A Proxy Gateway Solution to Provide QoS in Tactical Networks and Disaster Recovery Scenarios

Alessandro Morelli, Cesare Stefanelli, Mauro Tortonesi
Department of Engineering
University of Ferrara
Ferrara, Italy
+39 0532 974988
{alessandro.morelli, cesare.stefanelli, mauro.tortonesi}@unife.it

Rita Lenzi[1], Niranjan Suri[1,2]
[1]Florida Institute for Human and Machine Cognition
Pensacola, FL, USA
[2]US Army Research Laboratory
Adelphi, MD
+1 850 202 4444
{rlenzi,nsuri}@ihmc.us

ABSTRACT

Many important public services, such as security and public health, as well as the modern tactical military scenarios, rely on Service-oriented Architectures (SoAs) and commercial off-the-shelf (COTS) components to enable the quick development and deployment of distributed services to respond quickly, reduce costs, and ease system integration. However, SoAs make use of verbose networking technologies and require reliable and relatively high bandwidth communications. Tactical scenarios normally cannot rely on such infrastructure and events like natural disasters can severely damage the network infrastructure in rural and urban environments. Thus, there is a need to develop solutions that provide SoA-based application and services running on heterogeneous and often constrained devices that compose tactical and mobile ad-hoc networks with Quality of Service (QoS) levels that meet their requirements. This paper presents the QoS-enabling features and the gateway operational mode (GM) of ACM NetProxy, the network proxy component of a communications middleware specifically developed to support applications in challenged networks. GM allows nodes in an ad-hoc wireless network to be quickly organized and to shape outbound communications to reduce bandwidth consumption and provide QoS. Experimental results obtained during a test in a field demonstration event show its efficiency.

Categories and Subject Descriptors

C.2.1 [**Network communications; Wireless communication**]: QoS and quick setup of ad hoc wireless networks to restore most critical public services during disaster recovery.

General Terms

Management, Performance, Design.

Keywords

Communications middleware; network proxy; QoS; disaster recovery; tactical networks.

Q2SWinet'15, November 2-6, 2015, Cancun, Mexico
© 2015 ACM. ISBN 978-1-4503-3757-1/15/11…$15.00
DOI: http://dx.doi.org/10.1145/2815317.2815318

1. INTRODUCTION

A Service-oriented Architecture (SoA) is a design paradigm that enables quick and easy development of distributed services and applications. SoAs achieve these results by making available components that can be remotely accessed using network communication protocols through a well-defined interface. Most common SoA solutions base their communications on standard web service technologies, such as Simple Object Access Protocol (SOAP) or Representational State Transfer (ReST) Application Programming Interfaces (APIs), which rely on the exchange of XML or JavaScript Object Notation (JSON) documents over plain HTTP or embedded in SOAP Remote Procedure Calls (RPC) requests and responses. SoA promotes maintainability, extensibility, and loose coupling of components and it enables reusability of services, composability via orchestration or choreography, and their rapid (re)configuration. These advantages promoted the adoption of SoAs within large enterprises and organizations, which can benefit from the support of a fast and reliable network infrastructure, well suited for handling the exchange of verbose messages typical of SoA-based systems.

The benefits provided by SoAs argue in favor of their adoption in other networking environments such as Tactical Edge Networks (TENs). However, TENs are extremely challenging wireless networking scenarios, typically composed of a combination of Mobile Ad-hoc Networks (MANETs) and Wireless Sensor Networks (WSNs), which suffer from high packet loss due to weak signals, interference, and the possible presence of obstacles in the medium. Mobility of nodes further deteriorates communications in TENs by increasing the network churn rate and the frequency of disruptions in end-to-end connections. Moreover, connectivity technologies used in TENs are highly heterogeneous, composed of segments with different bandwidth, latency, reliability, and availability characteristics [1]. Finally, it is important to note that computational and connectivity resources of nodes, mission objectives, and battlefield characteristics all contribute to pose significant constraints on the admissible network configuration and on the role that can be played by each node.

Let us note that TENs are a very relevant communication environment outside of military applications also. In fact, military communication infrastructures and equipment are often deployed in disaster recovery situations – a civilian application of essential importance. In addition, research in military communications has produced several important results outside the warfighting domain, such as the concepts, methodologies, and tools that gave birth to research on opportunistic networking and fostered its development [20].

The peculiar characteristics of TENs make them unsuited for SoA-based applications, which were designed for the Internet and rely on the Transmission Control Protocol (TCP) to support the message exchange between two components. In fact, the verbosity of web services does not cope well with the limited bandwidth available in TENs, while TCP exhibits many problems in wireless, mobile networks that lead to bandwidth underutilization, increased network latency, and connection disruptions [2].

A middleware-based approach is a very interesting solution to enable the porting of SoA-based software to the tactical environment. Communications middleware specifically designed to support communications in TENs could provide applications with the right set of tools, abstractions, and techniques to optimize utilization of network resources and increase performance [1] [6]. For instance, providing better utilization of the available bandwidth, decreased latency and jitter, and resilience to connection disruptions would hide applications from the problems of challenged networking environments and enable the porting of SoA-based applications to TENs while at the same time significantly improving their QoS as well. Such a solution is very appealing due to its effectiveness and simplicity, but it requires changes to all applications that need to access remote resources over the network. In addition, a non-negligible portion of TENs is composed of significantly resource constrained devices, which cannot provide the computational capabilities to run dedicated communication components, i.e., middleware or legacy / proprietary software solutions, or any kind of adaptation component between those solutions and COTS applications.

To address this issue, this paper proposes a *transparent QoS proxy* approach that leverages the NetProxy component of the Agile Computing Middleware (ACM) [3] [4]. ACM NetProxy is a network proxy solution that enables the reuse of COTS and SoA-based applications, and was originally designed to run on each node and operate in an end-to-end fashion (Host Mode, or HM, in NetProxy terminology). This paper presents a new operational mode of ACM NetProxy, called Gateway Mode (GM), that provides several advantages over HM. GM brings the advantages of NetProxy to all nodes belonging to a network or a subnetwork, without having to host the proxy on each node. GM improves the network performance while also reducing the computational overhead on the nodes and simplifying the necessary configuration. Additionally, GM allows many resource-constrained devices, such as smart phones and sensor nodes, which would not be able to run ACM NetProxy, to still benefit from its features.

This paper discusses the operating principles of the GM feature and gives some insights on its implementation. It also presents some experimental results obtained using ACM NetProxy running in GM during a test in a field demonstration event.

2. CHALLENGES OF RUNNING SOA APPLICATIONS OVER TACTICAL NETWORKS

Tactical Edge Networks (TENs) are typically wireless ad-hoc networks, with little or no infrastructure support, that are set up in case of emergency, e.g., to provide the connectivity service during disaster recovery, or to be used in the battlefield during military operations, where rapid network reconfiguration is essential to withstand changes in the mission's targets or to respond promptly to enemy actions. The wireless nature of the system, ad-hoc connectivity and high network churn rate, mobility of nodes, their heterogeneity, and rapid changes in the network configuration greatly hinder communications between applications.

Nodes in TENs can be characterized based on the level of resources at their disposal (in terms of processing power, memory available, battery life, and so on), ranging from battery-powered sensors up to high-powered servers at the Tactical Operations Center (TOC) or Combat Operations Center (COC). Another classification concerns nodes' degree of mobility, which ranges from static, like sensors, to dynamic, e.g., soldier platoons, to extremely dynamic, such as (un)manned ground and air vehicles. A wide variety of networking technologies provides connectivity among all those different types of nodes, including SATCOM links, 3G/4G communications, and other local wireless solutions such as WiFi and Bluetooth [1]. The great variety in capabilities, mobility, and connectivity that describe nodes in TENs puts severe limits on the network configuration and on the set of nodes that can run some specific services.

Service-oriented Architectures have earned much success and adoption as a solution to the need for rapid service setup, deployment, and (re)configuration in large-scale systems. This is done thanks to the integration of multiple, independent components that can be accessed over the network via well-defined interfaces. This fosters the reuse of existing components, promotes loose coupling and interoperability, and reduces design and development times. Moreover, the usage of directory and/or discovery services permits the dynamic addition of entities to satisfy new needs that arise during the lifetime of the system and to increase fault resilience.

These features are extremely appealing in tactical environments, as proved by their adoption in projects such as the US Army Technical Reference Model (TRM) [5] and the US Marine Corps Tactical Service Oriented Architecture (TSOA). However, SoA implementations typically rely on verbose Web Services technologies over TCP connections. The resulting high bandwidth demands and the inability to cope with link disruptions that follow from such technologies only suit infrastructure networks capable of providing reliable, high-speed connectivity among the parts of the SOA.

These problems identify additional requirements for SoA-based applications that need to operate in TENs. A solution is needed to better support SoA services in the face of limited bandwidth, high and variable latencies, frequent link disruptions, and network partitioning.

3. ENABLING THE REUSE OF SOA AND COTS APPLICATIONS USING A GATEWAY NETWORK PROXY

Communications middleware specifically designed for TENs represent an appealing solution to effectively port SoA-based applications and services to that operating environment [1]. In fact, middleware solutions specifically designed to support communications in extremely challenging networking environments would provide applications with the right set of functionalities, abstractions, and techniques to optimize the access to and the utilization of network resources, and ultimately increase performance. By relying on such a solution, service designers and developers can focus their efforts on the business logic instead of dealing with networking-related issues and challenges. This promotes productivity and final software quality, and it also reduces design, development, and maintenance costs [6].

While this approach is very interesting due to its effectiveness and simplicity, it still requires all networked services and applications to be modified before they can exploit any of the communications middleware capabilities. In many cases, with legacy and COTS software solutions, such modifications are not possible or, even when technically feasible, would be prohibitively costly in terms of time and money. For these reasons, it is essential to find an effective way to bridge the gap between the communications middleware and the existing applications.

A rather simple and effective solution involves inserting a *transparent QoS adaptation layer* between the applications and the communications middleware. This layer consists of a smart component that is capable of remapping service requests and applications' communications to the right methods and concepts provided by the middleware in a completely transparent manner. This way, applications can have access to the optimizations and the enriched networking features of the communications middleware without the need for changes in their source code. This is an approach we followed with success with ACM NetProxy [4].

Nonetheless, the increasing presence of constrained devices in TENs pose several challenges to this approach. The very low resources available on many types of nodes in TENs often prevent additional components (or even the communications middleware itself) from running on those devices. Security and other tactical policies might also hinder the installation of additional software on some nodes, and the presence of multiple enterprises and stakeholders that control the nodes further aggravates this condition.

Instead, a more promising approach is a *transparent QoS proxy* based on the deployment of the transparent QoS adaptation component on a resource rich node operating as a gateway for constrained devices. This approach addresses the challenges described above by separating the location of applications from that of the communications middleware and QoS optimization components. More specifically, the transparent QoS proxy approach avoids the emergence of policy-related problems that would prevent the installation of additional components on the nodes. Furthermore, it allows the middleware to be installed only on nodes with sufficient computational and memory resources and located at convenient points in the network or covering some specific role in the TEN. For example, nodes that are central to the network or in gateway positions would be better able to observe the network traffic to infer the network state.

The transparent QoS proxy running on the gateway node intercepts and processes all traffic generated by (typically constrained) nodes within a portion of the network (labeled as "Internal Network" (IN)) and forwards it to destinations with the support of the communications middleware. The gateway node effectively separates the nodes belonging to the IN from the rest of the network, or "External Network" (EN), and hence it becomes a gateway for the IN. We will refer to this node as "gateway" or "proxy gateway", interchangeably. As a consequence, the network topology might affect the choice of the node that will host the proxy.

Of course, it is essential to pay attention at deployment time: all the nodes in the TEN that want to exploit the transparent QoS enhancement features convey the traffic to and from their communicating peers through the gateway node running the transparent QoS proxy. However, most typical network configurations often have a single gateway node that regulates communications to and from network addresses that are locally unreachable, and so this is usually not a problem. In any case, any communications that do not go through the gateway node can still proceed normally, but they will not take advantage of the QoS improvements provided by NetProxy.

4. ACM NETPROXY

The ACM [1] is a communications middleware that provides applications with a rich collection of capabilities to improve network performance and resource utilization. The features of the ACM include network monitoring, data transport, data dissemination, resource and service discovery, transparent network proxy, and network visualization. Many components of the ACM, including NetProxy, are available on GitHub (http://www.github.com/ihmc/nomads) as open source.

ACM NetProxy [3] [4] provides transparent integration between networked applications and the ACM. The most relevant features of NetProxy include QoS improvement and adaptation mechanisms such as network protocol remapping, traffic characterization, data compression, intelligent buffering, flow prioritization, connection multiplexing, and packet consolidation. Protocol remapping allows forwarding (part of) the traffic generated by applications over other network protocols or other components of the ACM, such as Mockets, DisService, and DSPro, transparently. Network traffic characterization constantly analyzes the volume of traffic in the network to provide the decision making component of NetProxy with updated information about what nodes and applications are generating traffic, what types of data are being transmitted, the current bandwidth consumption, and observed radio/link performance. Thus, the QoS related decision making process in NetProxy is both application- and network- aware.

Application-awareness allows NetProxy to choose the QoS improvement techniques that best suit the data exchanged, e.g., preferring data-specific compression schemas over general purpose compression algorithms, prioritizing the data produced by most critical services or addressed to important nodes of the TEN, or identifying traffic flows that would benefit from the delivery semantics and the features of other ACM components. Similarly, network-awareness guarantees that NetProxy can select the network protocols and/or the components of the middleware that better match the state and the characteristics of the network, in terms of bandwidth availability, average latency, link reliability, nodes' mobility, etc.

ACM NetProxy is open source and was designed for easy extensibility. These properties make it possible for users to simply enhance the protocol remapping functionality of NetProxy with support for other transport protocols or communications libraries, such as the Stream Control Transmission Protocol (SCTP) [13] and UDP-based Data Transfer (UDT) [14]. Additionally, NetProxy supports two operational modes, namely Host Mode (HM) and Gateway Mode (GM), which allows it to be used in many different network configurations and satisfy various requirements.

The objective of making NetProxy an extremely flexible solution while maintaining complete application transparency underlies all our design choices and motivated the development of many features. By giving applications access to the features of the middleware without requiring any changes to their source code, NetProxy becomes the keystone to enable the reuse of SoA and COTS components in TENs.

In order to provide its services to networked applications transparently, ACM NetProxy intercepts and process all packets they send over the network. This way, the proxy can acquire knowledge on the traffic that the network will need to accommodate, and exploit it to improve the decision making process to a point that single applications, or even other components of the middleware, would not be able to reach.

When packet interception occurs, NetProxy analyzes its content to extract useful information, including the source and destination IP:port pairs, the transport protocol used, other interesting fields in the protocol header, e.g., flags or enabled options in the header of TCP packets, and the type of data carried within the packet. NetProxy can further enrich this information with the data provided by other ACM components. For instance, Mockets has both an active and passive measuring system that allows the ACM component to learn about the characteristics of end-to-end connections and equipped network interfaces. NetProxy can leverage this feature of Mockets to acquire fundamental information, e.g., the RTT of the connection to the tactical operation center over the SATCOM interface.

Combining this knowledge together with the status of its internal buffers and user-provided information, NetProxy can build an accurate representation of the state of the network and the open connections and use it to manage better the available resources and respond to the QoS requirements of proxied traffic.

The configuration of the QoS module of NetProxy is currently read from a file; a template is available to users, who can modify it and specify several options to instruct NetProxy on how to manipulate the traffic flowing through the proxy gateway and enable QoS for specific data streams. Possible actions include enabling/disabling data compression and choosing data-specific compression algorithms, reducing the resolution of images and video streams, consolidating multiple packets addressed to the same destination, reserving a greater amount of bandwidth for some connections (traffic flow prioritization), temporarily disrupting connections when the available bandwidth goes below a threshold, temporarily switching reliability in specific end-to-end connections to partial reliability or best-effort, remapping communications over different transport protocols, and changing the information dissemination strategy for multicast and broadcast communications.

Some types of traffic manipulation, such as packet consolidation, protocol remapping, or data compression, require a second instance of NetProxy to perform the inverse operations on the other end of communications to preserve transparency. When this happens, it is not necessary that the operational modes chosen for the two instances of NetProxy are equal. Hence, the operational mode can be chosen uniquely to satisfy other types of requirement, such as those due to limited nodes' capability, network topology, or policies enforced by one or more tactical entities, thereby keeping the flexibility of our solution high.

5. THE GATEWAY MODE OF ACM NETPROXY

GM differs from HM in the way ACM NetProxy intercepts packets and in the role that the node running the proxy needs to assume in the network. In [3] we described in detail the design of HM and its main requirements. To summarize briefly, HM requires all nodes hosting components of a SoA or running pieces of COTS software that need to benefit from the ACM to run a copy of the NetProxy locally. To enable packet capturing before packets are sent out over the network, HM also necessitates the installation of a virtual network interface and to configure the proxy properly on each node.

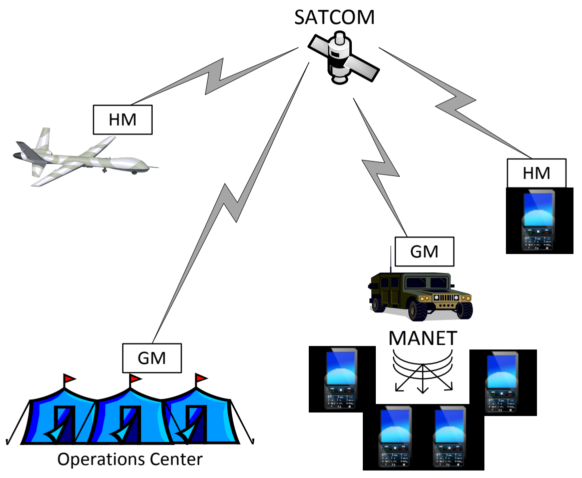

Figure 1.Tactical Network Scenario with NetProxy

When configured to run in GM, NetProxy implements the gateway network proxy solution described in section 3. Therefore, to function properly in this operational mode, ACM NetProxy has to run on nodes equipped with at least two network interfaces. The proxy uses one of them, named "internal interface", to intercept all packets generated by nodes in the IN and addressed to nodes that do not belong to the same subnetwork. The other interface, labelled "external", is used to send the captured traffic to remote destinations after the NetProxy has processed it. Figure 1 shows a TEN with two instances of NetProxy running in GM and two running in HM. In this example, all connections and messages generated by nodes belonging to the IN of some proxy gateway have to go through a proxying node to communicate to nodes on their EN.

Compared to HM, GM does not require the installation of virtual network interfaces, nor does it need to run a copy of NetProxy on each node that hosts SoA services or COTS applications. Furthermore, GM reduces the amount of configuration required, since there is only one instance for the entire sub-network, as opposed to one for each node when in HM.

Another very important difference is that GM allows NetProxy to build a comprehensive picture of the traffic that nodes in the IN are generating, what services are being requested, and what kind of data is being transferred. On the other hand, HM only allows the proxy to learn about local services, thus limiting the efficiency of the decision-making process. Moreover, it is often the case that the external network interface (ENI) of gateway nodes in TENs is different from the internal network interface (INI), thereby providing links with very different characteristics and that usually become bottlenecks in remote network communications. By running NetProxy in GM and installing it on a gateway node, the proxy can estimate the bandwidth capacity, the average latency, and the reliability of these links more easily; this information is extremely useful to further improve the quality of the decision-making process.

It is entirely possible to have the equivalent of NetProxy running directly on a router or wireless network device. For example, in a tactical networking scenario, a vehicle such as a Humvee could have a local area network (LAN) to support users tethered to the vehicle, which then connects to a wireless router / device that provides the off-vehicle connectivity. NetProxy in GM would either run between the LAN and the wireless device, or could be directly integrated into the wireless device for complete transparency.

When running in GM, ACM NetProxy uses libpcap (http://www.tcpdump.org/) to sniff packets on both the internal and the external network. To intercept and process network packets transparently, ACM NetProxy first needs to implement a customized version of the Address Resolution Protocol (ARP). Figure 1 shows the physical configuration of a tactical network where two nodes take on the role of proxy gateway and run NetProxy (the other two nodes are running NetProxy in HM). At the Operations Center (OC), NetProxy's INI is connected to the LAN and the ENI is connected to the SATCOM terminal. Likewise, on the tactical vehicle, the INI is connected to the MANET and the ENI is connected to the SATCOM terminal. Consistent with the design principle of transparency, no nodes attached to the vehicle MANET network or the OC LAN network require changes in their configuration. Three different cases might occur. The first case is very simple, and it refers to two nodes in the IN or in the EN that needs to communicate. In such a situation, the proxy gateway node will not partake in the ARP address resolution, but NetProxy will still cache the hardware/protocol addresses pair of the two nodes involved in the communication. This is possible thanks to libpcap, which allows the sniffing of packets regardless of their destination. Note that NetProxy maintains an associative array of hardware/IP address pairs, with the IP address as key, which is updated any time new ARP packets are sniffed. This also allows NetProxy to learn and keep track of which nodes belong to the IN, and which ones belong to the EN.

The second case involves a node in the IN that wants to communicate with a node in the EN, or vice versa. When either happens, NetProxy intercepts the ARP request, changes the source hardware address (SHA) of the requester with that of the ENI of the proxy gateway, and finally forwards the modified packet on the other network interface. Similarly, NetProxy will change the SHA of the corresponding ARP response before forwarding it back on the first network interface. After address resolution is resolved, data exchange can begin. NetProxy will forward packets from one network interface to the other if and only if two conditions are met: (1) the MAC destination address in the packets' header is that of the ENI; and (2) the IP destination address is that of some node in the other subnetwork. The reason we chose to modify the SHA field in ARP packets is twofold: for clarity's sake, as it makes it straightforward to identify packets that need to be forwarded from one subnetwork to the other, and for consistency with the third case.

The last case, and also the most interesting one, concerns a node (we will also refer to it as "source") in the IN that needs to send data to a remote host (also, "destination"). If ACM NetProxy is configured to remap the traffic between those two endpoints, then it is necessary that either the destination node runs an instance of ACM NetProxy in HM, or it is located behind another proxy gateway running NetProxy in GM. For the purposes of this paper, we will only consider this latter case. Before data exchange begins, the source will generate an ARP request to which NetProxy will reply to inform that the target IP address is reachable through the ENI of the proxy gateway. On the destination's side, the remote NetProxy instance will generate an ARP request to obtain the MAC address of the destination node. Once the address is resolved, the end-to-end communication can proceed and the nodes involved can take advantage of the features of NetProxy and of the ACM, in accordance with the configured options. Note that, in case NetProxy is not configured to remap or process the traffic between a node in the IN and some target remote node, the proxy will forward packets to the network gateway, which will take care of their delivery. This allows complete transparency from the point of view of the applications.

ACM NetProxy running in GM requires the assignment of an IP address only to the ENI and it will make use only of the MAC address of that interface when sending ARP packets. We made this design choice in order to reduce the consumption of IP addresses, which might be a limited resource in some complex network configurations, especially when many different parties are involved.

6. EXPERIMENTAL RESULTS

We experimentally evaluated the impact of ACM NetProxy in a reference scenario based on Agile Bloodhound, an annual technology demonstration event held by the US Department of Defense (DoD) Office of Naval Research (ONR) (http://www.onr.navy.mil/Media-Center/Press-Releases/2014/Agile-Bloodhound-ISR-C2-Logistics.aspx). The rationale is to simulate typical operations involving multiple information flows with different characteristics in terms of both the type and the amount of data transferred. Among them, the most relevant flows consist of friendly (blue) and enemy (red) force tracks, sensor reports (audio, images, and/or video feeds), documents (intelligence reports and logistics reports), and chat messages.

In the experiment, several military hub vehicles are connected to the Operations Center (OC) using SATCOM communications links. The purpose of each hub vehicle is to support and provide connectivity to a number of dismounted soldiers, who move either on foot or in vehicles of their own, and use their devices to set up a MANET for communications. The movements of soldiers and vehicles during the event reproduced the patterns of a realistic tactical mission. An instance of ACM NetProxy was running in GM on a node in the network of the OC where it was able to intercept all traffic to and from the SATCOM link. Similarly, all deployed hub vehicles had network gateway machines on which NetProxy was installed and configured to run in GM. This way, all traffic had to go through one of the NetProxy instances before being transmitted over the SATCOM links in any direction.

We configured all NetProxy instances to remap outgoing transmissions, both UDP and TCP, over a reliable Mockets connection open on the SATCOM link. We then enabled the QoS options for data compression with all streams (we used the Zlib library, an open source lossless compression algorithm available online at http://www.zlib.net), and packet consolidation for all UDP messages addressed to the same destination. Note that NetProxy always performs buffering of the traffic on the IN and sends buffered data out on the EN in accordance with the configured prioritization settings. Since no flow prioritization was configured for this experiment, by default NetProxy tries to achieve flow fairness by equally sharing the bandwidth available on the ENI, as measured by Mockets.

Given the very large amount of data collected during the experiment, this section only presents the analysis of one of the most significant portion of traffic: the one containing red and blue tracks and sensor reports flowing from the OC to one of the hub vehicle nodes (which served all the handhelds devices in the MANET). However, all instances of ACM NetProxy were configured to perform the same operations on the data, whose type and magnitude were comparable across the different teams deployed in the scenario. Therefore, we can state that the narrower focus of our restricted analysis does not affect the purpose of this discussion significantly.

For our analysis, we divided the whole duration of the experiment in time intervals of 0.1 seconds and allocated each network event in its corresponding slot. Table 1 presents a statistical summary of collected measurements. It compares the generated traffic (in bytes) and the number of packets sent before and after the traffic was processed by NetProxy. Reported statistics include arithmetic mean, standard deviation, and maximum number of bytes and packets sent over the network in a single interval. The traffic after going through NetProxy is substantially less than the amount generated by the nodes in the IN. Looking at the mean, the effects of data compression and packets consolidation are evident, and show a reduction of 30.6 percent in the average number of generated bytes and of 42.9 percent in the average number of packets sent. Finally, the standard deviation also appears significantly lower after the network traffic has gone through NetProxy. This result entails a less bursty and smoother network activity on the EN compared to the activity on the IN, as figures 3a and 3b below depict better.

Reducing burstiness is essential to enable NetProxy to provision the required QoS. First of all, it avoids many packets being lost on the bottlenecked links due to sudden peaks in the network activity in absence of congestion control. An example would be applications that rely on UDP to transfer data because reliable and/or ordered delivery of messages is not necessary. Smoother data flows also imply a wiser use of the bandwidth on the bottlenecked links because it cuts the frequency of peaks in network activity followed by periods with very low traffic, during which the available bandwidth would be wasted. Finally, keeping burstiness under control reduces the end-to-end jitter experienced by applications, a very important consequence for all classes of real-time applications.

Figures 2a and 2b show in more detail the effects of data compression and packet consolidation in ACM NetProxy. The figures present the data collected during one of the busiest time windows of the demonstration, which spans from 500 to 800 seconds after the beginning of the experiment and includes significant levels of network activity. Figure 2a represents, with light gray bars, the traffic (in KiB) flowing in the IN, and with a dark gray color the traffic sent over the EN by NetProxy. Similarly, Figure 2b highlights the difference between the number of packets flowing in the IN and the EN before and after NetProxy processed the traffic. The graphs show that NetProxy significantly reduces bandwidth consumption by sending less data out on the EN and generating less packets, which in turn also increases efficiency, especially with radios that are packet rate limited or when packet transmission is preceded by a channel access negotiation phase, such as with wireless network interfaces that implement the IEEE 802.11 specifications and standards [19].

Figures 3a and 3b depict the empirical density distribution of the number of bytes and packets, respectively, which were sent over the internal and external networks in each of the 0.1s long intervals in which we partitioned the experiment. The figures show how the buffering strategy implemented in NetProxy is capable of making traffic usage patterns much smoother and more regular, compared to the burstiness that would normally characterize them. This allows for an easier accommodation of the network traffic and leads to more predictable performance. We chose to limit the data reported on the X axis of the two graphs to 20 KiB and 20 packets, respectively, to better show the differences in shape between the two density distributions. In Figure 3a, very sharp peaks (representing the data measured in the IN) stand out against a smooth curve (that represents the data sampled in the EN). Similarly, Figure 3b shows that the density distribution of the number of packets in the EN during each interval has much gentler slopes than that describing the conditions in the IN in the same intervals. Finally, we note that the tail of the curves marked as “Before” would reach almost 200 KiB in Figure 3a, and go beyond 300 packets in Figure 3b. We chose not to represent all data because it would have resulted in almost unintelligible graphs.

Table 1. Mean, standard deviation, and maximum value of bytes and packets sent over the network in each 0.1s interval before and after traffic was processed by the NetProxy

	Generated Traffic (Bytes)			**Packets sent**		
	Mean	***Std. Dev.***	***Max***	***Mean***	***Std. Dev.***	***Max***
Before	5269.3	15510.9	192656	6.41	22.93	312
After	3656.8	5168.0	15777	3.66	4.79	39

7. RELATED WORK

The literature recognizes the efficacy of middleware-based approaches for the resource management in TENs. Both [6] and [7] focus their effort on optimizing the allocation of resources between competing applications and nodes in the network. More specifically, the authors of [7] propose a middleware that is capable of dynamically tuning the network’s configuration and QoS to meet the applications’ requirements under the constraints dictated by the current network conditions. Our solution, instead, focuses on providing QoS enhancements to applications transparently and remapping their communications over other components of the middleware in order to increase efficiency and reduce the impact on the network resources.

The authors of [8] also propose a transparent network proxy, which aims at increasing the performance of TCP by implementing advanced buffer and packet management solutions in wireless environments. The Space Communications Protocol Specification - Transport Protocol (SCPS-TP) (available on the Web at http://openchannelsoftware.com/projects/SCPS) is another transparent network proxy that enhances TCP and UDP for use in spacecraft communications environments. Our approach goes beyond these solutions, which focus only on improving TCP for use under specific conditions, as it exploits a comprehensive communications middleware that provides the delivery semantics and communication paradigms that best fit applications’ needs.

Other proposals, such as I-TCP [9], Mobile-TCP [10], and the Remote Sockets Architecture [11], represent proxy-based solutions that aim to improve TCP in wireless networks. Differently from our proposal, these systems are not transparent to applications and do not provide any specific QoS features to meet applications’ requirements, but they simply focus on increasing the throughput of TCP in wireless networks and its resilience to mobility.

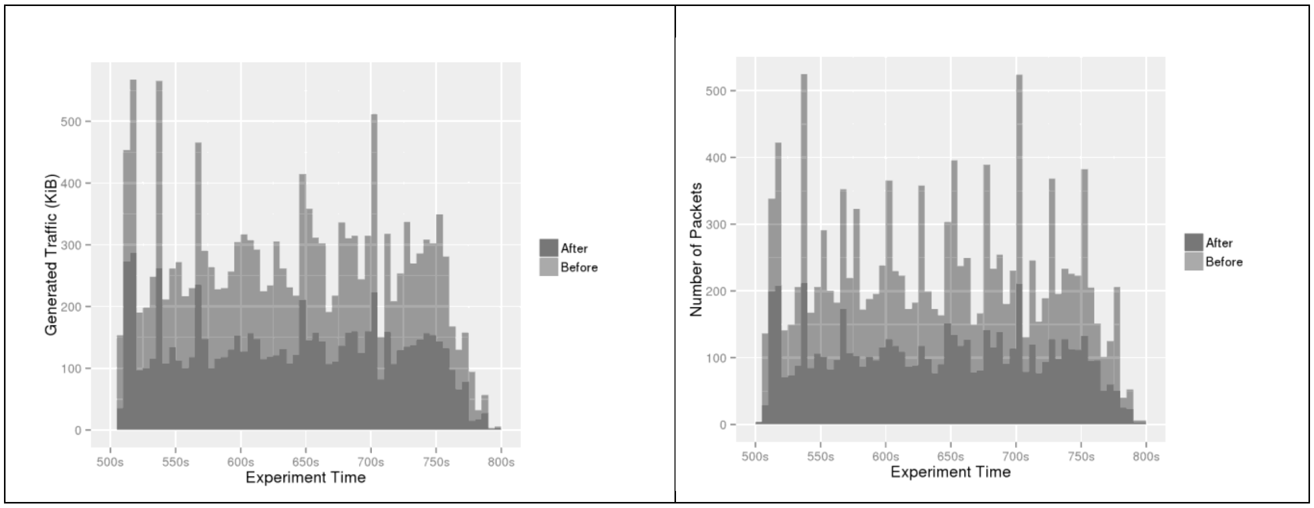

Figure 2. Difference in the number of a) bytes, and b) packets, sent over the network before and after traffic was processed by the NetProxy

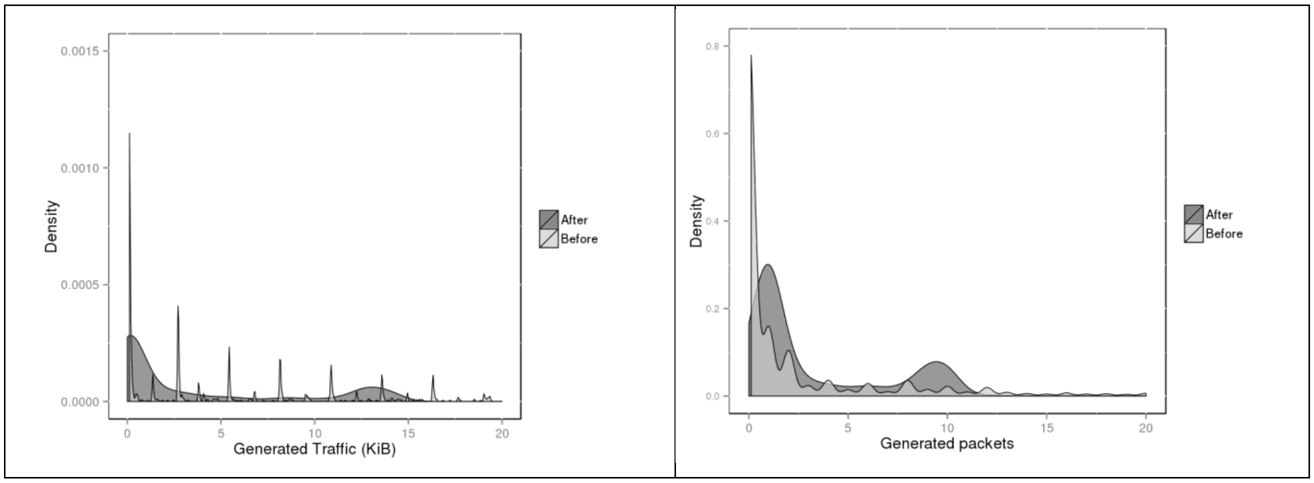

Figure 3. Density distribution of a) the number of bytes (plot limited to 20 KiB), and b) the number of packets (plot limited to 20 packets) before and after traffic was processed by the NetProxy

The ACM NetProxy can be classified as a splitting distributed Performance Enhancing Proxy (PEP) [12]. PEPs exist both as hardware and software solutions, and mostly focus on resolving specific issues that TCP exhibits over particular media or network configurations, such as wireless, satellite, or high bandwidth-delay product links. Unlike them, the NetProxy supports other protocols besides TCP and it adapts to a variety of networks. Moreover, NetProxy can be configured to provide a variety of QoS enhancements to specific data streams and communications.

Several works in the literature focus on systems and techniques to provision QoS to applications in TENs and MANETs. Hauge et al. study the issues of providing QoS in heterogeneous tactical networks and present two QoS-aware network architectures for inter- and intra- domain networks, respectively [15]. However, the paper does not present any experimental evaluation of the proposed solution, and the authors claim that the interactions between the two architectures needs further study. Authors of [16] propose a QoS routing system for MANETs based on the assumption that all nodes can take part in the routing process and that they are equipped with one or more network interfaces capable of operating at one of many independent channels. The paper then focuses only on the problems of clustering and channel allocation.

Kim et al. present a QoS framework for tactical networks based on commercial technologies like DiffServ and SNMP [17]. The framework assumes a hierarchical network architecture with leader nodes that enable communications between one layer of the hierarchy and the one above. These types of network architecture and nodes organization are essential to permit nodes to negotiate their QoS levels within the layers. The paper concludes presenting the results of a simple experimental evaluation, performed using a setup composed of only three static nodes.

In [18], the authors propose QAM, a QoS-aware middleware for communications in tactical environments. To the best of our knowledge, this work shows the highest number of similarities with the ACM. QAM includes components that provide tunable end-to-end connections, point-to-multipoint communications, quality adjustment and admission control features based on measurements of channels and open links, and a transparent proxy component for legacy applications. Nonetheless, the legacy proxy does not interface legacy applications with all components of QAM, but only with the admission control component. In addition, important features such as data compression and packets consolidation seem to be missing, and the QoS level provided by QAM is based on classes, so it cannot be independently configured for each flow.

8. CONCLUSIONS AND FUTURE WORK

ACM NetProxy GM function bridges the gap between services and applications and the Agile Computing Middleware, a communications middleware specifically designed to support communications in extremely challenged networking environments, such as TENs. As shown by experimental results obtained during a test in a field demonstration event, GM enables multiple nodes in subnetworks to benefit from NetProxy and the ACM. This is particularly useful to support handheld devices, embedded devices, and other resource-constrained devices that cannot directly run NetProxy in HM. It will be very interesting to investigate the impact of node mobility, with a particular focus on gateway nodes, on the efficacy of NetProxy to provide QoS to the nodes in the internal network. Other work will aim to increase the intelligence of the decision-making process in ACM NetProxy to enable dynamic adaptation of policies in place. Finally, we will focus further on network state estimation and reprioritization of resource allocation based on the current mission and/or node objectives.

9. ACKNOWLEDGMENTS

This effort was funded in part by the grant of 5‰ to the University of Ferrara - income tax return for 2011, and by the grant for student mobility offered by IUSS - Ferrara 1391.

10. REFERENCES

[1] N. Suri, E. Benvegnu, M. Tortonesi, C. Stefanelli, J. Kovach, J. Hanna, "Communications middleware for tactical environments: Observations, experiences, and lessons learned", IEEE Communications Magazine, Vol. 47, No. 10, pp. 56-63, October 2009.

[2] X. Chen , H. Zhai , J. Wang , Y. Fang, “TCP Performance over Mobile Ad hoc Networks”, Canadian Journal of Electrical and Computer Engineering, Vol. 29, Issue 1/2, pp. 129-134, Jan-Apr 2004.

[3] A. Morelli, R. Kohler, C. Stefanelli, N. Suri, M. Tortonesi, "Supporting COTS applications in Tactical Edge Networks," IEEE Military Communications Conference, 2012 - MILCOM 2012 , pp. 1-7, Oct. 29 2012-Nov. 1 2012.

[4] M. Tortonesi, A. Morelli, C. Stefanelli, R. Kohler, N. Suri, S. Watson, "Enabling the deployment of COTS applications in tactical edge networks," IEEE Communications Magazine, Vol. 51, No. 10, pp. 66-73, October 2013.

[5] S.T. Zhu, R.W. Wong, C.A. McDonough, R.R. Roy, J.M. Fine, J.P. Reiling, “Army Enterprise Architecture Technical Reference Model for System Interoperability”, IEEE Military Communications Conference, 2009 - MILCOM 2009, pp. 1-6, 18-21 Oct. 2009.

[6] A. Poylisher, F. Sultan, A. Ghosh, Shi-wei Li, C.J. Chiang, R. Chadha, K. Moeltner, K. Jakubowski, "QAM: A comprehensive QoS-aware Middleware suite for tactical communications", IEEE Military Communications Conference, 2011 - MILCOM 2011, pp.1586-1591, 7-10 Nov. 2011.

[7] A.S. Peng, D.M. Moen, Tian He; D.J. Lilja, "Automatic Dynamic Resource Management architecture in tactical network environments", IEEE Military Communications Conference, 2009 - MILCOM 2009, pp. 1-7, 18-21 Oct. 2009.

[8] Z. Zhuang, T.-Y. Chang, R. Sivakumar, and A. Velayutham, "Application-Aware Acceleration for Wireless Data Networks: Design Elements and Prototype Implementation", IEEE Transactions on Mobile Computing, vol. 8, no. 9, September 2009.

[9] A. Bakre, B. Badrinath, “I-TCP: Indirect TCP for Mobile Hosts”, in Proceedings of 15th IEEE International Conference on Distributed Computing Systems (ICDCS '95).

[10] Z. Haas, “Mobile-TCP: An Asymmetric Transport Protocol Design for Mobile Systems”, in Proceedings of 3rd International Workshop on Mobile Multimedia Communications (IWMM’95).

[11] M. Schlager, B. Rathke, S. Bodenstein, A. Wolisz, “Advocating a Remote Socket Architecture for Internet Access Using Wireless LANs”, Mobile Networks and Applications, Vol. 6, N. 1, pp. 23-42, Jan./Feb. 2001.

[12] J. Border, M. Kojo, J. Griner, G. Montenegro, Z. Shelby, “Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations”, RFC 3135, June 2001.

[13] Ł. Budzisz, J. Garcia, A. Brunstrom, R. Ferrús. “A taxonomy and survey of SCTP research”, ACM Computing Survey, Vol. 44, No. 4, Article 18, pp. 18:1-18:36, September 2012.

[14] Y. Gu and R.L. Grossman, UDT: UDP-based Data Transfer for High-Speed Wide Area Networks, Computer Networks (Elsevier), Vol. 51, No. 7, May 2007.

[15] M. Hauge, L. Landmark, P. Lubkowski, M. Amanowicz, K. Maslanka, “Selected Issues of QoS Provision in Heterogenous Military Networks”, International Journal of Electronics and Communications, Vol. 60, No. 1, 2014.

[16] A. Dimakis, L. He, J. Musacchio, H. Wilson So, T. Tung, and J. Walrand, "Adaptive Quality of Service for a Mobile Ad Hoc Network", IEEE International Conference on Mobile and Wireless Communication Networks (MWCN), October 2003.

[17] B. C. Kim, Y. Bang, Y. Kim, J. Y. Lee, D. G. Kwak; J. Y. Lee; J. S. Ma, "A QOS framework design based on DiffServ and SNMP for tactical networks," IEEE Military Communications Conference 2008 - MILCOM 2008, pp. 1-7, 16-19 November 2008.

[18] A. Poylisher, F. Sultan, A. Ghosh, S. Li; C.J. Chiang, R. Chadha, K. Moeltner, K. Jakubowski, "QAM: a Comprehensive QoS-aware Middleware Suite for Tactical Communications," IEEE Military Communications Conference 2011 - MILCOM 2011, pp. 1586-1591, 7-10 Nov. 2011

[19] IEEE Standard for Information technology - Telecommunications and information exchange between systems Local and metropolitan area networks - Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, http://standards.ieee.org/getieee802/download/802.11-2012.pdf

[20] L. Pelusi, A. Passarella, M. Conti, “Opportunistic networking: data forwarding in disconnected mobile ad hoc networks”, IEEE Communications Magazine, Vol. 44, No. 11, pp. 134-141, November 2006.