchronization is actually done inside their architecture, how many servers they use, etc. Either way, cloud-based systems can be considered as centralized systems since they always relay the sensor and actuator information through a centralized point, in this case a cloud (be it one or many connected servers). The main problems with cloud oriented solutions are that they have difficulties achieving requirement 1 on direct communication between end devices, requirement 2 on no central points of failure, and requirement 5 on an open and free to use system, because they are based on large scale servers. Typical examples of these cloud based architectures include: SicsthSense [6], ThingSpeak [7], Sen.Se [8], Nimbits [9], ThingSquare [10], EVRYTHNG [11], Paraimpu [12], Xively [13], XOBXOB [14], Thingworx [15], One Platform [16], Carriots [17], and many more.

B. Semi-Distributed Systems

The semi-distributed systems are often based on session initiation protocols, whereas they afterward use direct communication between the connected devices. Because of this, they usually contain a centralized point for coordinating the communication. Thus, semi-distributed systems are faster and to some extent easier to scale than centralized solutions, but they still have difficulties coping with requirement 2 and 5. Typical examples of these semi-distributed architectures include: ETSI M2M [18], SENSEI [19], ADAMANTIUM [20], and other platforms based on 3GPP IMS [21].

C. Fully Distributed Systems

Fully distributed systems operate in a peer-to-peer manner, where they both store and administer the information locally on each entity. To achieve this, they often utilize hash tables to enable logarithmic scaling when the number of entities increases in magnitude. These systems do not contain any single point of failure and are thus more resilient, though the distribution itself often requires additional overhead in order to maintain an overlay. The main problem associated with fully distributed systems is however that they place a larger responsibility on the end devices, and thus have difficult to achieve requirement 3. Examples of such systems are SOFIA [22], COSMOS [23], and MediaSense [24].

III. The SensibleThings Platform

In order to solve the problem and address the stated requirements we have created the SensibleThings platform, which is a realization and implementation of the MediaSense architecture explained in [24]. The SensibleThings platform can be seen in figure 1, which presents the different layers and components of the platform. These include an interface layer, an add-in layer, a dissemination layer, a networking layer, and a sensor/actuator layer. The layers are explained in detail in the original article, but they will be summarized here as well. The actual SensibleThings code is based on a fork of the MediaSense platform, but has been significantly improved since then. The focus has been on the open source aspect and maintaining the commercialization possibilities of applications that are utilizing the platform. Other differences include the actual implementation of the lookup architecture, communication protocol, and code interfaces.

A. Interface Layer

The interface layer is the public interface through which applications interact with the SensibleThings platform. The interface layer includes a single component, the SensibleThings application interface, which is a generic Application Programming Interface (API) for developers to build their own applications on top of.

B. Add-in Layer

The add-in layer enables developers to add optional functionality and optimization algorithms to the platform. Add-ins can for example help the platform meet specific application requirements, such as specifically demanded features or handle the available capacity in regards to computational power and bandwidth. The add-in layer manages different extensible and pluggable add-ins, which can be loaded and unloaded in runtime when needed. These add-ins are divided into optimization and extension components, but the platform can include any number of them at the same time.

C. Dissemination Layer

The dissemination layer enables dissemination of information between all entities that participate in the system and are connected to the platform. A variant of the Distributed Context eXchange Protocol (DCXP) is used, which offers communication among entities that have joined a peer-to-peer network, enabling exchange of context or sensor information in real-time. The operation of the DCXP includes resolving of so called Universal Context Identifiers (UCI) and subsequently transferring context information directly. Therefore, the dissemination layer includes three components, a dissemination core, a lookup service, and a communication system. The dissemination core exposes the primitive functions provided by DCXP, the lookup service stores and resolves UCIs within the system, and the communication component abstracts transport layer communication. In short, the dissemination layer enables registration of sensors in the platform, resolving the location of a sensor in order to find it, and the communication to retrieve the actual sensor values.

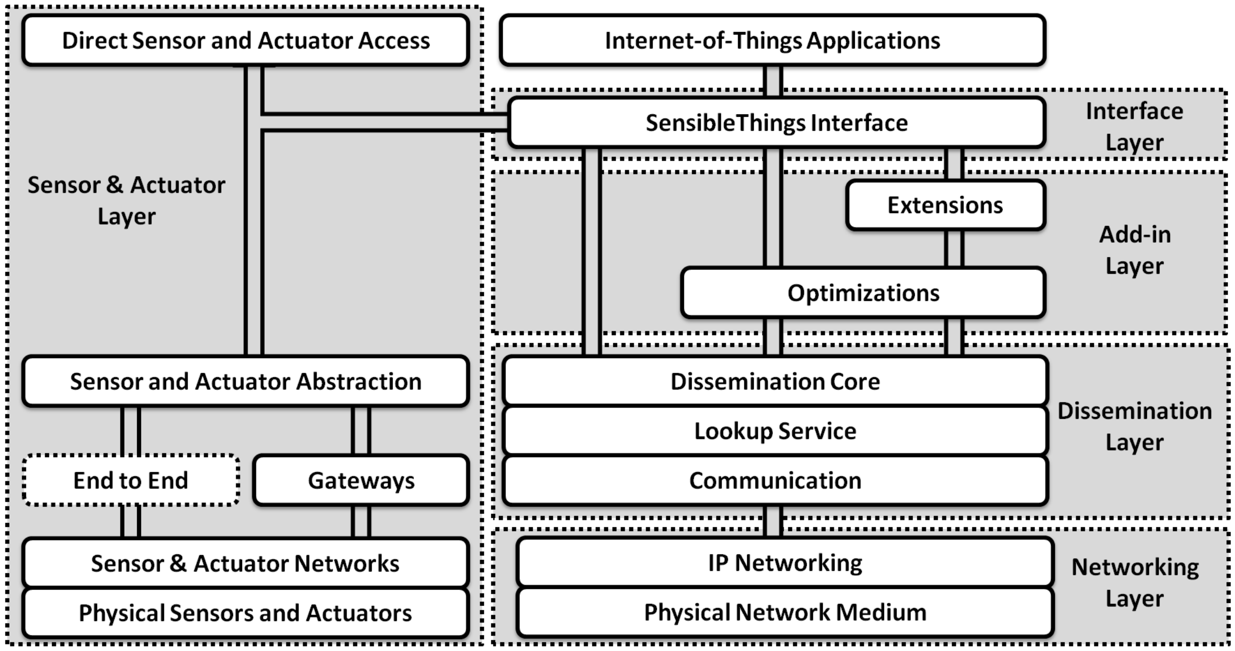


Figure 1. Overview of the SensibleThings platform's architecture

D. Networking Layer

The networking layer enables connection of different entities over current Internet Protocol (IP) based infrastructure, such as fiber optic networks or wireless and mobile networks. Hence, the networking layer is separated into two inner components, an IP network and the physical network medium. In short, the networking layer thus abstracts any underlying IP-based network architecture.

E. Sensor and Actuator Layer

The sensor and actuator layer enables different sensors and actuators to connect into the platform. The sensors and actuators can vary greatly and the platform therefore offers two options to connect them. Firstly, they can be connected directly if they are accessible from the application code, such as in the case of smartphone sensors. Secondly, the sensors and actuators can connect through the sensor and actuator abstraction. The abstraction enables connectivity either directly to wireless sensor networks or via more powerful gateways. Hence, the sensor and actuator layer is separated into five components: the directly accessible sensors and actuators, an abstraction component, different sensor and actuator networks, sensor and actuator gateways, and the physical sensors and actuators.

IV. Encountered Problems

This section outlines the different problems encountered during the development of the SensibleThings platform. The problems are divided according to what layer they belong to and explained in the following subsections. The last subsection describes issues related to the source code licensing.

A. Interface Layer Problems

The main problems of the interface layer were related to requirement 5 on easy usage, how to make the platform easy to understand and easy to implement. Different approaches were explored, but since almost all communication on the platform is done asynchronously, the listener java pattern is typically used in the application interface. Hence, almost all interface access with the platform is done through normal function calls, whereas the values are returned in event listeners.

B. Add-in Layer Problems

The add-in layer have also posed some specific problems, most prominently how the add-ins should be managed, loaded, and the API's chain of command. This relates to requirement 4 on extensibility and in requirement 5 on easy usage. There are also decisions to be made on what parts of the platform's API that should be considered primitive actions or be provided as add-in features. In the current platform, there are still some limitations as these issues have not been prioritized. For example, the add-ins have no chain of command and will therefore hijack functionality of the platform when enabled. Thus, some add-ins become mutually exclusive and will not function properly together.

C. Dissemination Layer Problems

The main problem faced when developing the dissemination layer was regarding the choice of lookup service that supports requirement 1, 2, and 3, namely quick dissemination, good scalability, and lightweight operation. There exists a number of Distributed Hash Tables (DHT) that the platform could use, where the most prominent are Chord [25], Kelips [26], and P-Grid [27]. All three choices have their separate advantages and disadvantages. Chord uses a ring structure which is difficult to maintain and has a logarithmic lookup time $O(log(N))$. Kelips uses affinity groups with a much simpler synchronization scheme and has fixed lookup time of $O(1)$, but it does not scale as well and has a larger overhead. P-Grid has a trie based structure with a logarithmic lookup time of $O(0.5\,log(N))$, but is more complex and difficult to maintain.

In the current platform, both Chord and Kelips are completely reimplemented to operate within the platform and with the same license as the rest of the code. We have also experimented with the currently available P-Grid code, but since that is using multiple source code licenses (including propagating open source licenses), it cannot be a part of the SensibleThings code at this stage. Currently the platform defaults to the Kelips DHT implementation, simply because that code is more stable than the Chord implementation when nodes join and leave rapidly.

The second problem faced in the dissemination layer was the choice of communication protocol. Requirement 1 states that the communication should be fast with low overhead. Therefore, the aim was to have the useful payload data already in the first packet. Because of this, a variant of a Reliable User Datagram Protocol (RUDP) is utilized as the default protocol. The problem with RUDP is however that the packets are sent in clear text, but to support industry applications the platform must provide the possibility of encryption. There exists several approaches for enabling this, such as different key exchange schemes with varying degrees of security and overhead. In the end, the decision was to support standard Secure Sockets Layer (SSL) encryption to at least make it possible to encrypt the data if needed. The encryption is however only useful to prevent eavesdropping, not man in the middle attacks, because all certificates will be self signed by the end devices. There is also a significant overhead related to SSL, and since there is an initial handshake the data will not come in the first packets.

There have also occurred different problems in relation to the serialization of messages, how the messages are coded when sent over the Internet (before any encryption). A binary serialization format is most suitable from a performance perspective, but a text based format is most usable from a human-readable perspective and a code-specific format is most easy to program. In the end, the choice was to support all different serialization formats but the default is set to Java's object serialization, to make it easier to develop new extensions. Likely, the Java serializer will be replaced in the future, in order to make transitions to other platforms and programming languages feasible.

D. Networking Layer Problems

The major problem faced in the networking layer was related to requirement 2 on stability and seamless communication. The first versions of the platform did not take Network Address Translation (NAT) and firewalls into consideration, it only worked if all devices was on the public Internet. However, today almost all consumer devices are connected to the Internet through either NAT or some type of firewall, either in their home or at their work, but also on the mobile phone networks. The NAT and firewall problem is only a question of configuration, given that the user is allowed to enable features such as port forwarding on the NAT routers, but that this rarely the case. For example, most normal users simply want things to work out of the box, most companies do not allow their employees to enable such features on their company network, and many Internet service providers are now enabling carrier grade NAT [28].

Multiple approaches were considered, ranging from IP version 6 (IPv6) solutions, to Universal Plug and Play (UPNP), different hole punching techniques, and finally simple proxy solutions. The chosen solution first tries the normal approaches, such as direct connections and UPNP. If this fails it instead utilizes distributed proxy nodes in the system. As the proxy solution stands in direct contradiction to requirement 1 on real-time communication without unnecessary relaying of information and requirement 2 on no central points of failure, it is only used as a last resort when there are no other solutions available.

There also exists problems related to the capacity of the Internet connections and their network delay, but since the SensibleThings platform is built on top of the existing Internet architecture, it cannot affect these parameters. Therefore, as long as useful payload data is sent in the first packet, the assumption is that it is being transmitted as fast as possible by the underlying network infrastructure.

E. Sensor and Actuator Layer Problems

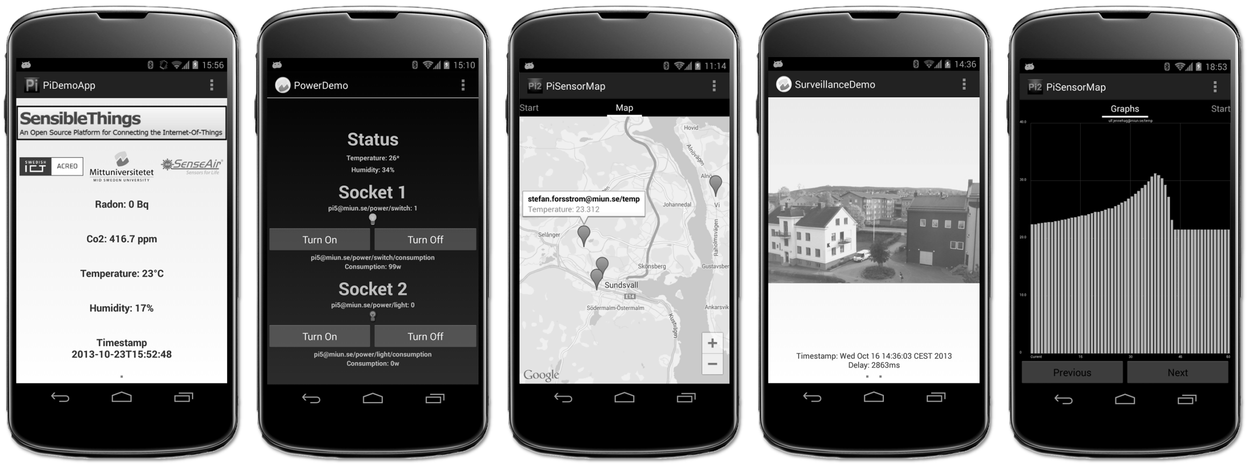
In the sensor and actuator layer, there were problems with the actual sensor hardware platforms that is available today, especially in regards to requirement 3 on being lightweight. Different vendors of sensors have different platforms that the sensors run on, especially when it comes to connecting large Wireless Sensor Networks (WSN). Typically, cheap analog sensors can be connected directly to a more powerful device, such as a smartphone or a Raspberry Pi [29]. But to connect traditional WSN architectures such as TinyOS [30] or Contiki [31], the platform must communicate via COAP [32] or other lightweight protocols that they can handle. Therefore, in most of the examples we have utilized either smartphones with sensors already built in, or Raspberry Pi devices with attached sensors. However, any device that can run the Java code for the platform can be a part of the system, and any low end device that can communicate via COAP can easily be connected via a more capable device.

F. Source Code License Problems

One purpose of the platform is to make it available for industry partners to develop their own applications and then commercialize the products, see requirement 5. This requirement made it impossible to use a strict and propagating open source license such as GNU General Public License (GPL). The amount of external code should also be kept to a minimum, in order to maintain the control over all the licenses in use. In the end, the decision was to use the GNU Lesser General Public License (LGPL) that allows companies to make commercial products on top of the platform, without forcing their products to be open source as well.

V. Results and Applications

The current results include launching our new development website for the SensibleThings platform [33]. This website will act as a portal for all developers who want to utilize the platform in their applications. The SensibleThings platform is provided free and under an LGPL version 3 open source license. Initial testing, demonstration, and evaluation of the platform has been conducted using a testbed with fixed and



(a) Sensor reading (b) Intelligent home (c) Object tracking (d) Surveillance (e) Historical values

Figure 2. Examples of applications using the SensibleThings platform.

mobile access to the Internet. In terms of performance we have measured the platform to be on par with UDP traffic. Table I shows the response times measured for resolving, and retrieving a specific sensor value in the platform. The table shows both the arithmetic mean μ and the standard deviation σ. The measurements were conducted with 103 devices connected to the platform. One workstation, one Raspberry Pi, and 100 emulated devices connected via 1 Gbit fiber optic based Internet connection. Lastly, one mobile device connected via 3G mobile Internet connection. The measurements were conducted on the workstation, the Raspberry Pi, and the mobile device, to show the difference between them and the effect of the proxy solution for NAT problems associated with the mobile device. The other 100 devices were put into the system to create background traffic, in order to emulate a real-world scenario with many connected devices.

Proof-of-concept demonstrator applications have been built using many different devices, sensors, and actuators, in order to show the versatility of the SensibleThings platform. The applications presented in figure 2 show some of these demonstrators. From left to right they are: (a) sensor value readings (radon, carbon dioxide, temperature, and humidity), (b) home automation with energy consumption measuring (for interacting with an intelligent home), (c) object tracking (for tracking different objects with attached sensors), (d) surveillance (for detecting trespassers), and (e) a set of historical measurements (for statistical usage). But the platform itself is versatile enough to be applied to even more areas. We foresee possible applications ranging from intelligent home, healthcare, logistics, emergency response, tourism, and smart-grids, to more social-oriented applications such as crowd sourcing, dating services, and intelligent collaborative reasoning.

VI. Conclusion and Future Work

In this paper we presented the challenges we have encountered when researching and developing the SensibleThings platform. The first contribution of this paper is the identification of the requirements for a functional and fully distributed IoT platform in Section I. The second contribution is a short survey of existing IoT platforms, presented in Section II.

TABLE I
Measured response times for platform.

		Workstation	Raspberry Pi	Mobile
Resolve	μ	4.8 ms	31 ms	230 ms
	σ	2.2 ms	14ms	68 ms
Retrieve	μ	4.8 ms	31 ms	280 ms
	σ	2.0 ms	12 ms	56 ms

The main contribution is however the explanation of the problems faced when building the SensibleThings platform, and the solutions that solve these problems. The proposed SensibleThings platform is shown to fulfill all the requirements stated in Section I. The platform can disseminate information to end devices quickly with low overhead (requirement 1). It is stable and operates without any central points of failure (requirement 2). The platform is lightweight and can run on mobile devices where the only central point is the bootstrap device (requirement 3), but any node can act as a bootstrap if this original node goes offline. It is extensible (requirement 4) as shown with the demonstrator applications. Finally, the platform is licensed under a well known and widely accepted open source license making it free to use and at the same time encouraging development of commercial a products (requirement 5). In comparison to related work, the SensibleThings platform can be classified as a fully distributed solution, where the overhead and communication is kept as lightweight as possible. We predict that this is the only type of solution that will scale for billions of connected devices and still be able to disseminate sensor information with real-time demands.

Our current efforts are directed toward improving and optimizing the existing code, as well as investigating other possible choices of DHT and communication protocol. We will also develop more extensions to satisfy specific applications demands, such as seamless integration with other IoT platforms and cloud infrastructures.

Acknowledgment

This research has been supported by grant 2010-00681 of VINNOVA the Swedish Governmental Agency for Innovation

Systems, and by grant 00163383 of the EU European Regional Development Fund, Mellersta Norrland, Sweden. The authors also want to thank our partners from industry and academia, in particular Acreo AB and Stockholm University.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," Computer Networks, vol. 54, no. 15, 2010, pp. 2787–2805.

[2] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. Johnson, "M2M: From mobile to embedded Internet," Communications Magazine, IEEE, vol. 49, no. 4, 2011, pp. 36–43.

[3] J. Hong, E. Suh, and S. Kim, "Context-aware systems: A literature review and classification," Expert Systems with Applications, vol. 36, no. 4, 2009, pp. 8509–8522.

[4] D. E. O'Leary, "Big data, the internet of things and the internet of signs," Intelligent Systems in Accounting, Finance and Management, vol. 20, no. 1, 2013, pp. 53–65.

[5] Ericsson. More than 50 billion connected devices. White Paper. [Online]. Available: http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf [retrieved: December, 2013]

[6] SicsthSense. [Online]. Available: http://sense.sics.se [retrieved: December, 2013]

[7] ThingSpeak. [Online]. Available: https://www.thingspeak.com [retrieved: December, 2013]

[8] Sen.se. [Online]. Available: http://open.sen.se [retrieved: December, 2013]

[9] Nimbits. [Online]. Available: http://www.nimbits.com [retrieved: December, 2013]

[10] Thingsquare. [Online]. Available: http://thingsquare.com [retrieved: December, 2013]

[11] EVRYTHNG. [Online]. Available: http://www.evrythng.com [retrieved: December, 2013]

[12] Paraimpu. [Online]. Available: http://paraimpu.crs4.it [retrieved: December, 2013]

[13] Xively. [Online]. Available: https://xively.com [retrieved: December, 2013]

[14] XOBXOB. [Online]. Available: http://www.xobxob.com [retrieved: December, 2013]

[15] ThingWorx. [Online]. Available: http://www.thingworx.com [retrieved: December, 2013]

[16] One platform. [Online]. Available: http://exosite.com/products/onep [retrieved: December, 2013]

[17] Carriots. [Online]. Available: https://www.carriots.com [retrieved: December, 2013]

[18] ETSI, "Machine-to-machine communications (m2m); functional architecture," (TS 102 690 V1.1.1), Tech. Rep., 2011.

[19] M. Presser, P. Barnaghi, M. Eurich, and C. Villalonga, "The SENSEI project: integrating the physical world with the digital world of the network of the future," Communications Magazine, IEEE, vol. 47, no. 4, 2009, pp. 1–4.

[20] H. Koumaras, D. Negrou, F. Liberal, J. Arauz, and A. Kourtis, "ADAMANTIUM project: Enhancing IMS with a PQoS-aware multimedia content management system," International Conference on Automation, Quality and Testing, Robotics, vol. 1, 2008, pp. 358–363.

[21] G. Camarillo and M.-A. Garcia-Martin, The 3G IP multimedia subsystem (IMS): merging the Internet and the cellular worlds. Wiley, 2007.

[22] A. Toninelli, S. Pantsar-Syväniemi, P. Bellavista, and E. Ovaska, "Supporting context awareness in smart environments: a scalable approach to information interoperability," in Proceedings of the International Workshop on Middleware for Pervasive Mobile and Embedded Computing. ACM, 2009, pp. 1–4.

[23] P. Bellavista, R. Montanari, and D. Tibaldi, "Cosmos: A Context-Centric Access Control Middleware for Mobile Environments," in Mobile Agents for Telecommunication Applications, 2003, pp. 77–88.

[24] T. Kanter, S. Forsström, V. Kardeby, J. Walters, U. Jennehag, and P. Österberg, "Mediasense–an internet of things platform for scalable and decentralized context sharing and control," in ICDT 2012, The Seventh International Conference on Digital Telecommunications, 2012, pp. 27–32.

[25] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in in Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, vol. 31. New Yourk, NY, USA: ACM Press, 2001, pp. 149–160.

[26] I. Gupta, K. Birman, P. Linga, A. Demers, and R. Van Renesse, "Kelips: Building an efficient and stable p2p dht through increased memory and background overhead," in Peer-to-Peer Systems II. Springer, 2003, pp. 160–169.

[27] K. Aberer, "P-grid: A self-organizing access structure for p2p information systems," in Cooperative Information Systems. Springer, 2001, pp. 179–194.

[28] S. Jiang, D. Guo, and B. Carpenter, "An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition," RFC 6264, Tech. Rep., 2011.

[29] Raspberry pi. [Online]. Available: http://www.raspberrypi.org [retrieved: December, 2013]

[30] P. Levis et al., "TinyOS: An operating system for sensor networks," in Ambient intelligence. Springer, 2005, pp. 115–148.

[31] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in Local Computer Networks, 2004. 29th Annual IEEE International Conference on. IEEE, 2004, pp. 455–462.

[32] Z. Shelby, K. Hartke, and C. Bormann. Constrained application protocol (COAP). [Online]. Available: http://tools.ietf.org/html/ietf-core-coap-14.txt (2013)

[33] SensibleThings. [Online]. Available: http://www.sensiblethings.se [retrieved: December, 2013]