Extending Service-oriented Architectures to the Tactical Edge

G. Benincasa[1], E. Casini[1], R. Lenzi[1], A. Morelli[1], E. Benvegnù[1], N. Suri[1,2], K. Boner[3], S. Watson[3]

[1]Florida Institute for Human and Machine Cognition, Pensacola, FL USA
[2]U.S. Army Research Laboratory, Adelphi, MD USA
[3]Space and Naval Warfare Systems Command San Diego, CA USA

***Abstract*—Service Oriented Architectures allow for seamless integration of heterogeneous systems and extensive service reuse; characteristics that led to a wide adoption of this paradigm in the enterprise and military environment. While SoAs are currently deployed in tactical environments mainly at higher-echelon levels, it is necessary to allow for the exchange of information all the way down to the edge nodes deployed on the ground and back. Because most SoAs implementations were designed to work on reliable infrastructure networks, porting SoAs to the tactical environment requires a complete redesign of the protocol stack to support unreliable, transiently disconnected networks. In this paper, we present our approach to integrate the U.S. Marine Corps's Marine Command and Control Systems and Applications SoA (MC2SA SoA) with DisServicePro, a middleware that supports proactive dissemination and information on demand in tactical edge networks.**

***Index Terms*: Service-oriented Architectures, Tactical Networks, Information Dissemination, Quality of Information.**

I. Introduction

SERVICE-oriented Architecture (SoA) -based approaches lead to the development of independent services that are interoperable and that can be composed to implement complex distributed applications by defining a workflow. Integration of heterogeneous services and service reuse are valuable features that might allow for significant savings in terms of development time and total cost. These considerations are of particular relevance in the enterprise environment, which requires the development of large and robust applications. Service integration is of particular relevance in the context of military applications where, as stated in [1], coalition forces, or different branches of the same force, use different systems and sharing information across forces/branches is of paramount importance. Because of the benefits that SoA architectures entail, their adoption has been recommended for the development of military applications.

SoAs methodologies have been largely adopted to build web applications or to connect enterprise applications and therefore most of the current implementations were developed to work in the internet, or over enterprise private network, and therefore rely on protocols that require static and reliable network topologies. Tactical networks pose entirely new challenges that did not apply to the context for which SoAs were proposed. First and foremost, because tactical operations are highly dynamic, the nodes are interconnected via rapidly deployable wireless ad-hoc networks. These networks may be bandwidth constrained and unreliable and may often be partitioned. Moreover, the set of nodes operating in the network is highly heterogeneous: they may range from battery-operated, computationally constrained sensors that communicate with low-bandwidth, short range, power-efficient radio links, to rack mounted servers that can communicate over satellite links.

These challenges call for the design of new ways to support SoAs in the tactical environment. These new ways should not rely on the presence of infrastructure nor on the presence of end-to-end links from the source to the destination(s).

Decentralized and Peer-to-peer (P2P) architectures offer a viable solution to the problem of the lack of infrastructure. In P2P architectures nodes are not assigned pre-defined roles and each of them can provide service to other peers and consume services from other peers. This approach is more robust and can allow for partial functioning of the network even in case of partitioning.

Store-carry-and-forward networks have been proposed to deal with disconnected networks. The protocol stack on which tactical SoAs are developed should include protocols that were designed along these paradigms.

While previous research has focused mainly on the problem of service discovery, in this paper we focus on the exchange of information between higher echelon Command and Control networks and the edge nodes. In particular, we describe our effort to integrate the existing U.S. Marine Corps' Marine Command and Control Systems and Applications SoA (MC2SA SoA) with the Proactive Dissemination Service (DisServicePro) P2P middleware to support (a) command and control information delivery from the a command-center (for example, a Combat Operations Center or a COC) to the edge nodes, (b) P2P dissemination of situation awareness and sensor data among the edge nodes and (c) from the edge nodes back to the COC. The paper also presents other components that were necessary to the integration. In particular, the

Mockets communication library that offers efficient point-to-point communication over unreliable wireless links and the ACI NetProxy that passively captures the traffic to/from an application and redirects it to Mockets. The NetProxy allows the integration of Mockets with legacy applications that cannot be modified to use Mockets directly. Finally the COC–SoA Bridge interfaces MC2SA SoA to DisServicePro. It is worth noting that the integration did not require any change in the existing MC2SA SoA.

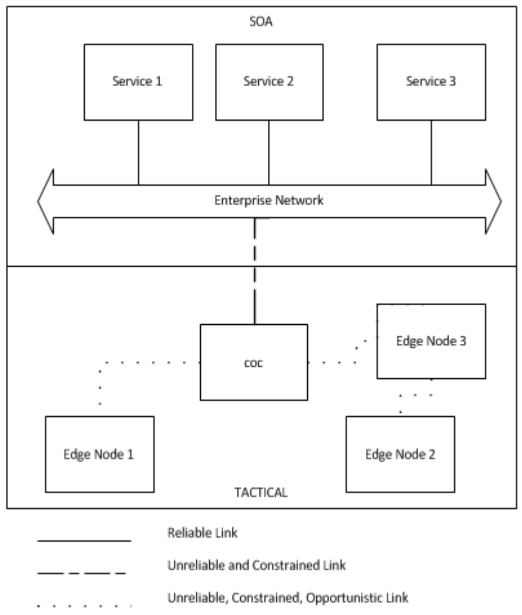
In the following sections we first provide an overview of current SoA implementations that have been proposed for tactical networks and a survey of SoA implementations over P2P systems. Secondly, we provide a system-level overview of our architecture and a description of its components, followed by a scenario that has been developed and that includes a quantitative analysis of the benefits introduced by DisServicePro. Finally, we then conclude and discuss future work.

II. Related Work

An exhaustive overview of the challenges in developing SoAs in tactical networks and their requirements can be found in [1]. For instance, traditional SoA approaches rely on centralized registries to discover the services that are available and to locate a node that offers the service, which introduces single points of failure un the network and poor scalability. Moreover, many implementations of SoA utilize verbose messaging systems (usually based on XML or, more recently, on JSON) that rely on protocols that were designed for wired, reliable networks.

In [2], the authors highlight that one of the challenges to support SoA in tactical networks is to "*enable users to exchange information with each other at all operational levels*". Edge nodes operate in constrained, low-bandwidth, and disconnected environments; in this context the client-server paradigm commonly adopted to implement SoAs may not be well suited. A more promising approach to connect to edge nodes is implementing SoAs on top of peer-to-peer (P2P) middleware. Because they do not rely on specific nodes to act as servers, P2P systems are more robust to network partitioning and can ensure partial functionality even if the network is partially disconnected. A study of the requirement and of the suitability of P2P systems for tactical network environments can be found in [3]. Approaches that implement SoAs on top of P2P networks have been proposed in [6-9, 10]. In [7] the authors propose p2pSOA, a P2P SoA middleware to support pervasive connectivity among personal devices. However p2pSOA relies on JXTA, which was not designed for mobile ad-hoc networks. In [8] an approach that uses distributed hash tables (DHTs) for registering the available service is presented. An approach that relies on DHTs will be inefficient at best on a MANET and would probably not work in case of network partitioning. An approach that does not rely on DHTs but on on-demand service discovery is presented in [6]. A comprehensive middleware that integrates software agents and SoAs is described in [4]. In [4], data is disseminated using the distributed, p2p implementation of XMPP described in [5].

Figure 1



Other approaches, like the ones in [9-10] advocate for the adoption cross-layer architectures. In particular, in [9] they propose a cross layer approach to integrate the service discovery with the routing algorithm (OLSR), while in [10], Facchini et. al. propose an approach were the cross layering includes the SoA layer. This latter approach allows greater integration among all the layers, and allows for greater control over the network. However adopting this approach may require changes to the SoA APIs, which may not always a feasible solution.

III. Tactical SoA Vision

The architecture depicted in Figure 1 illustrates the possible Tactical SoA (TSoA) system architecture. The SoI (Service-oriented Infrastructure) is running at the higher echelons, along with other services. These components are likely to be deployed at COCs with enterprise-class intranet works within the COC and therefore it is reasonable to assume they are connected through reliable links. At this level, client applications are directly connected to the SoI and can make information available to the other components and receive information from other components by publishing/subscribing to the SoI. At a lower architecture level, the COC Bridge is in charge of connecting the SoI to the edge nodes. The COC is expected to be deployed on the ground and be connected to the higher echelons, where the SoI is deployed, by means of point-to-point, unreliable, low-bandwidth links. The COC Bridge distributes messages from the SoI to the edge nodes and publishes the messages from the edge nodes to the SoI. In order to support the distribution of information such as situational awareness data, or command and control information, it is critical that the communication protocol between the COC and the edge nodes supports efficient point-to-multipoint transmission. The link from the COC to the edge nodes and between the edge nodes themselves may not always be available, or in some cases may even never be available, requiring that the communication between COC and edge node be performed by mean of mobile nodes (for instance UAVs) that ferry the messages between the two sides. It is therefore critical that the system is able to take advantage of the transient links and that the nodes in the edge network operate in a store-and-forward fashion. When links to the edge network are available, it is important to maximize the utility of the information being transmitted. Therefore, messages should be prioritized based on their relevance to the users and their missions. It is also of paramount importance that the edge nodes are able to share relevant information with each other, without the auxiliary support of explicitly designated servers and infrastructure at the COC.

IV. Tactical SOA Components

In this section, we present the components of our implementation for tactical SOA.

A. SoI

U.S. Marine Command and Control Systems and Applications (MC2SA) Service-oriented Architecture (SoA) was developed with the intent of governing service-oriented interactions between software elements on a wide range of tactical systems. These systems include legacy Tactical Data Systems (TDSs) and newly developed software services specifically designed to interact with the SoA. The SoA Architecture is composed of three parts: the Task Service Layer, the Entity Service Layer and the Utility Service Layer. The Utility Service layer is the layer that acts as a container for the Service-Oriented Infrastructure (SoI), one of the key components of the system architecture of this approach. The SoI provides the non-business-centric infrastructure that is the basis for all other services in the architecture of the SoA. The SoA and the set of services built on the SoI are deployed as capability modules, each of which consists of one or more virtual machines (VMs). The advantage of this approach is that virtualization enables deployment of the software in any environment that supports a virtual machine solution, assuming that adequate computing resources are available.

The SoI offers a Publish-Subscribe service that provides the capability to publish data and subscribe for data based on specified filter criteria. Published data is structured according to a specific Information Model that defines an Information Object type, which represents a specific data format and associated policy settings. This allows subscribers to filter on a known set of criteria regardless of data type and content. Also, the SoI provides a feature called Mediation that allows the system to perform transformations on Information Object payloads, which are exchanged via the SoI and/or stored in the SoI. This provides the functionality for mediation of data within the SoI like the handling and translation of XML payloads between Information Object Types. Other services provided by the SoI are the Information Repository that handles CRUD (Create, Read, Update and Delete) operations and the persistence policy configuration of the above mentioned. At last, the SoI provides also a Tactical Messaging feature that covers the handling of incoming Variable Message Format (VMF) and United States Message Text Format (USMTF) messages, which can be mediated and distributed by the SoI over disadvantaged networks through DisServicePro, via the SoA Bridge.

B. DisService

Dissemination Service (DisService) is a P2P publish/subscribe message-oriented middleware that was designed to perform efficient disruption-tolerant information dissemination in wireless tactical environments. Because DisService was specifically designed for wireless networks, it relies on the assumption that the cost of a local broadcast is equivalent to the cost of using unicast to a neighbor. DisService takes advantage of this assumption by always using local broadcast (or multicast) to send a message to a neighbor, which allows other neighbors to opportunistically listen to the communication that can therefore cache the message. The use of opportunistic listening increases the availability of the message and allows the recipients of the message to retrieve missing messaging or missing fragments of a message, from neighbors other than the sender. This capability has proven to significantly increase the reliability of the transmission in presence of unreliable and asymmetric links [13]. Because different applications may have different requirements, DisService supports all the combinations of reliable/unreliable and sequenced/non-sequenced message delivery. When reliable delivery is selected, DisService adopts a receiver-initiated reliability, which integrates well with the opportunistic listening feature (a node may decide to request a missing fragment to any peer). Moreover, DisService was designed to support point-to-multipoint communications and different peers may have different reliability requirements; using a receiver-initiated reliability mechanism allows each application to enforce the reliability model that best matches its needs.

New messages in DisService can be published in two different modes: they can be instantly pushed over the network, or they can be made available for other peers to retrieve. In the latter case, only a small metadata that describes the message and advertises its availability is sent over the network. Receiving peers can decide whether to retrieve the message or not. This feature is particularly useful when the message being published is large and should not be sent unless explicitly requested. When messages are made available, they can also be divided in chunks that are individually intelligible, but that may contain either incomplete or lower resolution information than the original message. A peer may subsequently request individual chunks until the reassembled information reaches a level of detail that satisfies the requirements of the application. This feature may allow for significant bandwidth savings when it is not necessary to retrieve a subset of the chunks. Moreover, even when the whole set of chunks has to be retrieved, partial information will be available before the complete message is retrieved.

DisService allows the registration of modules to control the replication and forwarding of messages and management of the data cache.

C. DisServicePro

The Proactive Dissemination Service (DisServicePro) [14] extends DisService and introduces new capabilities to support proactive information dissemination. DisServicePro attempts to predict what information will be useful and therefore likely to be requested by a neighbor and pre-stage it to the neighbor ahead of time when the link is still available. The prediction is performed by matching the *user context* of the neighbor against the *metadata* of the message.

The user context of the nodes includes information about the user such as the paths (one current path, and possibly multiple alternative or backup paths) that the user is expected to follow, the mission that he or she is performing, his or her role, etc. Moreover, the user can configure a value of *useful distance*, which determines the size of the area around the path for which the user is interested in receiving information.

The metadata describing the data contains a set of pre-defined attributes (such as time of creation, location, area of relevance, target mission, pedigree, etc.) but can be customized to match the needs of the application.

The matching is based upon a set of policies that take into account the spatial and temporal relevance of the message for the user (by matching the area of relevance of the metadata against the user's paths), the role of the user and the mission it is assigned, the expiration time of the message and the importance that the message was assigned by the user generating the message. Moreover, the prediction can be augmented by also taking into account the personal preferences of the user. These preferences are learned by the history of the previous messages that the user requested. The metadata of the messages that are considered useful for a node are then scheduled for transmission ordered by the *utility value* that the prediction algorithm assigned them. Upon receipt of metadata, a node can decide whether to retrieve the corresponding message.

Finally, in order to limit the transmission of information with limited predicted utility, nodes can set *a utility threshold* under which messages are not pre-staged, even if predicted to be of relevance for the user.

D. SoA Bridge

The SoA Bridge is a key component within this scenario that interconnects the two different architectural approaches (enterprise and tactical). It allows the reliable connection and communication between the SoA represented by the MC2SA SoI (described in the next paragraph) and the P2P message-oriented middleware DisServicePro, with the goal of enabling tactical information from the above mentioned SOA environment to be disseminated directly to the edge nodes using the P2P features offered by DisServicePro.

In order to achieve its goals within the SoA environment, the SoA Bridge has been designed and implemented over the MC2SA SoI's SDK (Software Development Kit) to take advantage of the Advanced Message Queuing Protocol (AMQP) messaging bus and publish-and-subscribe model. The primary task of the SoA Bridge is in fact to subscribe to a specific set of AMQP topics defined at the SoI level and receive strategic information from different information sources and clients connected to the SoI. All information published by clients within this scenario (e.g. Joint Tactical Common Operating Picture Workstation, or JTCW) is translated into a SoI native message format. This allows the correct parsing and validation of the exchanged messages between the nodes and the SoI.

Furthermore, the SoA Bridge supports the capability of filtering and forwarding messages generated by any of the edge nodes all the way back to the SoI. These messages are then ready to be eventually consumed by any client that supports either a one-way or two-way communication (e.g., a publish-and-subscribe mechanism) to the SoI, generally through the AMQP protocol.

E. Mockets and NetProxy

Traditional communication protocols such as TCP and UDP were not designed for tactical networks. TCP identifies packet loss as congestion thus reducing the sending rate and cannot distinguish between a lost packet and a late acknowledgment typical of high-latency links. UDP does not suffer from these issues but in many situations we need to be able to transmit packets reliably. For these reasons while putting in place the described architecture we decided to leverage on Mockets for the communication.

Mockets is a communication library specifically designed to provide high performance in mobile ad-hoc networks [15]. Mockets is a flexible communication protocol that runs over UDP and offers an end-to-end connection capable of several delivery services: messages can be sent reliably or unreliably and in a sequenced or non-sequenced fashion. One of the main components of Mockets, the Network Condition Monitor [16], collects end-to-end network statistics on the state of the communication link such as peer reachability, bandwidth and latency. By leveraging on the information collected, Mockets can adjust its internal parameters to maximize throughput and minimize latency, thus increasing the liveliness of data in various deploying environments.

The NetProxy allows the integration of Mockets with legacy systems using TCP or UDP. The NetProxy transparently captures the outgoing TCP or UDP traffic on configurable ports and redirects it to Mockets. The NetProxy can also be configured to perform traffic shaping and prioritization.

F. AMQP

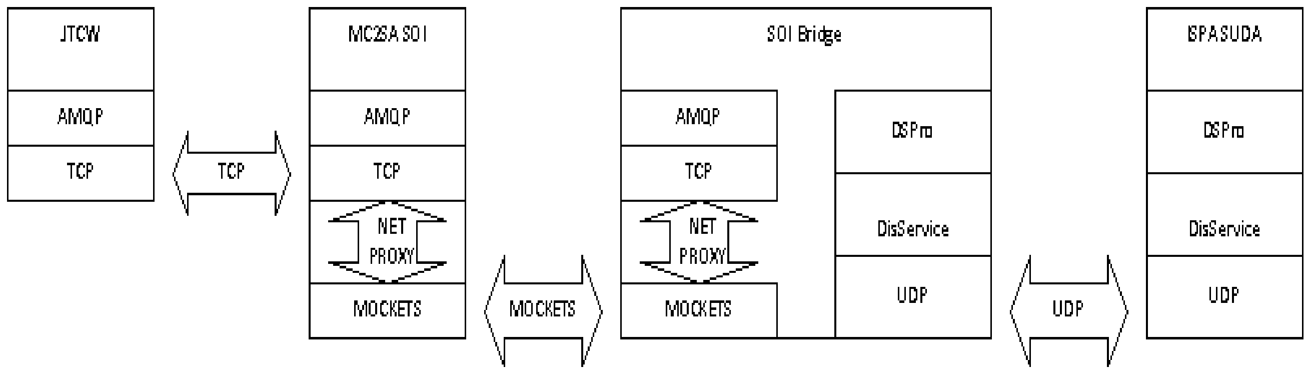
The Advanced Message Queuing Protocol (AMQP) [11] is an application layer protocol for message-oriented middleware designed with an open standard specification. AMQP offers key features for our approach like point-to-point and publish-and-subscribe message routing and queuing. AMQP allows the design and implementation of messaging architectures using a clear set of message patterns with a secure and easy to deploy model. In fact, the primary model used in this scenario is a publish-and-subscribe mechanism that allows the distribution of information to a set of recipients according to specific subscription criteria. This particular queuing model has provided the whole set of features needed by our implementation.

G. JTCW

Joint Tactical Common Operating Picture (COP) Workstation (JTCW) Client and Gateway (C&GW) is a client application that provides features like the display of tactical track data from a Tactical Database Manager (TDBM) data server and a full cartographic and track API support.

JTCW offers a modular architecture and supports a variety of digital map data, a full overlay editor with export/import

Figure 2



capabilities. Furthermore, JTCW supports the MIL-STD-2525B standard and Naval Tactical Data System (NTDS) symbology for track display and overlays, which was particularly useful in our context.

JTCW uses a C2PC Client Data Connector that provides an interface between the SoI and C2PC clients that allows JTCW to exchange tactical data with our SoA. This also allows for data to be shared with C2PC clients that do not currently have access to. Since the C2PC Client Data Connector interfaces with JTCW, it allows fine-grained control for users to be able to subscribe for data from the SoI and work with it locally. The C2PC Client Data Connector makes extensive use of AMQP to deliver messages and tactical data.

V. Implementation Details

There are three major components involved in realizing the goal of extending enterprise-level SoAs running on higher-echelon nodes to connect with tactical dismounted users, and they are depicted in Figure 2. The first component is the SoA Bridge, which interconnects the MC2SA SoI with DisServicePro. One of the challenges was being able to efficiently fetch and translate messages coming from clients connected to the SoI (e.g. JTCW) in a format that is easy to process and efficient to transmit and disseminate to tactical edge nodes by DisServicePro. As a prerequisite, all the clients that want to exchange data with the SoI need to use the SoI native XML format, which follows the XML 1.0 Specification produced by the World Wide Web Consortium (W3C).

When a client, like JTCW, sends tactical data (through AMQP) to the SoI, the message is forwarded by the AMQP daemon (running on the SoI) to the SoA Bridge via its subscriber component. The SoA Bridge was developed in accordance with the MC2SA SoA Software Development Kit (SDK). The SoA Bridge connects to the SoI via a secure TLS (Transport Layer Security) connection.

Once the SoA Bridge has been authenticated, it subscribes to information that might be published in the SoI. Currently, this consists primarily of tracks. As new track information is published, or tracks are updated, the information is pushed to the SoA Bridge, which subsequently transforms the information into an intermediate format (called the SoI-DisServicePro exchange format).

In this particular implementation, JTCW and the MC2SA SoI were deployed in the same Local Area Network (LAN), while the SoI Bridge connects to the MC2SA SoI over a Wide Area Network (WAN) and it is performed over mockets, by transparently redirecting the AMQP's TCP traffic via the NetProxy. While this is a possible configuration, there may be cases where applications (such as JTCW) and MC2SA SoI may not be deployed in the same LAN, and they may connect over WAN instead. In these cases, it would be possible to use the NetProxy to also redirect the application- MC2SA SoI traffic over mockets.

The SoI-DisServicePro exchange format message is completely transparent at the DisServicePro level (it is data that must be replicated following the specified policies) but can be instead processed, transformed and possibly dropped by the SoA Bridge and the applications running on the edge nodes. The correct processing of the SoI- DisServicePro exchange messages is important in order to keep track of the message at the application level and to filter the above-mentioned messages based on the needs of the application.

This leads to the implementation of a concrete solution for handling different types of tactical data messages sent by clients like JTCW. As described above, generate and submit tactical data (e.g. Tracks) to the SoI by pushing SoI native XML messages to the AMQP daemon that handles subscriptions from other clients and routes the information to them. While this model is adequate for clients operating at upper echelons, it does not scale down to the tactical edge, where bandwidth and connectivity is severely limited. This adaptation is performed by the second component, DisServicePro, which selects only the set of tracks that are relevant to each dismounted user and pushes that subset to the edge node.

The third component is the end-user application that receives information from DisServicePro and displays it to dismounted users. DisServicePro has been successfully integrated with the ISPA SUDA (Small Unit Decision Aid) application for Android™, which offers a map-oriented display of geographically relevant data on a mobile device.

The DisServicePro and SoA Bridge components are two-way, in the sense that they can also handle information being generated at the edge by the SUDA application and disseminate the information appropriately. The information is disseminated in a decentralized manner to other edge nodes, as well as being made available at higher echelons via the SoI. The published data can then be received by any connected subscriber (with the related topic) or pushed by the SoI Web Services to external applications. This functionality was successfully tested with Google Earth™ and NASA World Wind.

VI. Experimentation Scenario

Blue Force Tracking (BFT) is a tactical application that provides the allied force with situational awareness data of their positions and is critical to avoid friendly fire accidents. It is therefore necessary to disseminate BFT information over the network in a timely fashion. However BFT information has limited spatial relevance: nodes that are far apart that cannot interact do not need to know about their respective positions and hence may not need to share BFT information among each other. Thus, bandwidth can be saved to transmit more relevant information. On the other hand, nodes at higher echelons may want to have a global view of the position of the forces and may need to receive BFT from each node in the network. DisServicePro can support both cases by configuring the appropriate values for utility based on distance and the proper coordinates that describe the area of relevance for the BFT message. For example, the COC may set up a value for the useful distance that contains the whole area where the operation is taking place, while an edge node may want to limit the reception of BFT messages only to the nodes that are in a few-mile-range from it. Each time two nodes encounter, they both will match the metadata they have in their own local caches against each other's node contexts and matching metadata is exchanged. Because the nodes operate in a store-carry-and-forward fashion, the BFT message may either reach the COC directly, or through a harvester node, and therefore the BTF information can reach the SoI and its subscribers.

VII. Experimentation Results

Table I

Useful Distance (Meters)	*Number of Messages*	*Perc. of Pre-Staged Messages*
100	56	0.06%
1000	161	0.16%
10000	209	0.21%

While extensive testing of the system and its capabilities have been performed, in this section we try to give a sense of the benefits provided by the filtering capabilities of DisServicePro (that is, the ability of DisServicePro to pre-stage relevant information to neighboring nodes). For this reason, we designed a simple experiment that tries to reproduce a real world scenario in which a node follows a track of six waypoints and with total length of approximately 2.5 KM, and, as it moves along the path, it receives data that were generated by sensors located in the area surrounding the path.

The sensors are within an area of 4 KM by 3 KM, which also contains the whole path of the node. Moreover, for additional realism, the sensors are not uniformly distributed over the whole area; instead they are clustered around several strategic locations. For repeatability, we assume that the sensor data is generated prior the deployment of the moving node; therefore, instead of simulating each sensor node, we simulate a single node (that we assume to be the COC) that is pre-load with all the messages that were generated by the sensors (1000 messages). In reality, these messages would originate dynamically from deployed sensors. In order to evaluate the effectiveness of the filtering, we run the experiment configuring the mobile node with different values of useful distance (described in Section IV, paragraph C), while the utility threshold is set to 60% of the maximum value of utility.

The results are reported in Table I, and show that the filtering capabilities of DisServicePro effectively reduce the number of messages that are sent to the node. The filtering has the twofold benefit to limit the amount of data that is transmitted over the network, and to limit the operator overload: the user will not be distracted by the arrival of new, irrelevant data.

VIII. Conclusion

In the paper we presented our approach to extend SoAs to the tactical environment. One of our objectives was to allow full compatibility with the U.S. Marine Corps' MC2SA SoA, without requiring any modification to the MC2SA systems. The objective was achieved by implementing an intermediate component, the SoA Bridge that allows both the SoA and the P2P subsystems to exchange tactical data while at the same time achieving connection reliability on constrained links and consistency of the information on the edge nodes. This approach allows the two subsystems to dynamically change their behavior, based on the requests and the tactical data that they are receiving from each other at runtime, while adhering to configured policies that might be in place.

Tests and experiments to evaluate the reliability of this architectural solution were conducted using the NOMADS Tactical Network Emulation Testbed (which utilizes a modified version of the Mobile Ad-hoc Network Emulator (MANE) [17]). The results show the benefits of DSPro, the SoI Bridge, Mockets, and NetProxy. In particular, the benefits of DSPro to downselect the number of tactically relevant messages for each edge user significantly reduces the bandwidth utilized and the number of irrelevant messages presented to the user.

Acknowledgment

This work is supported by the Office of Naval Research under grant N00014-09-1-0012 and the U.S. Army Research Laboratory under cooperative agreement W911NF-11-2-0095.

References

[1] Suri, N. (2009). Dynamic service-oriented architectures for tactical edge networks. Proceedings of the 4th Workshop on Emerging Web Services Technology - WEWST '09 (pp. 3-10). New York, New York, USA: ACM Press.

[2] Lund, K., Eggen, A., Hadzic, D., Hafsoe, T., & Johnsen, F. (2007). Using web services to realize service oriented architecture in military communication networks. IEEE Communications Magazine, 45(10), 47-53.

[3] Suri, N., Benincasa, G., Tortonesi, M., Stefanelli, C., Kovach, J., Winkler, R., Kohler, U. S., et al. (2010). Peer-to-peer communications for tactical environments: Observations, requirements, and experiences. IEEE Communications Magazine, 48(10), 60-69.

[4] Mayk, I., Regli, W., Nguyen, D., Mai, M., Chan, A., Urness, T., Goren, B., et al. (2011). Net-centric information and knowledge management and dissemination for data-to-decision C2 applications using intelligent agents and service-oriented architectures. 2011 - MILCOM 2011 Military Communications Conference, 1568-1573.

[5] Robert, N. L., Macker, J., Millar, D., William, C. R., & Taylor, I. (2010). XO: XMPP overlay service for distributed chat. 2010 - MILCOM 2010 Military Communications Conference, 1116-1121.

[6] Suri, N., Marcon, M., Quitadamo, R., Rebeschini, M., Arguedas, M., Stabellini, S., Tortonesi, M., et al. (2008). An Adaptive and Efficient Peer-to-Peer Service-Oriented Architecture for MANET Environments with Agile Computing. NOMS Workshops 2008 - IEEE Network Operations and Management Symposium Workshops, 364-371.

[7] Galatopoullos, D. G., Kalofonos, D. N., & Manolakos, E. S. (2008). A P2P SOA enabling group collaboration through service composition. Proceedings of the 5th international conference on Pervasive services - ICPS (p. 111). New York, New York, USA: ACM Press.

[8] Sacha, J., Biskupski, B., Dahlem, D. Cunningham, R., Meier, R., Dowling, J., and Haahr, M. Decentralising a service- oriented architecture. In Journal of Peer-to-Peer Networking and Applications, October 8, 2009.

[9] Halonen, T., & Ojala, T. (2006). Cross-layer design for providing service oriented architecture in a mobile Ad Hoc network. Proceedings of the 4th international conference on Mobile and ubiquitous multimedia- New York, New York, USA: ACM Press.

[10] Facchini, C., Granelli, F., & Da Fonseca, N. L. S. (2010). Cognitive service-oriented infrastructures. Journal of Internet, 4(1), 269-278.

[11] Vinoski, S. (2006). Advanced Message Queuing Protocol. IEEE Internet Computing, 10(6), 87-89.

[12] Lin, B., Ioup, E., & Sample, J. (2010). The NRL Geospatial Hub for ocean sensor processing and collaboration. OCEANS 2010 MTS/IEEE SEATTLE (pp. 1-5). IEEE.

[13] Benincasa, G., Rossi, A., Suri, N., Tortonesi, M., & Stefanelli, C. (2011). An Experimental Evaluation of Peer-To-Peer Reliable Multicast Protocols. 2011 - MILCOM 2011 MILITARY COMMUNICATIONS CONFERENCE (pp. 1015-1022). Baltimore, MD.

[14] Rota, S., Benincasa, G., Interlandi, M., Suri, N., Bonnlander, B., Bradshaw, J., Tortonesi, M., et al. (2010). Supporting information on demand with the DisServicePro Proactive peer-to-peer information dissemination system. 2010 - MILCOM 2010 MILITARY COMMUNICATIONS CONFERENCE (pp. 561-568). San Jose, CA.

[15] Suri, N., Tortonesi, M., Arguedas, M., Breedy, M., Carvalho, M., & Winkler, R. (2005). Mockets: A Comprehensive Application-level Communications Library. 2005 - MILCOM 2005 MILITARY COMMUNICATIONS CONFERENCE. Atlantic City, NJ.

[16] Stefanelli, C., Tortonesi, M., Carvalho, M. Suri, N. (2007). Network Conditions Monitoring in the Mockets Communications Framework. 2007- MILCOM 2007 MILITARY COMMUNICATIONS CONFERENCE. Orlando, FL.

[17] N. Ivanic, B. Rivera, B. Adamson, “Mobile Ad Hoc Network Emulation Environment”, in Proceedings of 2009 IEEE Military Communications Conference (MILCOM 2009).