Enabling the Deployment of COTS Applications in Tactical Edge Networks

Mauro Tortonesi[1,*], Alessandro Morelli[1], Cesare Stefanelli[1], Ralph Kohler[2], Niranjan Suri[3,4], Scott Watson[5]

1) Department of Engineering
University of Ferrara
Ferrara, Italy
{mauro.tortonesi,alessandro.morelli,cesare.stefanelli}@unife.it
2) US Air Force Research Lab
Rome, NY, USA
Ralph.Kohler@rl.af.mil
3) Florida Institute for Human & Machine Cognition
Pensacola, FL, USA
nsuri@ihmc.us
4) US Army Research Lab
Adelphi, MD, USA
5) Space and Naval Warfare Systems Center
San Diego, CA, USA
scott.c.watson@navy.mil
*) Corresponding author:

Name: Mauro Tortonesi
Address: Department of Engineering
University of Ferrara
Via Saragat, 1
44122 Ferrara (FE)
Italy
Phone: +39 0532 974888
Fax: +39 0532 974888
E-Mail: **mauro.tortonesi@unife.it**

Enabling the Deployment of COTS Applications in Tactical Edge Networks

Abstract

The increasing adoption of COTS hardware and software technologies in tactical scenarios raises the issue of supporting the deployment of legacy and COTS applications in extremely dynamic and challenging environments such as Tactical Edge Networks (TENs). COTS applications adopt standards devised for wired Internet environments or corporate networks, such as Service Oriented Architectures, and TCP and UDP , thus exhibiting severe reliability and performance problems on TENs. To support the reuse and the deployment of COTS applications in TENs, there is the need to develop solutions that mediate the application requirements with the communication semantics of TENs. This article presents an overview of the challenges in deploying COTS applications in TENs and presents NetProxy, a state-of-the-art solution explicitly designed to address them.

1. Introduction

There is a growing interest in adopting Commercial Off-The-Shelf (COTS) hardware and software technologies in military application environments, such as Tactical Edge Networks (TENs) [1]. In fact, the adoption of COTS technologies enables reaping the benefits of economies of scale, and facilitates and hastens the development and deployment of complex distributed applications by leveraging robust and widely adopted standards and software components.

Legacy and COTS applications were (and sometimes continue to be) developed using standards devised for wired Internet environments or corporate networks, such as Service Oriented Architectures (SOAs) and other web-inspired technologies, and TCP and UDP. When these applications are deployed on TENs, their performance is significantly degraded. In fact, the reliance on Web protocols results in excessive bandwidth utilization in an environment where bandwidth is already scarce. Reliance on TCP results in decreased throughput, decreased bandwidth, and other failures such as connection resets.

In order to support the deployment and reuse of COTS and legacy applications in tactical environments, there is a need to develop solutions that mediate between the communication semantics required by the applications and those that can be reasonably supported by TENs. More specifically, the tactical scenario demands an intelligent substrate that transparently captures the data transmitted by the applications, manipulates it to reduce its footprint, and re-maps it over TEN-specific communication solutions.

This article presents an analysis of the challenges in deploying COTS applications in TENs, which motivated us to develop NetProxy, a state-of-the-art solution explicitly designed to address those issues. NetProxy was originally introduced in [2], which focuses on technical aspects of the tool such as its architecture and implementation, and later extended for the present work. NetProxy is a component of the Agile Computing Middleware, a comprehensive communications solution we designed to support the development of distributed applications for TENs [3].

We note that while the performance and reliability of COTS applications is also a relevant issue with wireless and mobile ad hoc network environments, its importance in TENs is much more significant. As a result, we believe that this is an interesting research area that could lead to results with impact outside the military application environment. We hope that this article will attract the interest of researchers working in wireless communications and stimulate them to develop methodologies and solutions that push forward the state of the art.

2. COTS Applications and SOAs in Tactical Networks

The deployment of COTS and legacy applications in TENs presents many significant challenges related to the use of software architectures and communication protocols that are not suited for highly dynamic and unreliable networking environments. More specifically, the most significant problems exhibited by COTS applications originate from their reliance on SOAs and TCP.

SOA-based applications typically adopt client-server architectures with long-lived and (semi-)static bindings between software components. In fact, while the fundamental building block of SOAs is represented by short-lived operations, that is, synchronous Remote Procedure Calls (RPCs), usually SOA-based clients exploit information about server locations (e.g., obtained from WSDL documents) for several subsequent requests, thus leading to a relatively tight and brittle coupling between software components. This model is perfectly appropriate for LANs or wired Internet environments but that does not match the resilience and adaptivity requirements of TENs. In fact, applications running in TENs must deal with frequent disconnections for multiple reasons (nodes could move and fall out of communications range, networks may become partitioned, client devices may need to be temporarily shut down or stop network activity). In addition, TEN applications often need to switch to different service providers, exploiting new resources (e.g., Unmanned Aerial Vehicles, UAVs) as they come in topological proximity and, when possible, leveraging peer-to-peer (P2P) architectures and/or local replicas of data and services [3].

The coupling between components in SOA-based applications can be loosened via the adoption of enterprise message buses (EMBs) that also reduce the risk of building stove-piped systems. However, the reliance on COTS messaging systems and protocols results in applications that fall short of matching the needs of TENs. In fact, COTS solutions such as JGroups (http://www.jgroups.org/) and Advanced Message Queuing Protocol (AMQP)-based middleware

being proposed and adopted in TENs were not designed to withstand temporary network disconnections between components without disrupting service sessions or to support the opportunistic exploitation of resources.

In addition, SOA-based technologies adopt application protocols, e.g., HTTP, and middleware that hinder the adoption of performance optimizations such as request aggregation and/or pipelining and reuse of trasport-layer connections. SOA-based applications also make heavy use of verbose and bandwidth-expensive XML-based data representation protocols that are an excessive burden for TENs.

The Marine Corps Systems Command's Marine Air Ground Task Force (MAGTF) Command & Control (C2) Systems & Applications (SA) Service-oriented Infrastructure (SOI) represents an illustrative example of how difficult it is to run SOA-based applications in TENs. SOI relies on the JBoss middleware and on various enterprise message buses. Applications typically integrate with SOI using Apache Qpid, a cross-platform Advanced Message Queueing Protocol (AMQP)-based messaging middleware, conveying all messages over TCP. Therefore, when running on a TEN, those applications will either not function or provide poor performance when communicating with SOI.

These considerations led to the development of SOA-specific solutions, such as DSProxy [4], that facilitate the deployment of SOA-based COTS applications in TENs by operating as adapters between the applications and military grade store-and-forward communication solutions, such as the X400-based STANAG 4406 messaging system, and techniques, e.g., request optimization and distributed caching, that improve the applications' fault-tolerance and performance.

However, while proxy-based adaptation has proved to be a very effective technique for enabling the deployment of SOA-based COTS application in military environments, SOA-specific approaches present several significant drawbacks. More specifically, SOA-specific adapters such as DSProxy do not consider non-SOA-based COTS and legacy applications and only focus on the optimization of application-level protocols.

Instead, there is the opportunity to improve the performance of COTS applications by applying communication optimization techniques that operate contextually at both the application and transport levels. This is particulary important in COTS applications that perform long service sessions and adopt transport-level communication semantics based on the reliable stream model provided by TCP.

The reliable stream model is not compatible with the "always exploit the best connection" operational mode, which is a fundamental requirement for TENs. In fact, nodes in a TEN often have more than one communication link; for example, a slow but reliable third generation (3G) connection, a SATCOM connection that may not work in bad weather, and/or a faster local connection (e.g., to an airborne relay node), which is not always available. To achieve the best performance, applications (and the middleware) need to dynamically switch between the

available links to exploit the best connectivity. However, TCP-based applications are not capable of dynamically switching service sessions between different network interfaces, thus forcing users to manually shut down service sessions and restart them in order to take advantage of faster (and cheaper) links when available. Note that while similar issues are also actively studied in Third Generation Project Partnership (3GPP) networks [5], the need to always exploit the best connection in TENs is an even more important and complex issue due to the wide heterogeneity of link types.

In fact, the mismatch between the semantics of transport-layer protocols upon which COTS applications are built and those that could be reasonably provided over TENs also represents a major issue. At the transport protocol level, COTS applications leverage the semantics proposed by commonly used protocols, such as TCP and UDP, which provide reliable and sequenced delivery of a stream of data and unreliable unsequenced delivery of messages, respectively.

TCP and similar protocols are also very inefficient in highly dynamic environments such as TENs. In fact, TCP implements a single first in-first out (FIFO) transmission queue and does not enable applications to liberate the queue from stale data waiting for (re)transmission (an operation that the stream-oriented nature of TCP, which does not preserve the boundaries between messages, would also make rather difficult to implement). In cases where the information generation rate outpaces the network capability, this limitation forces the transmission of obsolete messages, significantly reducing the applications' goodput. The negative impact of the TCP queuing model on latency is also well recognized also in wired Internet environments [6]. However, these issues are further exacerbated in TEN applications by the time-sensitive and multiple-priority nature of traffic, and by the adoption of peculiar and high-latency communication solutions, such as tactical radio links with DAMA modulation that implement two-party communications over unidirectional links.

In addition, in wireless environments, TCP implementations often misinterpret packet losses (caused by higher channel error-rate, medium contention/collision, and/or node mobility) as congestion symptoms and consequently trigger congestion control mechanisms that significantly reduce the transmission rate. This approach leaves the wireless channel underutilized and generates traffic flows with lower throughput and higher latency than the network is actually capable of delivering.

At the other end of the spectrum, UDP does not provide reliability levels or flexible mechanisms for group communications suited for TEN communications. In fact, the best-effort delivery semantics implemented by UDP places the burden of ensuring the delivery of important messages on the applications, for instance by implementing ad hoc retransmission schemes. UDP broadcast communications are also inadequate as a basic mechanism for group communication in TENs, which instead call for disruption-tolerant communication schemes.

The limitations of TCP and UDP have a major impact on COTS and legacy tactical applications that typically leverage TCP for reliable end-to-end communication and UDP for best effort

broadcast/multicast communications. An example of these legacy applications is GeoChat, a multi-user chat application that is part of the Air Force Special Operations Command's Battlefield Air Operations (BAO) Kit. GeoChat uses TCP for file exchange, incurring relatively frequent failures and low performance, and UDP multicast for chat messages, incurring frequent loss of messages.

Instead, TEN applications are better served by communication solutions that provide a wider range of delivery semantics, thus enabling the differentiation of delivery mechanisms according to the importance of the messages being transmitted. In particular, reliable and sequenced delivery (as provided by TCP) is very expensive and should be used only when absolutely necessary. The importance of providing customizable delivery mechanisms in wireless environments is widely recognized, as demonstrated by recent research efforts on SCTP, which supports multiple streams, multiple associations, and partial reliability mechanisms [7]. In fact, state-of-the-art TEN-specific communication solutions recognize the need for smart buffer management and go beyond the features provided by SCTP by also enabling fine-grained control of message delivery semantics and mobile service sessions [1].

Finally, the limited bandwidth available for communications in the tactical environment requires applications to adopt communication schemes that are as efficient as possible. However, COTS applications were typically designed for wired Internet environments with the assumptions of having frequent and evenly distributed transmission opportunities and low round-trip times. As a result, they often implement rather simple communications schemes that do not adopt any optimization, for example, by aggregating or parallelizing requests or reusing already established connections for following requests [8]. These assumptions do not hold in TENs, thus leading to very poor performance results.

3. Bridging the Impedance Mismatch

Running COTS applications on top of communication solutions specifically designed for TENs is often impossible or impractical, as it requires modifications to the applications' source code. Clearly, this approach cannot be applied to third party or legacy software, because either the applications' source code is not available or the required modifications would be too expensive.

Instead, an interesting approach relies on the development of specific adaptation components or middleware to enable the deployment of COTS and legacy applications over TEN-specific communication solutions. This approach follows a school of research, dating as far back as 1995, with proposals such as I-TCP [9], Mobile-TCP [10], and the Remote Sockets Architecture [11], which investigates proxy-based architectures to address both the performance and mobility issues that TCP exhibits in wireless networks. More recent proposals, such as A^3[12], suggest the adoption of transparent proxies and sophisticated buffer and message management solutions to accelerate TCP-based applications in wireless environments.

Despite their interesting potential, so far proxy-based solutions have mostly focused on the application performance improvements instead of on the adaptation between different communication models and/or protocols. In addition, those solutions have received limited interest from researchers, who instead have preferred to focus on the development of new protocols or wireless-friendly TCP implementations. However, the deployment of COTS and legacy applications on TENs calls for an intelligent substrate that lies between COTS applications and TEN-specific communication solutions, thus making proxy-based solutions a key technology to bridge the impedance mismatch between these two software layers.

More specifically, the adaptation substrate should use proxy components to implement the transparent remapping of traditional TCP- (or UDP-) based communication semantics, adopted by COTS applications, to TEN communication middleware, e.g., for information dissemination or for mobile sessions support. At the receiver side, another proxy instance should translate the received data back to the applications through a TCP- (or UDP-) based interface.

The proxy-based approach has the advantage of being application-transparent. It works with COTS applications without requiring any modifications and without breaking the expected TCP- or UDP-based communication semantics. It also does not leak any abstraction or concept of TEN-specific communication solutions to the higher software layers. By providing adaptation features at the proxy level, all applications can immediately benefit from the functions provided by communication solutions explicitly designed for TENs without any modification.

At the same time, the proxy-based approach enables the realization of an adaptation substrate that is both application-aware and network-aware, which can therefore put in place specific solutions to optimize communications according to the specific COTS application requirements (or semantics) and the current operating conditions. For instance, for some applications, it could leverage peer-to-peer (P2P) communications or opportunistic information dissemination solutions. For other applications, it could tailor the communications according to the current state of the network, prioritizing the transmission of essential messages and discarding (or deprioritizing) messages of secondary importance according to the available bandwidth.

Therefore, the proxy-based approach represents a very effective solution that is particularly well suited to support the deployment of COTS and legacy applications in TENs.

4. The Agile Computing Middleware NetProxy Solution

Following the approach described in the previous Section, we designed NetProxy, a solution that transparently intercepts any (TCP- or UDP-based) network traffic generated by COTS applications, analyzes it, and conveys it over point-to-point connections and/or point-to-multipoint information dissemination channels provided by the Mockets and DisService communication middlewares.

The Mockets middleware is a communication solution explicitly designed to address the issues that the standard communication protocols, such as TCP and UDP, exhibit on TENs. Mockets enables the smart management of transmission buffers, allowing applications to exploit different delivery semantics and transmission priorities for different classes of messages, discard obsolete data in the transmission queue, and so on. In addition, Mockets supports session mobility and dynamic service rebinding, and implements mechanisms that monitor the current network state and export that state to the applications, enabling them to detect connection loss, monitor network performance, and react accordingly (e.g., by tailoring their Quality of Service). Unlike TCP, the Mockets middleware does not assume to operate over low-error rate channels, but implements an adaptable congestion control mechanism that is specifically devised for the challenges that are characteristic of the mobile ad hoc environment. Additional details on the improved performance of Mockets on TENs are described in [1]. The challenge, addressed by the NetProxy described in this article, is enabling COTS and Legacy applications to benefit from Mockets withouth having to modify them.

DisService, on the other hand, is a P2P information dissemination solution specifically designed to support dynamic network topologies and highly mobile nodes such as UAVs [3, 13]. DisService enables nodes to participate in multiple groups of interest, which communicate through independent information dissemination channels (subscriptions), thereby supporting the multiple patterns of data dissemination required by tactical applications. DisService enables disruption-tolerant information dissemination and relies on store-and-forward communications, aggressive distributed data caching, and opportunistic resource (communications, storage, and processing capacity) management to improve the performance of the information dissemination process. DisService also implements an adaptive contact prediction mechanism that detects recurrent mobility patterns of highly dynamic nodes, such as UAVs loitering over the battlefield, and uses this knowledge to optimize the information dissemination process.

Mockets and DisService represent complementary solutions that address a large number of the communication requirements in TEN environments. NetProxy, whose architecture is depicted in Figure 1, relies on Mockets and DisService to provide COTS applications with communication solutions well suited for TENs. Mockets, DisService, and NetProxy are components of the Agile Computing Middleware.

More specifically, NetProxy operates transparently to COTS applications by capturing their TCP and UDP packets, extracting their payload, processing the data according to the application-specific traffic management configuration, and handing them over to Mockets or DisService for efficient delivery to the destination. At the receiver side, another instance of NetProxy performs the inverse task. NetProxy also supports temporary disconnection, stream compression, and network activity logging.

While the proxy-based approach introduces some overhead, our experience demonstrates that the performance gains stemming from more efficient and target-appropriate protocols significantly outweigh the computational burden and lead to major performance improvements

overall. In addition, NetProxy enables devising sophisticated application-specific optimizations that can improve the performance of COTS applications even further.

4.1. Conveying COTS Applications' Traffic over TEN-Specific Communication Solutions

To optimize the performance of COTS applications on TENs, NetProxy provides traffic manipulation functions and enables implementation of user-defined policies for the smart management of traffic. This allows addressing the many requirements that COTS applications might have.

More specifically, users can configure NetProxy with policies that target specific communications (e.g., depending on the type of traffic, the protocol being utilized, or the source/destination addresses). Policies can be dynamically updated at runtime, without stopping and restarting running applications. This provides a standard and centralized configuration point for all the functionalities provided by NetProxy, allowing for faster and easier management of the communication flows in the network.

NetProxy supports many different traffic manipulation functions. For instance, NetProxy allows the forwarding of traffic over a Mockets service session, a DisService subscription, or both. This might be suited for COTS applications that issue notifications to a server, as it helps to disseminate information to other nodes that might be interested.

In addition, by leveraging the session mobility feature provided by Mockets, NetProxy can transparently rebind local service session endpoints to a different network interface, taking advantage of faster (and cheaper) links when available. This addresses the issues of deployment scenarios, commonly found in TENs, where nodes have several links (e.g., SATCOM, 3G, airborne relay, and/or Wi-Fi) and need to switch between them to exploit the best possible connectivity.

NetProxy also provides application-specific mechanisms that significantly improve the applications' robustness and performance through relatively straightforward reconfigurations or modifications. In fact, while for some applications the simple adoption of generic solutions (e.g., standard data compression mechanisms) will deliver significant performance improvements, other applications might require specific and more sophisticated schemes.

NetProxy enables the performance optimization of important applications through the definition and implementation of application-specific management policies/configurations and the development of dedicated acceleration plugins. Plugins have to provide a set of rules that enable NetProxy to identify the traffic belonging to the applications for which they are designed and to implement a simple callback interface that enables them to integrate with the NetProxy traffic manipulation function.

A particularly interesting optimization technique involves the adoption of application-specific transcoding and compression schemes. For instance, it is often possible to reduce the size of request and response messages of COTS applications that rely heavily on XML by switching to more efficient data representation and document formats, such as JavaScript Object Notation (JSON). In addition, compression of message content before the actual transmission is often a very effective way to reduce bandwidth consumption. However, some applications cannot adopt generic full-payload compression schemes. This is especially true with HTTP-based applications that often have to traverse middleboxes such as load balancers that operate on the assumption that HTTP headers are readable and quickly accessible. The adoption of generic compression schemes would break these distributed application setups, which instead call for application- (or, more precisely, protocol-) specific compression schemes. In addition, compression schemes need to be deployed on a case by case basis, as sometimes they might increase latency and, with resource constrained clients, may be computationally too expensive.

Another very compelling application-specific optimization involves request optimization. By developing application-specific plugins that leverage knowledge about the communication semantics of COTS applications, it is possible to optimize the applications' performance by automating the transmission of requests that the COTS application is likely to perform, or implementing request desequencing/parallelization schemes.

Application-specific plugins could also be used to enhance the applications' resilience and fault-tolerance. For instance, applicable plugins could act as adapters that provide a more robust service interface in front of existing services. This could turn stateful interactions in a series of idempotent operations, which are widely recognized as a fundamental tool to implement resilient services [14].

In addition, NetProxy enables sophisticated network-aware traffic management policies that permit enforcing different delivery mechanisms and priorities according to the specific type of traffic and the current network condition. For instance, if the quality of the communication link degrades, NetProxy can be configured to enforce reliable delivery only for critically important messages while transmitting all the other messages in a best-effort fashion, or discarding them after the expiration of the lifetime or maximum retransmission count attribute associated with the messages. When the network conditions improve, NetProxy might resume using reliable delivery for each message that is transmitted.

Finally, NetProxy facilitates the reuse of existing connections for multiplexing multiple connections at the application level. While this might seem a rather straightforward optimization, in our experience it often leads to significant performance optimization, as demonstrated by the first experiment presented in the next Section.

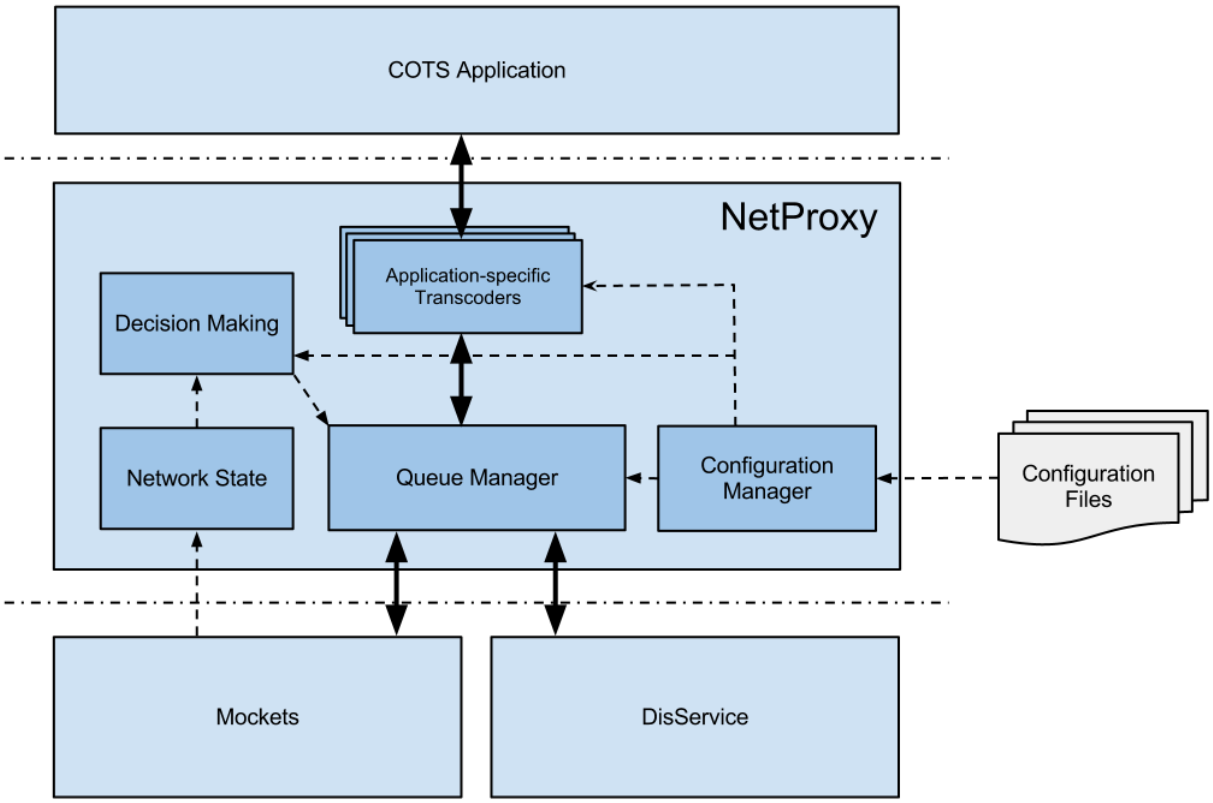

Figure 1. NetProxy architecture and interface with applications and lower level-communication solutions.

5. Experimental Results

To demonstrate the effectiveness of proxy-based solutions to support the deployment of COTS applications in TENs, we present the results obtained from two different experiments. The experiments are designed to reproduce and illustrate common issues that COTS applications exhibit in TENs.

To evaluate the performance of the combined use of the NetProxy and Mockets middleware, and to compare this solution to TCP, we ran the experiments in an emulated environment, which allowed us to reproduce the characteristics of a TEN. More specifically, we used an enhanced version of the Mobile Ad-hoc Network Emulator (MANE) [15], a tool designed to reproduce the characteristics of unreliable environments such as TENs, to set up the connectivity between the nodes involved in the testing.

The nodes are part of the NOMADS testbed, which comprises 96 servers connected through a 100Mbps Ethernet LAN. The hardware configuration of the machines consists of HP DL140 Servers (Dual Xeon Dual Core CPUs at 3.06Ghz, with 4GB of RAM). MANE can control bandwidth, latency, and reliability for each link, thus allowing the evaluation of different systems

and protocols in a reproducible, laboratory controlled environment. The reliability parameter is based on the packet error rate (PER); hence, a 90 percent reliability value is equal to a 10 percent PER value.

For the first experiment, we wrote a simple client application that generates an HTTP SOAP request, sends it to a Web Server located on another node of the testbed, and waits for the response. The application was also responsible for measuring the throughput. We kept the bandwidth of the link stable at the value of 1 Mb/s throughout the experiment, while we considered the reliability values of 87, 90, 93 and 95 percent. We configured the client application to repeat the request 50 times with the same link conditions before we modified the configuration of MANE to vary the reliability of the link.

Figure 2 shows the results obtained by running the experiment described above connecting client and server using TCP, TCP via NetProxy, or Mockets via NetProxy. The higher throughput achieved by Mockets and NetProxy clearly stands out from the graph. It is also worth noting that TCP via NetProxy performs better than plain TCP. The reasons of this behavior are manyfold. First of all, many standard SOAP implementations, such as Apache Axis 2 (http://axis.apache.org/axis2/java/core/), by default send every SOAP request using a separate TCP connection. This means that a large portion of the traffic is sent during the TCP slow start phase, which significantly limits bandwidth usage. This issue does not occur with NetProxy, which multiplexes all the traffic directed to a single node onto the same connection, reusing it for following request. Furthermore, NetProxy has a buffering mechanism that allows the generation of larger TCP segments, therefore reducing the protocol overhead.

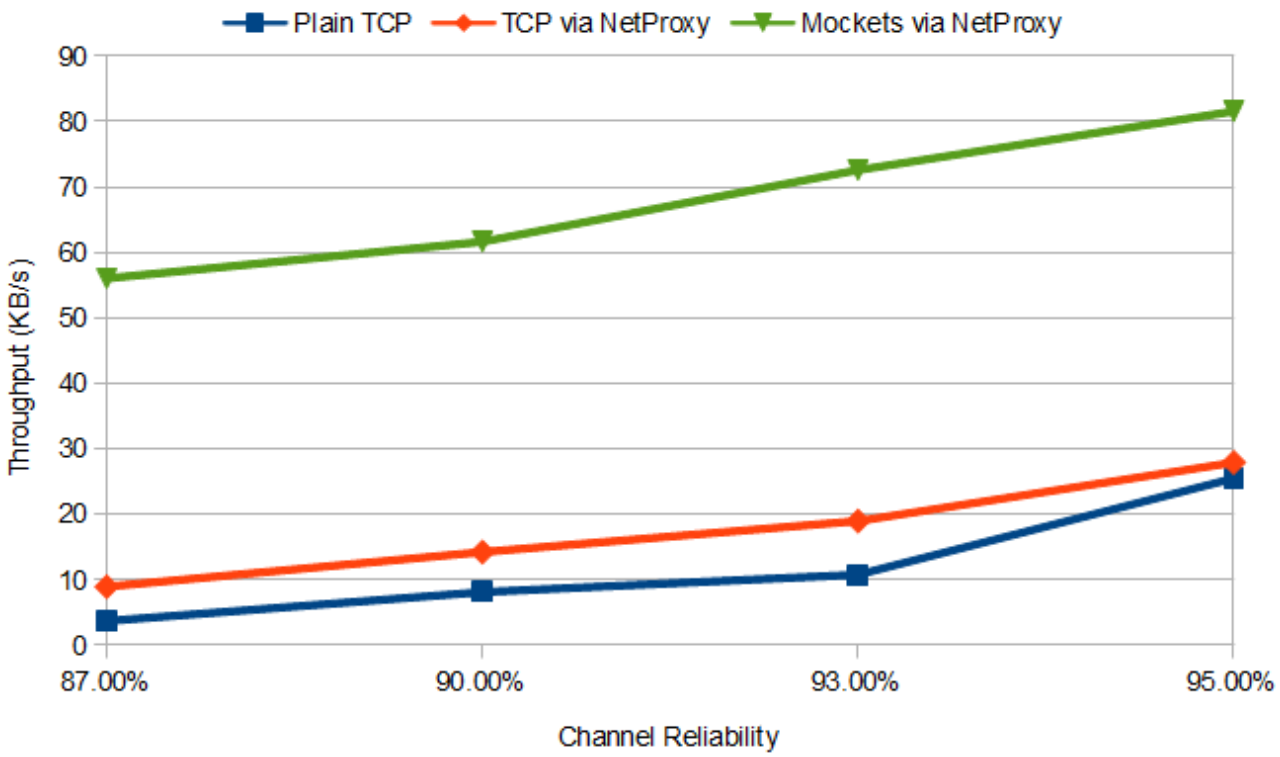

Figure 2. Measured throughput running the experiment with: a) TCP; b) TCP via NetProxy; c) Mockets via NetProxy.

Figure 3 shows the results obtained running the same experiment after enabling the compression feature of NetProxy. We used Mockets via NetProxy to send data between the two nodes while varying the compression algorithm. The figure shows that with compression enabled, we could achieve a very high gain in the measured throughput. This is due to the verbosity of the HTTP and SOAP protocols, which can therefore be compressed very efficiently. Despite the better compression ratio of LZMA compared to Zlib, the use of Zlib resulted in higher throughput. This is due to the higher demands on computational resources required by LZMA.

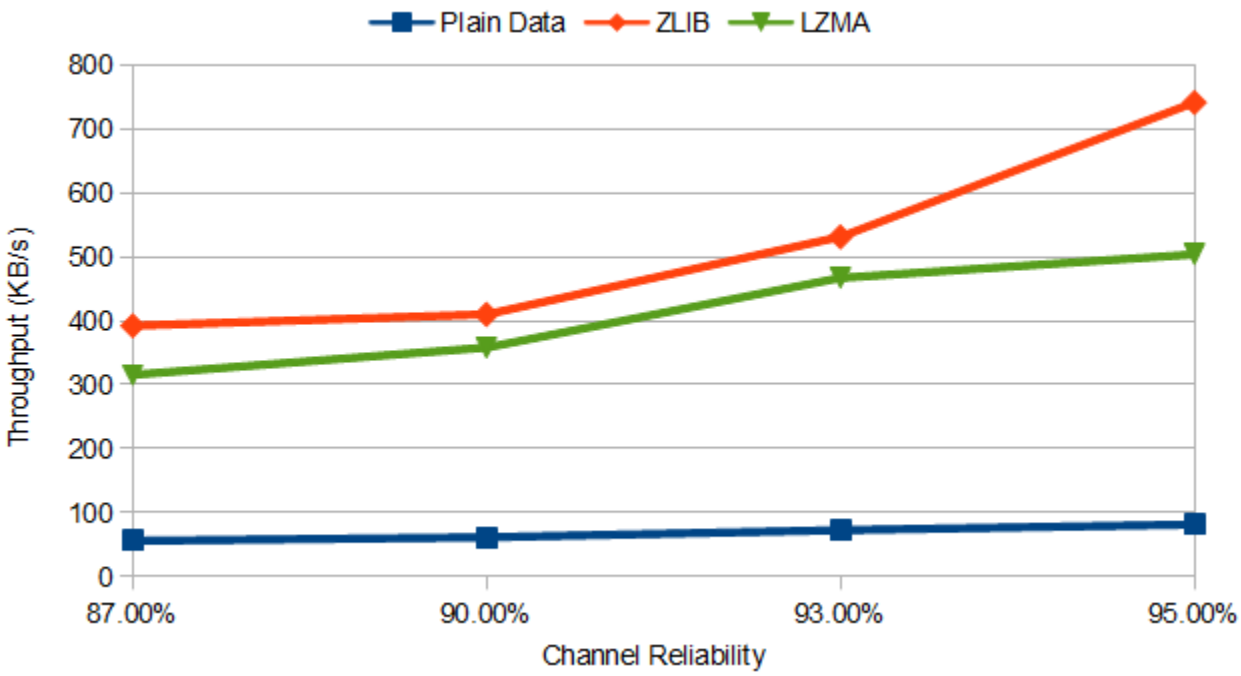

Figure 3. Measured throughput running the experiment wih Mockets via NetProxy and: a) compression disabled; b) Zlib compression; c) LZMA compression.

For the second experiment, we used Apache Qpid (http://qpid.apache.org/) to build a basic publish-subscribe architecture. Figure 4 presents the configuration we used for the experiment. Two instances of Qpid (Qpid A and Qpid B) were running on two different nodes of the testbed, and two applications, a publisher and a subscriber, were running on a third node (the publisher and subscriber were located on the same machine to enable accurate time measurements). The subscriber registered itself to Qpid B, while the publisher published messages to Qpid A. To dispatch the published messages to the subscriber, Qpid A was connected to Qpid B. We configured the connections between the publisher and the subscriber applications and the relative Qpid instances to use TCP over the 100Mb/s Ethernet LAN provided by the NOMADS testbed, while we used MANE to configure the link between the two nodes running the Qpid instances. The nodes we used in this experiment were CentOS 6.2 Linux virtual machines (VM), each VM running on an Ubuntu 8.04 LTS server. Again, we fixed the link bandwidth to 1Mb/s using MANE, and then repeated the experiment with the values 85, 90 and 95 percent for the link reliability parameter. To collect the results, the publisher published 10 copies of messages of 100, 256 and 512 kbytes for each reliability setting, and then measured the throughput.

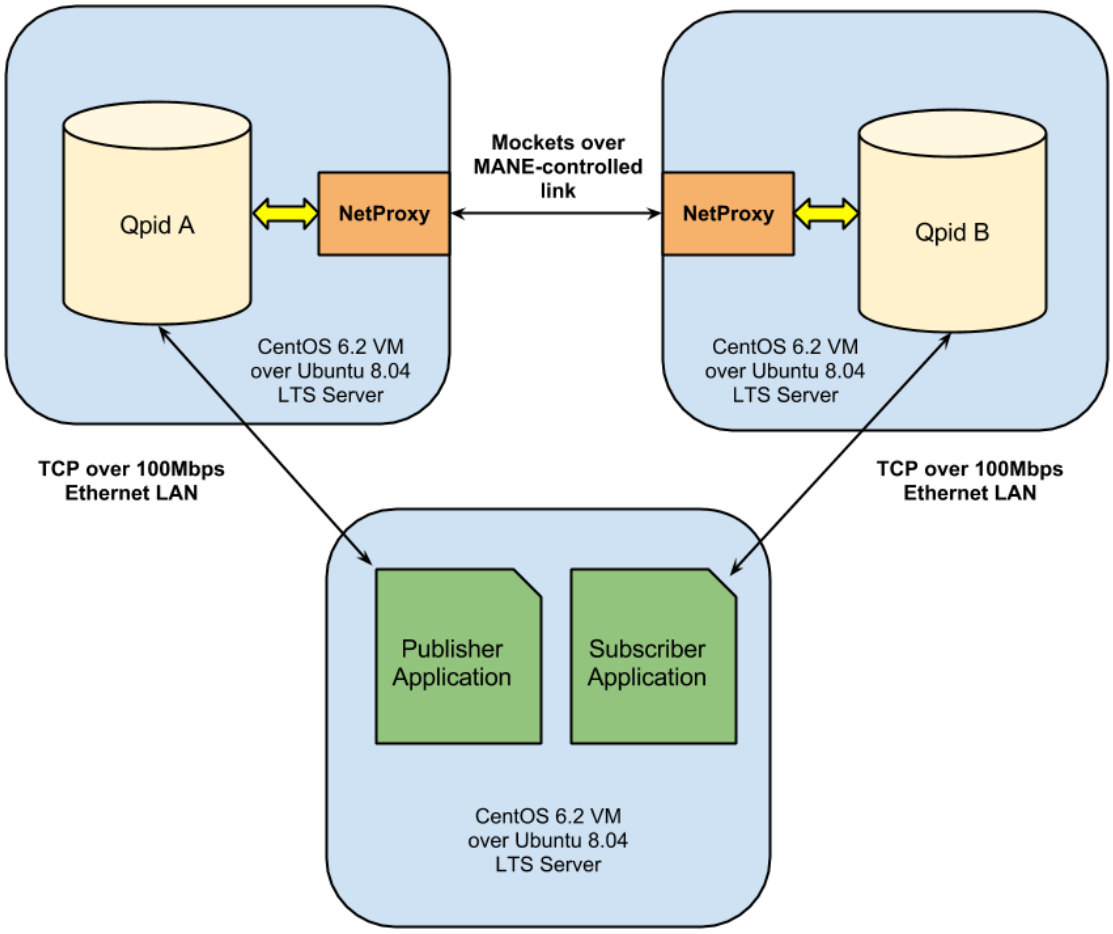

Figure 4. Configuration for the second experiment.

Figure 5 shows the results of the second experiment. They are divided into three charts, based on the message size, which graph the average measured throughput. The results demonstrate that, while the performance of both the solutions decreases with the decrease of the link reliability, the decreasing trend of TCP is much steeper than with NetProxy and Mockets. Moreover, in almost all the tests, the standard deviation measured when relying on Mockets and NetProxy to connect the nodes was lower than the standard deviation measured when TCP was used. The results also show that the transport protocol implemented by Mockets still has room for improvements when employed over reliable links. However, our past experience with the use of Mockets in real TEN scenarios has always exhibited significant improvements in terms of goodput, latency, and temporary connection disruption tolerance when compared to traditional TCP-based solutions. We found that the reason lay in the relatively high number of packet retransmissions occurring during the experiments, which corresponds to an average PER level higher than 10 percent.

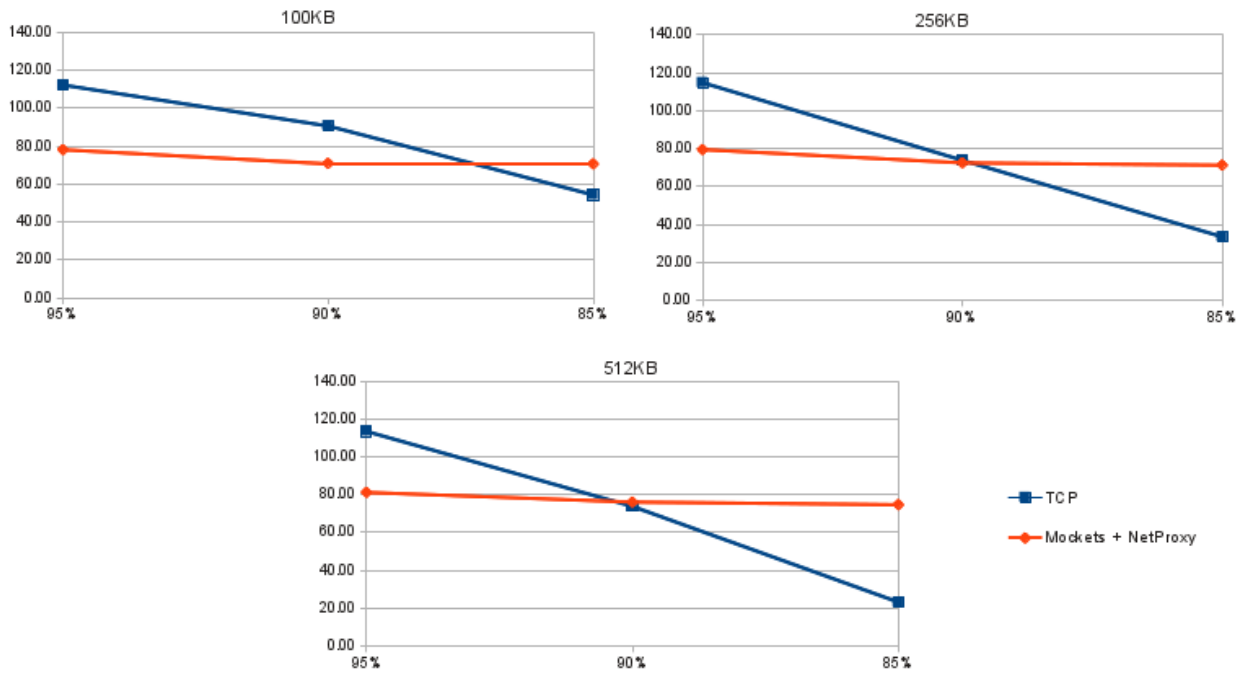

Fig. 5. Average measured throughput (in kilobytes per second) for messages of different sizes: a) 100 kbytes; b) 256 kbytes; c) 512 kbytes; and channel reliability set to 95, 90, and 85 percent. Reported results refer to tests run with TCP and Mockets via NetProxy.

6. Conclusions

Proxy-based technologies represent an interesting and very effective approach to enable the deployment of COTS applications in TENs. The potential of proxies in increasing the applications' robustness and performance is so significant that their adoption deserves to be investigated beyond TEN environments.

The diffusion of proxy-based technologies could open up very interesting scenarios. First, it could lead to a world in which SOA- and TCP-based interfaces are perceived by developers as a commodity. Applications will be built on top of them and engineers will use proxies to deploy those applications anywhere.

In addition, tools like NetProxy enable the management of the security-related configuration of several applications in a single place, at the proxy level. This facilitates the deployment of not secure-by-design COTS applications in security-critical environments such as TENs.

Proxy-based adapters could open the way for experimental development tools for SOA-based applications that automate application-specific traffic management or that support the development of dedicated plugins for proxy-based solutions. For instance, such tools could leverage additional (semantic) information inserted in WSDL documents via annotation schemes. This represents a very promising research direction that we are planning to explore within the NetProxy project in the near future.

We are also planning to test the NetProxy dynamic (re-)configuration capabilities and to evaluate its performance with additional commonly used COTS applications. These results will be reported in future publications.

References

[1] N. Suri, E. Benvegnù, 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 (Special Issue on Military Communications), pp. 56-63, October 2009.

[2] A. Morelli, R. Kohler, C. Stefanelli, N. Suri, M. Tortonesi, "Supporting COTS Applications in Tactical Edge Networks", in Proceedings of IEEE MILCOM 2012 Military Communications Conference, 29 October - 1 November 2012, Orlando, FL, USA.

[3] N. Suri, G. Benincasa, M. Tortonesi, C.Stefanelli, J. Kovach, R. Winkler, R. Kohler, J. Hanna, L. Pochet, S. Watson, "Peer-to-Peer Communications for Tactical Environments: Observations, Requirements, and Experiences", *IEEE Communications Magazine*, Vol. 48, No. 10 (Special Issue on Military Communications), pp. 60-69, October 2010.

[4] K. Lund, E. Skjervold, F.T. Johnsen, T. Hafsøe, A. Eggen, “Robust web services in heterogeneous military networks”, *IEEE Communications Magazine*, Vol. 48, No. 10 (Special Issue on Military Communications), pp. 78-83, October 2010.

[5] M. Louta, P. Bellavista, “Bringing Always Best Connectivity Vision a Step Closer: Challenges and Perspectives”, *IEEE Communications Magazine*, Vol. 51, No. 2, pp. 158-166, February 2013.

[6] A. Erbad, C. Krasic, “Sender-side Buffers and the Case for Multimedia Adaptation”, *ACM Queue*, Vol. 10, No. 10, October 2012.

[7] Ł. Budzisz, J. Garcia, A. Brunstrom, R. Ferrus, “A Taxonomy and Survey of SCTP Research”, *ACM Computing Surveys*, Vol. 44, No. 4, Article 18, August 2012.

[8] K. Fall, S. McCanne, “You Don't Know Jack about Network Performance”, *ACM Queue*, Vol. 3, No. 4, pp. 54-59, May 2005.

[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] 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.

[13] M. Tortonesi, C. Stefanelli, E. Benvegnù, K. Ford, N. Suri, M. Linderman, "Multiple-UAV Coordination and Communications in Tactical Edge Networks", *IEEE Communications Magazine*, Vol. 50, No. 10 (Special Issue on Military Communications), pp. 48-55, October 2012.

[14] P. Helland, “Idempotence Is Not a Medical Condition”, *ACM Queue*, Vol. 10, No. 4, April 2012.

[15] Mobile ad-hoc network emulator (MANE), available at: http://cs.itd.nrl.navy.mil/work/mane/index.php

Biographies

RALPH KOHLER, JR. [F] (Ralph.Kohler@rl.af.mil) is a principal engineer at the U.S. Air Force Research Laboratory (AFRL), where his primary research interests are the nexus of information management, wireless and tactical networks, and the capabilities that combining the two make possible. He received his Master’s degree from Syracuse University and a member of the AIAA.

ALESSANDRO MORELLI (alessandro.morelli@unife.it) received his B.Sc. degree in Information Engineering and M.Sc. degree in computer science engineering from the University of Ferrara, Italy. He is currently a Ph.D. student in the Engineering Department of the University of Ferrara. His research interests focus on distributed and mobile computing, opportunistic networking, and smart cities.

CESARE STEFANELLI [M] (cesare.stefanelli@unife.it) received his Laurea degree in electronic engineering and his Ph.D. in computer science engineering from the University of Bologna, Italy. He is currently a full professor of computer science engineering in the Engineering Department of the University of Ferrara. His research interests include distributed and mobile computing, adaptive and distributed multimedia systems, network and systems management, and network security.

NIRANJAN SURI [M] (nsuri@ihmc.us) is a research scientist at the Florida Institute for Human & Machine Cognition (IHMC) and also a visiting scientist at the U.S. Army Research Laboratory, Adelphi, Maryland. He received his Ph.D. in computer science from Lancaster University, England, and his M.Sc. and B.Sc. in Computer Science from the University of West Florida, Pensacola. His current research activity is focused on the notion of agile computing, which supports the opportunistic discovery and exploitation of resources in highly dynamic networked environments. His other research interests include coordination algorithms, distributed systems, networking, communication protocols, virtual machines, and software agents.

MAURO TORTONESI [M] (mauro.tortonesi@unife.it) received his Laurea degree in electronic engineering and his Ph.D. in computer science engineering from the University of Ferrara, Italy. Currently, he is an assistant professor in the Engineering Department of the University of Ferrara. His research interests include distributed and mobile computing, QoS management, network and service management, business-driven IT management, and e-maintenance.

SCOTT WATSON (scott.c.watson@navy.mil) is a senior systems architect with the Composeable Services Branch in the Command and Control Department of the Space and Naval Warfare Systems Center - Pacific. He has over 10 years’ experience designing software

systems and applications. He is currently the principal investigator for the Mobile Modular Command and Control for Company level operations (M2C3) project. He has a B.S. in computer science from San Diego State University, California. He recently received the Navy Civilian Meritorious Service award for work in the area of disruption-tolerant networking.