Università degli Studi di Ferrara

DOTTORATO DI RICERCA IN
"Scienze dell'Ingegneria"

CICLO XXVIII

COORDINATORE Prof. Stefano Trillo

Communications Middleware for Challenging Networking Scenarios

Settore Scientifico Disciplinare ING-INF/05

Dottorando
Dott. Alessandro Morelli

Tutore
Chiar.mo Prof. Cesare Stefanelli

Cotutore
Prof. Mauro Tortonesi

Anni 2013/2015

Table of Contents

INTRODUCTION

Since its creation under the name of ARPANET in 1969, the Internet has gone through a number of radical innovations and revolutions that completely changed the network infrastructure and the way people communicate, access, and use the information and services provided. From a system of just two machines set up for research purposes, the Internet grew to connect millions of nodes, pushed by the launch of the World Wide Web and its immense impact on commerce. Today, the availability of diverse wireless connectivity technologies, the pervasiveness of mobile devices that enable users to be always connected, the rise of a number of services that more and more people each day consider essential for their lives, and the possibilities that these new technologies and services are opening in other fields are changing, once again, the Internet substantially. These new technologies, services, and opportunities, however, are also raising new requirements, for which traditional communication solutions, originally devised for static and fixed network infrastructures, are proving inadequate.

During the same years of the rise of the Internet, the increasing number of people that populates the cities around the world has posed severe challenges to the ability of providing services to the citizens [1]. The concept of smart city has emerged to address those challenges and it refers to the use of Information and Communication Technologies (ICT) to provide the population with access to the city assets [2]. At the same time, the pervasiveness among the population of smartphones and tablets make them extremely cheap resources and valuable sources of environmental data for the public administration [3]. In fact, today's mobile smart devices are equipped with various sensors and heterogeneous network connectivity technologies that permit data gathering at the edge of the network and successive transfer to the cloud for storage, analysis, and information extraction.

The efficient gathering, processing, and delivery/dissemination of data require the design of effective solutions that take into account resource availability on the nodes, the network's characteristics, and users' preferences. Nodes in smart cities range from very small, low-power sensors, such as CO2 level detectors, battery powered devices with enough resources to perform some basic data processing, e.g., smartphones and tablets, up to powerful servers on the cloud. The network is highly heterogeneous, where the coexistence of both wired and wireless solutions provides multiple connectivity choices and raises the opportunity to exploit device-to-device (D2D) communication techniques to promote the offloading of particularly congested parts of the network, reduce the usage of expensive connectivity solutions, and tackle network partitioning [4] [5]. Finally, the users might put

restrictions on data collection, processing, and transmission, or switch off some or all network interfaces on their devices, in order to reduce memory consumption and save battery life.

Heterogeneous wireless networks, nodes' mobility, D2D communication techniques, and the cooperation between devices with diverse resource availability are the fundamental pieces of *next-generation networking (NGN) scenarios* [6]. On top of them, also the emergence of novel applications built on the extensive use of location- and social-based features, which enable a strong interaction with the environment and the territory [3], but also with nearby people and friends, characterizes the NGN environment. Such scenarios are growing more and more common today, and well describe the networking environment of smart cities.

Wireless networking solutions based on ad hoc and D2D communications that do not require a network infrastructure, the presence of devices with heterogeneous resources, and strong node mobility also lay the basis for the "Network-centric Warfare" concept [7], a revolution in both the fields of military warfare and disaster recovery. T*actical Edge Networks* (TENs) are the fundamental piece of those scenarios and identify a networking environment that has many common characteristics and faces several common problems with NGN scenarios. In fact, TENs are normally composed of a combination of networks that goes from Mobile Ad-hoc Networks (MANETs) and Wireless Sensor Networks (WSN), up to the infrastructured networks (Local and Wide Area Network, respectively LAN and WAN) [8]. Different network types are usually connected to each other through intermittent, heterogeneous links, sometimes provided by highly mobile nodes that bring temporary connectivity to otherwise disconnected portions of the network. Thus, similarly to what happens in NGN scenarios, TENs commonly suffer network partitioning and necessitate solutions that can exploit heterogeneous connectivity.

Another common characteristic between next-generation and tactical networks is resource availability, which varies a lot depending on the nodes and the type of network: while LANs provide a very solid basis for node communications, in contrast MANETS and WSN typically offer very low bandwidth, variable latency, and are subject to frequent packet loss [9]. Moreover, network disruption and reconfiguration, caused by nodes' mobility, increase both the network churn rate and the frequency of link intermittency and makes end-to-end connections more prone to disruption [10]. As a result, applications that rely on traditional communication solutions, which assume continuous connectivity, suffer from performance issues and may fail frequently in TENs.

Moreover, in TENs there is a great interest towards the adoption of Commercial Off-The-Shelf (COTS) hardware and software solutions, the reuse of existing (legacy) applications, and the exploitation of Service-oriented Architectures (SoAs). In particular, SoAs enable the building of extensible and scalable services for tactical scenarios that can be easily reconfigured to adapt to changes in the mission and respond quickly to the enemy's actions [11] [12] [13]. At the same time, the adoption of COTS software and the reuse of legacy applications enable reaping the benefits of economies of scale and facilitate both the development and deployment of complex distributed applications. However, these technologies were, and most times still are, developed using communication standards, like TCP, that were devised for the wired Internet. At the same time, SoA-based technologies make heavy use of verbose and bandwidth-expensive XML-based data representation protocols. The resulting high bandwidth demands and the inability to cope with link disruptions that follow from such technologies do not suit TENs.

The characteristics of NGN scenarios and TENs and the inadequacy of protocols like TCP and UDP, which typically exhibit poor performance and unsuited communication semantics in mobile and wireless networks, highlight the need for solutions to support applications running in such challenging scenarios. Therefore, it has become extremely important to study and devise new communication paradigms that can handle better the set of problems arising in those scenarios and satisfy application requirements. In this sense, the research on the fields of *Opportunistic Networking* and *Information-centric Networking* (ICN) seems very promising. The Opportunistic Networking concept supports communications in highly dynamic scenarios by taking advantage of contact opportunities between nodes, store-and-forward techniques, and knowledge on nodes' mobility patterns and human relations [14] [15] [16]. Differently, ICN shifts the focus of communication completely, moving from the location of information to information itself, so that any node in the network can become content provider [17] [18].

Besides designing novel communication paradigms, features like seamless handoff, awareness of network conditions, location, and user's social relations, and prediction of future resource availability and user's requests will also be crucial to support next-generation and tactical applications. Moreover, the efficient functioning of future smart cities calls for effective ways to engage private citizens in the process of data collection and dissemination. Finally, in order to enable deployment and reuse of COTS, legacy, and SoA-based applications in tactical environments, there is the need to develop solutions that can mediate between the communication semantics required by the applications and those that can be reasonably supported by TENs [19].

The development of a communications middleware specifically designed for challenging networking environments and that provides a rich set of functionalities to support applications development represents a very interesting approach to address the problems arising in these scenarios. Network-aware middleware solutions that take into consideration the characteristics of the mobile and tactical environment would support applications in implementing a continuous adaptation process of the communication function to reach a trade-off between their requirements and the current network conditions. This would allow developers to write applications based on new communication paradigms and network-aware programming models without having to rely on low-level, often not portable, and error-prone system calls and networking primitives. The Agile Computing Middleware (ACM) is a successful example of a communications middleware specifically designed to support the development of applications designed to run in extremely dynamic wireless networks, such as TENs [6] [20].

My contribution to the state-of-the-art of scientific research tackles the problems described above from two directions. First, I focused on addressing the problem of enabling the (re)use of COTS, legacy, and SoA-based applications in TENs. Stemming from the observation that running those applications on top of middleware solutions is often impractical or even impossible, as it would require to change the applications' code, an interesting approach consists in the development of specific adaptation components (or middleware) to enable the deployment of COTS and legacy applications and SoA-based services over TEN-specific communication solutions [11] [19] [20]. More specifically, my solution proposes the use of application-transparent proxy components to remap TCP- and UDP-based communications to communication middleware specifically designed to support applications in TENs.

The result of this research effort is the *Agile Computing Middleware NetProxy*, the component of the ACM that bridges the gap between tactical applications and the middleware. NetProxy supports the remapping of TCP and UDP communications over the components of the ACM that can best satisfy applications' requirements in TENs. Besides protocol remapping, NetProxy also provides support for temporary disconnections and link disruptions, stream compression, intelligent buffering, traffic filtering and forwarding, packets consolidation, flow prioritization, connection multiplexing, and network activity logging. Finally, NetProxy supports two different operational modes, namely Host Mode and Gateway Mode, to match an assorted set of requirements, restrictions, and network configurations. Experimental results show that NetProxy can make a more efficient use of the available bandwidth, reduce its consumption, and smooth the shape of network traffic.

Moreover, exploiting protocol remapping, NetProxy can improve resilience to link disruption and enable remote service invocation over lossy links.

My research areas of interest also focus on the intelligent resource management in smart cities and other NGN scenarios. In this area, I developed a middleware called *ICeDiM* that combines features from the research in the fields of Opportunistic and Information-centric Networking. ICeDiM takes advantage of heterogeneous connectivity solutions and in-network caching to optimize the usage of the scarce network resources and ultimately increase the network performance. My research work on ICeDiM also led to the definition of Application-level Dissemination Channels (ADC), a concept that aims at engaging users in sharing their devices' resources while also defining rules that help in managing resource sharing at the middleware level. ICeDiM was tested in an advanced simulation environment and results show that it provides high message delivery ratios when compared to other solutions from the scientific literature on Opportunistic Networking. Additionally, ICeDiM can keep overhead and latency in the network under control.

Still on the topic of optimizing resource management in NGN scenarios, I worked on an extension of the DisService component of the ACM that leverages techniques from the research on Opportunistic Networking. The goal of my work is to enable the prediction of future contacts with resource-rich nodes ("message ferries") that could become an alternative way to deliver messages to destination. This extension of DisService builds predictions by analyzing the history of past contacts with other nodes and applying a mathematical approach to discover complex periodically recurring patterns. Then, the prediction of future contacts rests on the repetition of those patterns. DisService takes advantage of this knowledge to promote the offloading of the cellular network: if a prediction about an approaching "message ferry" node is available, DisService can choose to wait for its arrival and hand packets over to that node using an ad hoc link, e.g., Bluetooth or Wi-Fi, instead of transmitting the data via a 3G/4G link. Results obtained from experiments run in a simulated environments show that my solution can effectively reduce the amount of traffic delivered over the cellular network.

This Thesis is organized as follows. Chapter 1 sets the background by describing the characteristics of next-generation and tactical edge networks and the applications operating in those scenarios, on which my research activity focuses. Then, Chapter 2 discusses the issues that traditional networking solutions face in challenging networking environments, the characteristics of the information model of applications running in such scenarios, and eventually describes novel communication paradigms that better fit those environments. Chapter 3 introduces the middleware-based approach, of which the ACM is an example.

Chapter 4 describes NetProxy and I give details on its design, architecture, and implementation; in addition, the chapter presents experimental results obtained from three different experiments. Chapter 5 and Chapter 6 discuss my work on ICeDiM and the extension of DisService, respectively, and presents the results I obtained from several experiments in simulated scenarios. Chapter 7 discusses relevant related work that can be found in the literature. Finally, the last Chapter concludes the Thesis by summarizing my contribution to the state-of-the-art of the scientific research and discussing possible future research directions.

1. CHALLENGING NETWORKING SCENARIOS

The communications networks that modern military forces use to deliver mission-critical data to soldiers in the battlefield or during disaster recovery situations represent extremely challenging scenarios. Applications running in those environments need to cope with a multitude of problems that include frequent link disruption and packet loss due to interferences and the enemy's activity, network partitioning and elevated churn rate caused by nodes' mobility, heterogeneous connectivity technologies, and limited resources. Similarly, the networking scenario that will arise in the next future in smart urban and building environments also involves many obstacles to communications between applications. In fact, also in these environments, heterogeneous nodes and connectivity technologies, frequent link disruption and packet loss due to interferences and fading, network partitioning due to nodes' mobility, and constrained resource availability on nodes, e.g., because of low processing power and memory equipped on sensor nodes, or because of users' preferences, which limit resource sharing on devices, will threaten the correct operation of applications and services.

Those two scenarios identify a very interesting area of research and provide a valuable opportunity to understand better the problematics of network communications in challenging networking environments. The outcome of this study lays the foundations for the development of solutions to improve network performance and hide the most critical issues and obstacles to communications from applications.

1.1. The Next-generation Networking Scenario

The number of people living in the cities worldwide has been in the rising trend since way before the Internet era, and studies state that the urban population will almost double by the middle of the 21st century [1]. This incessant growth places new challenges to the city management under many points of view. The concept of smart city has emerged to address these challenges, describing a modern urban environment where ICT becomes the main provider of means and techniques to effectively access and exploit the assets of a city, such as its social and economic capitals. Many actors are making a pervasive and intensive use of ICT techniques to realize effective and sustainable solutions that aim at improving the quality of life of the smart citizens in many different areas [21] [22] [2].

The efficient gathering, processing, and dissemination of data are essential to implement new services in dynamic and heterogeneous environments such as the smart city. In fact,

while smart cities will have a connected core where the collected data is stored and processed through cloud-based analytics [23], a large part of the value of their infrastructures will reside at the edge of the network, where a plethora of smart objects, sensors, vehicles, and people occasionally connect to the network through different communication technologies.

Modern cities already have a broad range of sensors scattered around their territory, e.g., traffic and speed cameras, temperature sensors, CO2 level sensors, noise measuring sensors, and photodetectors, which will likely widen further in the next years. At the same time, the ever-growing density of smartphones and tablets, their pervasiveness among the population, and their availability on the urban territory at no cost for the public administration make them extremely valuable sources of data. In fact, today's mobile smart devices are equipped with more and more sensors and have the necessary computational and memory resources to produce and collect various types of raw data [3].

In this context, the rapid evolution of innovative mobile and pervasive computing applications is contributing to change significantly the way people use the Internet [24]. Such applications, characterized by an extensive use of location- and social-based features and an ever-increasing interaction with the surrounding environment [3], contribute to the realization of smart cities and other environments and enable the dynamic formation of virtual social communities.

The combination of heterogeneous wired and wireless networks, with significantly different administrative and/or channel characteristics that enable users to choose between multiple access points (a trend that is clearly visible in the emergence of heterogeneous networks in 4G/LTE infrastructures [25] [26] and in the consequent interest towards device-to-device communications [4] [5]), nodes' mobility, the possibility of network partitioning, and the cooperation between devices with diverse resource availability are the fundamental pieces of *next-generation networking (NGN) scenarios* [6]. On top of them, also the emergence of novel applications, which enable users to interact deeply with the environment and the territory, but also with nearby people and friends, strongly characterize the NGN environment. Such scenarios are growing more and more common and quickly becoming part of people's everyday life, fundamentally changing cities, buildings, and our homes.

Fig. 1 shows an example schema of a NGN scenario where several base stations and public Wi-Fi access points (APs) provide connectivity to nodes in the network. In that example, a 5G macrocell provides wireless service coverage to a large area, for instance a city or another urban conglomerate. Within its range, small cells (such as microcells, picocells and

femtocells) offer connectivity to devices within a smaller radius from the cell. Typically, macrocells have a transmission range in the order of kilometers (up to 30 km), while the radius of microcells is in the order of hundreds of meters (up to 1-2 km). Differently, picocells have a coverage of tens of meters and femtocells, which are used to bring the so-called “5 bar signal” to small rooms and offices in buildings, have the lowest range [27]. The use of small cells in addition to macrocells permits to offload the main base station and increase the quality of service offered to the users of the network.

In addition to macrocells and small cells, the example in Fig. 1 shows that nodes can also exploit public Wi-Fi APs that provide Internet connection. The number of AP is constantly growing worldwide (see http://www.ipass.com/wifi-growth-map/ for an interactive map on the worldwide distribution of Wi-Fi hotspots) and Wi-Fi is becoming more and more important for the future of mobile networks [28]. Ad hoc connectivity will also be important in the future [29] to enable communications where other network infrastructures are absent or when the network is highly congested, such as when large crowds gather in relatively restricted areas to attend a social event. Finally, Fig. 1 also highlights the presence of a Wireless Sensor Network (WSN), which might represent the network of sensors deployed on the territory of a smart city. In order to make it possible to access the data produced by those sensors, it is also necessary to connect the WSN to the city network.

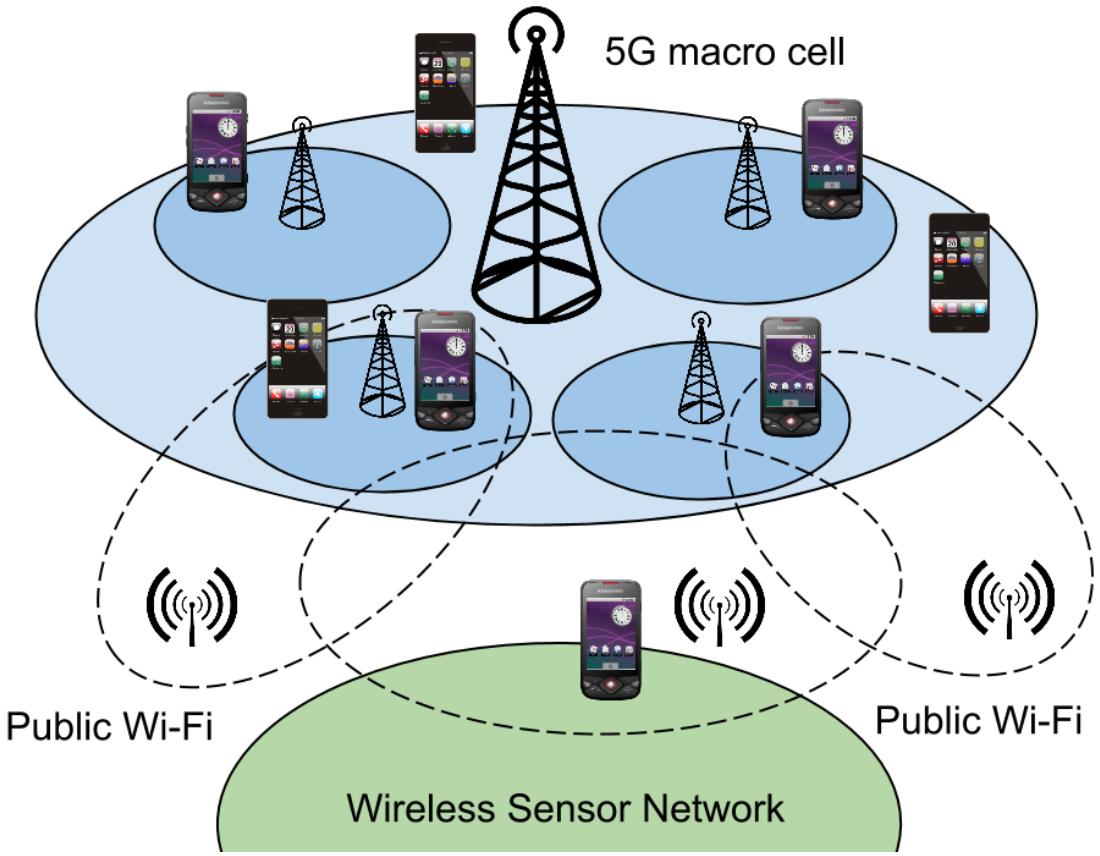

Fig. 1 The next-generation networking scenario

1.2. Tactical Edge Networks

The importance of network operations in modern warfare is today very well established and it is the basis of the "*Network-centric Warfare*" concept [7]. Network-centric Warfare scenarios rely on robust network communications to support timely exchange of information between geographically dispersed entities. However, TENs, the basis for network-centric operations on the combat field, provide one of the most challenging networking environments. Therefore, it is necessary to devise solutions to hide communication issues from applications and services based on the Network-centric Warfare concept.

Tactical networks are normally composed of a combination of networks that range from Mobile Ad-hoc Networks (MANETs), established between soldiers, land and air vehicles, small robots, and other units on the battlefield, and Wireless Sensor Networks, set up between sensors deployed at the tactical edge, up to the infrastructured networks (Local and Wide Area Network, respectively LAN and WAN) that connect the Tactical Operations Center (TOC) or Combat Operation Centers (COC) and other operations headquarters [8]. Fig. 2 shows the diagram of a tactical network, in which multiple networks are connected to

each other through intermittent, heterogeneous links, sometimes provided by highly mobile nodes that deliver temporary connectivity to otherwise disconnected portions of the network. Platoons and other nodes on the battlefield might be able to exchange data with the TOC via radio or satellite links, or by exploiting temporary communication paths established by other nodes that approach them. Thus, it is quite common for some nodes in TENs to suffer network partitioning.

While the COC normally has access to a powerful and stable networking environment, which provides high bandwidth and low latency, jitter, and packet loss, in contrast mobile wireless networks at the tactical edge are characterized by low bandwidth, variable latency, and frequent packet loss. Experiments have also shown that TENs are subject to intermittent connectivity problems. In particular, urban environments have many dead spots, which cause nodes to lose connectivity. In a tactical network, traffic is commonly relayed through multiple hops, so the loss of connectivity to one node may also affect connectivity to other nodes. These turbulent and chaotic network conditions stem from the inherent characteristics of radio communication systems, which include path loss, multipath fading, interference, channel contention and collisions, and multipath routing, resulting in highly variable bandwidth with time and spatial dependencies [9] [30] [31]. Network reconfiguration caused by nodes mobility increases the network churn rate and frequency of link intermittency and makes end-to-end connections more prone to disruption [10]. Further complications arise from velocity differences between air and ground units, which produce major fluctuations in end-to-end channel conditions. As a result, applications that assume continuous connectivity suffer from performance problems and may fail in these conditions.

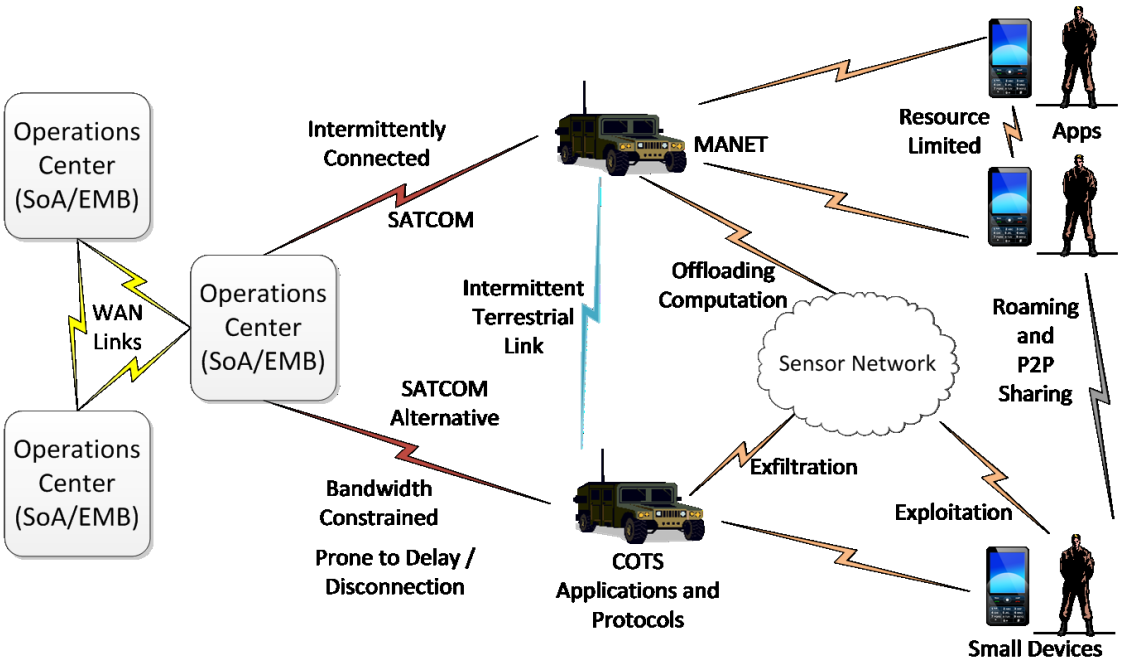

Fig. 2 Example of a diagram of a tactical network

It is possible to characterize nodes in TENs based on the level of resources at their disposal, in terms of processing power, memory available, battery life, and so on. At one end, battery-powered sensors deployed in WSNs and handled devices carried by soldiers on the battlefield are examples of nodes with reduced processing power and limited memory. At the other end, wheeled and Unmanned Air Vehicles (UAVs) are equipped with high-powered servers that can carry out computationally expensive tasks at the tactical edge, while the core of computational, storage, and memory resources reside in the TOC/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. Additionally, nodes can be equipped with one or more network interface controllers (NICs); hence, TENs are highly heterogeneous networking environments. Commonly used technologies in TENs include SATCOM links, 3G/4G communications, military radios, other local wireless solutions such as Wi-Fi and Bluetooth, optical line-of-sight technologies, and so forth [10].

It is often the case, for example due to mission-specific policies or other types of constraints, that some nodes or links are subject to explicit limitations. For instance, some special nodes might be required to minimize the amount of data sent on the network, in order to conserve battery charge or to operate in a clandestine manner. In addition, the characteristics of each type of node are crucial to determine its strategic role in the tactical network. For instance, unattended ground sensors are usually battery-powered and have severe constraints on

power utilization, due to the need to operate for long time, up to a month or more, after deployment. Similarly, UAVs only have short connection windows with other nodes because of their fast air mobility, which causes links to be intermittent. Thus, the great variety in nodes' processing power, mobility, freedom of action, and available connectivity technologies puts severe limits on the admissible network configuration and on the set of nodes that can run central tasks in a TEN [13]. Because of this, each node is fit to take on a few specific tasks and the perfect coordination among all different entities is essential for the success of tactical missions [9].

Mobile UAVs or other airborne assets, such as the Joint-Surveillance Target Attack Radar System (J-STARS, http://www.airforce-technology.com/projects/jstars/), play one key role in TENs. In fact, besides monitoring the battlefield to offer a wider and more complete view of allied and enemy forces positions, UAVs can opportunistically carry data between two or more disconnected portions of the tactical network [8]. This way it is possible, for instance, to provide networking capability to platoons on the battlefield that have no direct connection to the TOC, or to fly over a sensor field to harvest collected data and then carry them to the TOC (or other nodes dedicated to sensor data analysis) for processing. Fig. 3 shows an image representing the scenario just described. Other nodes often present in tactical networks include satellites, manned aircraft and ground vehicles, robotic vehicles (e.g., for search and rescue), high-mobility multipurpose wheeled vehicles, ships, unmanned underwater vehicles (UUVs), and unattended ground sensors (UGSs).

Finally, let us note that TENs are very relevant also outside of military applications. 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 and tools that gave birth to research on Opportunistic Networking and fostered its development [32].

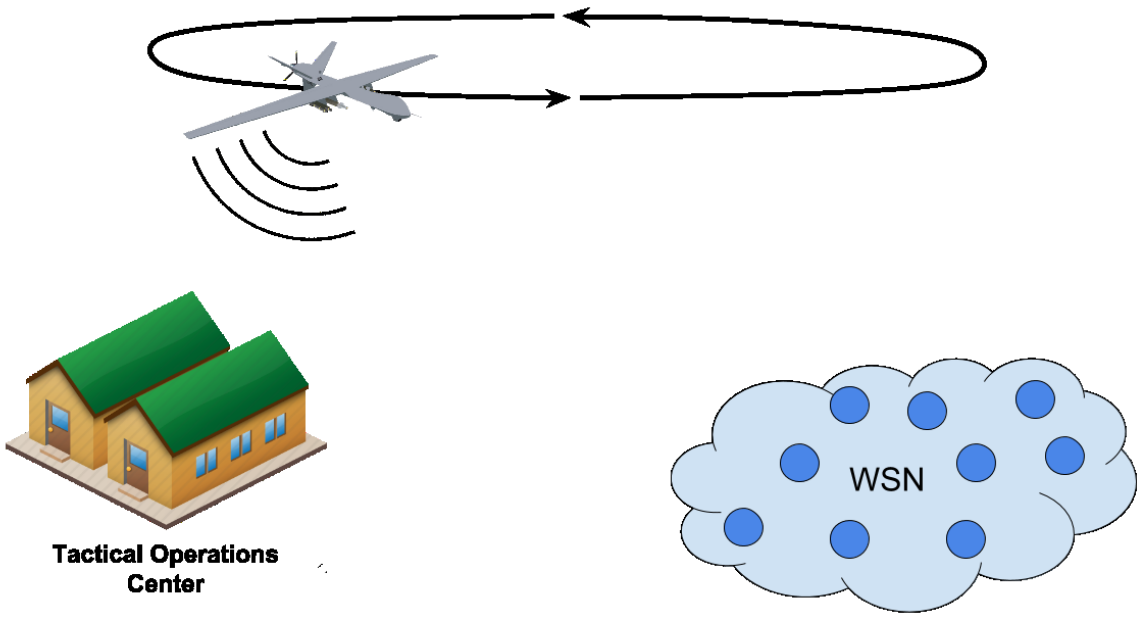

Fig. 3 Example Scenario of an UAV harvesting data from a Wireless Sensor Network and then delivering them to the Tactical Operations Center

1.3. Requirements for Applications Communicating in Challenged Networking Scenarios

The case of smart cities gives an idea about the challenges that NGN scenarios will pose. In those environments, huge amounts of data produced by applications, smart devices, vehicles, and sensors will have to be moved from the edge of the network to the city data center, where processing will take place in order to extract useful information [23] [33]. In a similar way, sensors, soldiers, and other units on the battlefield need to transfer the data they produce to those nodes in the TEN that are equipped with enough memory and computational power to process them, or to the TOC for further analysis and/or permanent storage.

The high dynamicity and heterogeneity of next-generation and tactical networks, with frequent and severe fluctuations in channel conditions, node mobility, disconnected operations, and network partitioning not being the exception but the rule, make the adoption of end-to-end communications particularly challenging [34] [35] [36] [37]. On top of that, the unstable conditions of channels and links might also affect negatively transmissions, favoring network partitioning and disruption of end-to-end paths, and increasing packet loss. These conditions are especially harsh in TENs, where factors like the enemy actions and mission-related constraints can also limit or impede node communications. For these reasons, there is a constant threat on the Quality of Service (QoS) of applications and

services running in NGN and tactical scenarios that, in turn, also affects the Quality of Experience (QoE) offered to the users.

1.3.1. Application Requirements in Next-Generation Networking Scenarios

The new ways through which innovative smart devices, applications, and users will interact in the next future will influence the QoS that the network is capable of providing. In the next paragraphs, I am going to illustrate a few examples of features that I envisage that next-generation applications will support in the near future. The discussion of such features will help me identify the requirements of applications running in challenging networking environments.

Fig. 4 shows a young woman, Sybil, who is walking into a shop. At that moment, her smartphone is connected to the 4G mobile network. Once Sybil enters the shop, the network-sensing component installed in her smartphone detects the presence of the store's Wi-Fi network and initiates the procedure to connect to it. Observing a much higher bandwidth now available at no charge to the user, other applications installed and running on Sybil's smartphone can automatically start a process to "promote" the quality of the offered services. For instance, a social networking app could decide to reduce the compression level applied to pictures and videos before their upload to increase their resolution. While this happens, the shop's customer relationship management (CRM) service transmits a beacon message that activates a dedicated application (which, for the sake of the example, I will call SmartShop) on Sybil's smartphone. SmartShop logs Sybil into the shop's CRM software, where she is recognized and her purchase history analyzed. Hence, as shown in Fig. 4, SmartShop is able to recommend her two dresses from a new collection that could be of her liking, along with a special discount to reward her loyalty as a customer.

Fig. 4 Smart handover with user recognition

In Fig. 5, a woman named Susie has just arrived at work and is walking to her office. She is very passionate about space and planets, and she is watching a video on her tablet about the Solar System. By the time she enters her office, she is just halfway through the video, and so she decides to continue watching it on the much bigger screen she has in her office. Therefore, Susie approaches the 4K monitor she normally uses to give work presentations and, with a gesture of her hands, she is able to move video reproduction from her tablet to the new screen seamlessly. Thanks to the much higher resolution supported, Susie can now admire details about the surface of planets that she could not notice when she was watching the video on her smaller device.

Fig. 5 Transfer of application sessions

Fig. 6 illustrates a man, John, who is walking on the sidewalk in front of a bus stop. Since he missed the live broadcast of the morning news earlier that day, while walking to work he is also watching the rerun of the news on SmartStream, an app installed on his smartphone that plays streaming videos over 4G. Unfortunately, it is rush hour, and cellular communications are rather disturbed because of all the load and interferences caused by nearby users, which translate into occasionally dropped video frames. However, other people at the bus stop next to John watched the morning news that day, so SmartStream can transparently use Device-to-Device (D2D) communication, for instance via Wi-Fi or Bluetooth, in an attempt to retrieve lost frames from nearby peers without soliciting the retransmission of missing frames over the cellular network. This way, SmartStream can hide from John the bad QoS offered by the 4G network, and he is able to enjoy the video without any interruption caused by delays in data retrieval from the cellular network.

The scenarios depicted above emphasize a set of fundamental features on which next-generation applications will need to rely in order to provide the desired QoE to the users. Among them, multihomed smartphones and other portable devices (devices that are equipped with more than one NIC) will need to be able to switch from one interface to another without interrupting the communication session. *Seamless handoff* will be crucial to enable the relocation of open connections to the interface that offer the best quality of service (QoS), in line with the Always Best Connected (ABC) paradigm [38] [39], or when nodes need to change network due to mobility. In fact, NGN scenarios will be highly dynamic and heterogeneous, characterized by the overlapping of several different wireless networks (4G/5G and Wi-Fi, macrocells and micro/pico/femtocells, Bluetooth, Zigbee, and so on). Seamless connection handoff mechanisms will also be the key to take full advantage of alternative connectivity solutions, which will increase rapidly in the next years (such as the number of open Wi-Fi APs [40]), with the ultimate goal of offloading the cellular network and mitigating the growth in mobile traffic expected for the next years [41]. Next-generation networking scenarios will also present the opportunity to exploit multiple communication links at the same time [42] [43].

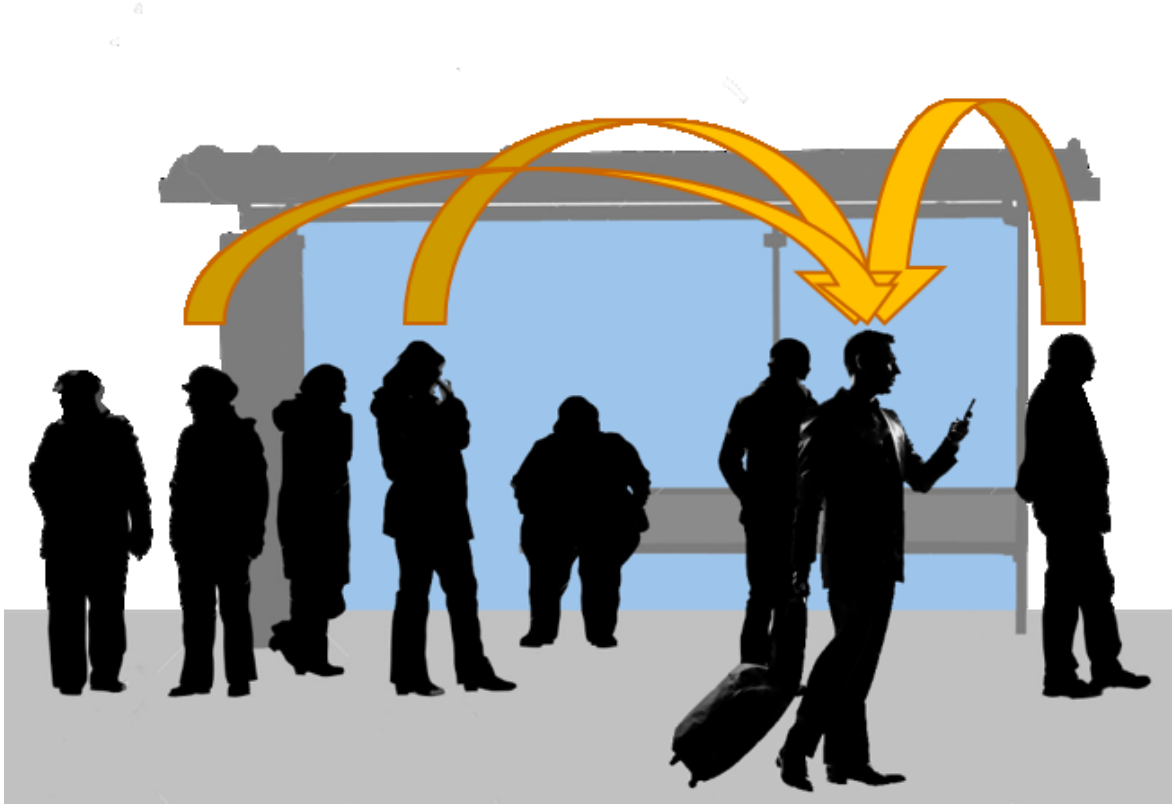

Fig. 6 Transparent retrieval of missing data from nearby users

To provide services such as the one presented in Fig. 4, applications operating in smart environments will need to be aware of the dynamic conditions of the network and the coexistence of multiple NICs with diverse characteristic in terms of bandwidth, latency, power consumption, and costs for the user. *Network awareness* will allow applications to scale their Quality of Service (QoS) dynamically to match the quality of connectivity provided by the network and the NIC and to comply with user-defined policies, which could require a reduction in data consumption when costly connections are being used or the prioritization of some communications over others. In fact, next-generation networking environments will host multiple applications competing for scarce resources, raising the very real need to mediate between applications' QoS requests, considering their requirements and preventing excessive resource consumption.

Services like the one presented in Fig. 5 will also raise the need to implement *location-aware* applications that consider quality of communication and available resources as primary context aspects [44]. *Social awareness* will also be a characteristic trait of next-generation applications, a trend that appears from the emergence of mobile social networking middleware: platforms that monitor and manage contacts between people and support the creation of virtual social communities according to physical proximity criteria, common interests, or the relationships between the participants [45]. Next-generation

applications will implement features with high social impact and strong correlation with devices' location, which call for the capability of adapting network usage to its conditions [46] and to tailor their behavior to the state and resources available on the devices with which they interact [47].

In addition, next-generation applications will provide access to the information and services that are most relevant to users in a specific moment and context, without any explicit action from them. These features will require the support from *push-based information delivery models* and the design of *user-tailored service request prediction models*. Examples include the generation of personalized offers when customers enter a shop, information on traffic status on the road to work, the number of people waiting in line at the cash of one's favorite shop in the department store he/she is in, the access to friends' reviews about the restaurant of which a person is in front, the sharing of opinions with other fans of your favorite TV show who are sitting at the same coffee shop, the automatic retrieval of (part of) a requested object, e.g., a video, an image, a web page, etc., from nearby devices that had previously accessed the same content, and so forth.

The case of smart cities brings under the spotlight two other challenges: how to engage private mobile smart devices (and their owners) in the production and collection of environmental data for the sake of the community; and how to transfer raw sensor data from the edge of the network to the cloud for processing. In fact, as citizens carry around their portable devices on a daily basis, it is impossible for them to connect to the wired network infrastructure, which would be an effective and cheap means to transfer collected data to the cloud. At the same time, it might also be impossible to use the wired network to provide connectivity to a considerable part of traditional urban sensors deployed in the city. In fact, sensors might be installed in areas not covered by the wired network or in positions where it is difficult to bring a cable, or they might lack a proper network interface, such as Ethernet, which would enable sensors to be physically plugged into the network, because it would also increase their unit cost.

The usage of the cellular network to transfer data from sensors to the cloud is a very interesting possibility, as the majority of smartphones and other smart devices can connect to it and many sensors today offer the possibility to install a Cellular Network Interface (CNI). However, this approach also presents several limitations. First, due to the enormous growth in the mobile data traffic expected for the next years [41], the cellular network will probably be unable to satisfy, by itself, all service requests coming from the users. Thereby, adding sensor data to the traffic already weighing on the cellular network would further threaten its ability to serve users' requests. Second, if private devices were to send collected sensor

data using the cellular network, then the owners would consume part of the bandwidth purchased from a mobile network operator, thus resulting in additional costs for private citizens. Therefore, in this context, *mobile offloading* methodologies and techniques are drawing much interest in the research community [25] [48].

Finally, security and privacy requirements of next-generation mobile applications are other central topics in the context of the future Internet scenario. In terms of authentication, authorization, and nonrepudiation, they will be similar to the requirements of current generation mobile applications. However, there are some notable differences. In particular, confidentiality, anonymity, and location privacy will become increasingly important as the modern networks enter the next-generation [49]. While security and privacy raise very important matters in this context, a deeper discussion on the topic is not among the goals of this Thesis.

1.3.2. Application Requirements in Tactical Edge Networks

There is a big interest in adopting Commercial Off-The-Shelf (COTS) hardware and software technologies in military application environments [11] [12], as it enables reaping the benefits of economies of scale and it facilitates and hastens both 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).

SoAs offer 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, which 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) [50] and the US Marine Corps Tactical Service Oriented Architecture (TSOA, https://marinecorpsconceptsandprograms.com/programs/command-and-controlsituational-awareness-c2sa/tactical-service-oriented-architecture-tsoa) [51] [52].

However, SOA-based technologies adopt application-level protocols, e.g., HTTP, and middleware that hinder the adoption of performance optimizations such as request aggregation and/or pipelining and the reuse of transport-layer connections. SOA-based applications also make heavy use of verbose and bandwidth-expensive XML-based data representation protocols. 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 [19]. These problems identify additional requirements for SoA-based applications that need to operate in TENs. In order to support the deployment and reuse of COTS, legacy, and SoA-based applications in tactical environments there is the need to develop solutions that mediate between the communication semantics required by the applications and those that can be reasonably supported by TENs. Such solutions will need to cope with limited bandwidth, high and variable latencies, frequent link disruptions, and network partitioning [51] [19].

Typically, COTS and SoA-based applications operating in TENs are extremely heterogeneous. They handle various data formats, from continuous streams (e.g., video) to images, audio files, and text messages. Each application potentially demands different types of service from the communications network, which may require sequential message delivery and/or different reliability levels (reliability and sequencing are orthogonal characteristics). Some data flows contain independent messages and do not have sequencing constraints, while others do. In addition, certain flows carry critical information that require reliable message delivery, while others carry non-time-sensitive or disposable information that can be re-scheduled to be sent at a second moment in time, in a best effort or partially reliable manner.

Because of variable reliability, sequencing, and latency requirements, in TENs different data flows have different priorities. For example, low-latency flows such as video feeds that need to be viewed in real time have higher priority than sensor reports. In addition, not all video streams have to have equal priority: all depends whether it is mandatory that the video is seen in real time, in order to take strategic decisions, or its visualization may be postponed. Moreover, priorities are not static: as tactical objectives and environmental conditions change over time, the priority of different data flows will change as well.

The network bandwidth available in TENs is often very limited. If nodes generate more traffic than the network (or the link) can accommodate, then data may accumulate in a transmission queue at the application, Operating System (OS), or radio/NIC level. If these queues become full, new data will be dropped or the application will block until the queue

starts to empty. Data queuing can cause temporary spikes and long-term drifts in end-to-end latency along with transmission of unnecessary data. For example, let us consider a node generating GPS position updates at a rate of 1 Hz. If that node loses network connectivity for 30s, the application/OS/NIC queue will accumulate 30 messages. When connectivity will be restored, all 30 messages will be transmitted in the order they were originally generated and enqueued. However, since the only message of real interest is the most recent one, the preceding 29 messages are useless and they simply waste bandwidth and increase latency. It can be extremely difficult to design applications that handle this issue with data queuing, as traditional network communication APIs give no control over packets already in the transmission queue.

Another very important characteristic of a TEN is that it can be reconfigured at any time, due to nodes that may be dynamically added or removed from the system. For example, in a military environment, a new set of sensors may be planted into a battlefield, thereby increasing the number of available nodes. In military environments, both the enemy action and the usage of consumable resources (such as a ground sensor running out of battery power) can cause a reduction in the number of nodes. In addition, many nodes such as aircraft, UAVs, and other highly mobile vehicles may enter and leave the environment at arbitrary times. From the point of view of the TEN, all of these aforementioned situations lead to nodes entering (when a connection can be established) and/or leaving the network (when connection is lost) at times that are usually not predictable. Moreover, the connection to a node can be lost due to a temporary problem in the link, and re-established later.

Another important task in TENs is resource sharing. In fact, nodes can also contribute to the mission goals by sharing processing power, memory and storage capabilities, and display and visualization functionalities. While some nodes provide very specific resources, like a battery-powered sensor with an embedded processor, limited memory, no second storage, and a low-bandwidth wireless connection, other nodes may share general purpose processing power. It is possible, for example, to have UAVs or tanks equipped with powerful server racks, configured with very high amounts of memory, storage capabilities, and no power limitations. By offloading computationally intensive tasks to powerful general-purpose machines located in their proximity, other constrained special-purpose nodes can thus increase battery life, tasks are accomplished more quickly, and results of such tasks can be delivered more effectively to the stakeholders in the tactical network.

Finally, the limited bandwidth available for communications in the tactical environment requires applications to be as efficient as possible. Compression of data segments before transmission is an effective technique to reduce bandwidth consumption. However,

compression requires additional time due to processing on the nodes and, in presence of devices with low computational power, it could increase latency too much.

The characteristics of applications operating in TENs described above call for the design of abstractions and tools that applications can use to tailor the consumption of network resources to the services they implement, in order to reduce the burden placed on the network while still satisfying users' expectations. Those tools should provide features like the right set of delivery semantics, multiple reliability levels, manipulation of queued packets, dynamic flow prioritization, efficient compression algorithms, and so on. Additionally, the support for resource sharing and rapid network reconfiguration, necessary to respond to enemy action and to changes in the mission goals, requires solutions that implement very efficient methods to discover and access new resources that become available in the network, and that are resilient to high network churn rates, (frequent) packet loss, and (temporary) node disconnection.

2. A COMMUNICATIONS MIDDLEWARE FOR CHALLENGED NETWORKS

The characteristics of NGN and tactical scenarios pose a major threat to applications and services. Moreover, the requirements of next-generation and tactical applications, as described in Section 1.3, further increase the challenges that developers will have to face.

Traditional solutions for implementing networked services typically make use of the communication capabilities offered by protocols such as TCP and UDP, the de-facto standard for the majority of Internet applications. However, TCP was designed for the commercial Internet and reliable infrastructured (wired) networks and, as a result, it generally provides poor performance when operating over challenged wireless networks. On the other hand, UDP provides connectionless communication semantics that are incapable of meeting the requirements of many types of applications.

These problems make it hard for applications running in challenged networking scenarios to provide a satisfying experience to the users. This raises the possibility to study and evaluate different communication paradigms that can better support communications in scenarios like next-generation networks and TENs. Before choosing to switch from traditional communication solutions to completely new paradigms, it is also necessary to analyze the characteristic of information. In fact, the nodes' operating context, including their social relationships or their tasks in the tactical mission, affects significantly some important properties of the information that applications produce and consume. This delineates an information model that highlights common aspects of applications running in challenging scenarios that developers need to consider when choosing the communication paradigm that best fit next-generation and tactical applications.

In this Chapter, the Thesis first analyzes the problems of using traditional networking solutions, like TCP and UDP, in extremely challenging networking environments. Then, the Chapter discusses the information model that arise in NGN and tactical scenarios. In its last Section, the Chapter presents an analysis of *Opportunistic Networking* and *Information-centric Networking*, two innovative communication paradigms that aim at making a more efficient use of network resources and supporting nodes' communications in challenged and constrained networks.

2.1. Difficulties of Traditional Communication Solutions in Challenged Networks

TCP (for Transmission Control Protocol) is a transport layer protocol of the Internet Protocol Suite, and the de-facto standard for the majority of Internet applications and services, such as HTTP(S), e-mail, SSH, etc. TCP was designed with the target of maintaining strict separation between the layers of the ISO/OSI stack.

TCP establishes an end-to-end connection between two applications running on nodes of an IP-based network (thereby, it is common to refer to both protocols at the same time as "TCP/IP") and provides the abstraction of a reliable, ordered stream of bytes flowing between them. TCP guarantees reliability by means of a mechanism based on acknowledgments (ACKs) of correctly delivered packets and on the timeout-driven retransmission of unacknowledged segments. To ensure the ordered delivery of data, TCP uses a sequence number to identify the first byte of each segment; this allows the receiver to also restore the correct transmission sequence in case of packet loss. TCP also provides flow control and congestion control, to share the bandwidth between multiple connections fairly and avoid network collapse.

The TCP/IP model was designed for wired infrastructure environments, hence it exhibits several weaknesses in networks with low bandwidth, frequent packet loss, variable latency, and prone to temporary disconnection, such as next-generation and tactical networks [30] [53]. In fact, TCP/IP does not consider several mobility-related aspects that are of fundamental importance for next-generation mobile applications and any service running in NGN environments and TENs. Assuming a continuous end-to-end connectivity between two communicating peers, TCP breaks as soon as one of them becomes temporarily unreachable. This might happen for multiple reasons, such as when a node switches its 4G connection in favor of Wi-Fi when it enters a building, or when it leaves the network due to mobility, e.g., because a UAV flies away from the area above a platoon. Another common cause of end-to-end connection disruption with TCP is network partitioning; this might happen, for instance, if one of the nodes along the end-to-end path breaks or moves away from the connectivity range of its neighbors.

Because of these problems, the use of TCP forces software developers to implement cumbersome solutions at the application level to deal with connection disruption. In addition, TCP suffers from severe performance issues in wireless environments, as its congestion avoidance mechanisms often misinterpret packet losses caused by the unreliability of the wireless channel as symptoms of congestion and needlessly reduce the transmission rate

[11]. To tackle this problem, researchers have invested much effort to improve TCP over the course of its existence. This has led to the development of a number of alternative congestion-avoidance algorithms, which add features like slow-start, fast recovery, fast retransmit, and different ways to manage the size of the congestion window in response to packet losses and successful ACKs.

TCP implements a single first in-first out (FIFO) transmission queue and its API 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. For instance, Blue Force Tracking (BFT) applications have the goal to send to other nodes in the network the position of ally forces on the battlefield. If packets queue up in the TCP buffer because a node running a BFT application temporarily loses connection, when connectivity is restored all queued packets are transmitted in order. Since the only interesting information is the latest node position, all data sent before the latest update cause a waste of resources. In fact, applications like BFT do not strictly require reliable byte streams, which would be excessively expensive, especially in presence of connections with low bandwidth and/or high latency.

The negative impact of the TCP queuing model on latency is well recognized also in wired Internet environments [54]. However, this issue is particularly troubling in time-sensitive and multiple-priority applications running in unreliable networking environments like TENs and NGN scenarios. Moreover, 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, further increase problems.

When applications do not require in-order and reliable data transmission, developers' only choice consists in switching to other transport protocols, like UDP. Connectionless UDP-based communications are often inadequate for many applications. In fact, best-effort communication semantics force the development of ad-hoc retransmission schemes at the application level to ensure reliable delivery of critical messages. At the same time, UDP broadcast is usually inadequate for group communications, as it is not disruption tolerant and it does not allow the delivery of messages to be restricted to a subset of nodes in the network.

2.2. The Information Model in Extremely Dynamic Mobile Networking Environments

The problems that traditional communication solutions suffer in challenging networking scenarios make it very hard for developers to implement applications that can provide a satisfying experience to the users. To this purpose, several studies appeared in the literature that proposed new communication paradigms that depart from traditional solutions based end-to-end direct communications like TCP and UDP. However, before studying and evaluating novel solutions, it is essential to analyze the operating context of applications running in NGN and tactical scenarios. This analysis will disclose some important properties of the information produced and consumed by applications that need to be considered in the evaluation of novel communication paradigms.

Next-generation mobile applications will provide services that interact with people and things, enriched by the processing of large quantities of data gathered from the many types of sensors that monitor the environment around the users. This strong link between users, things, and the surrounding environment considerably affect the type of content that is produced. In a similar way, the location, direction, and role of nodes in a TEN also affect the content that nodes produce and in which they are most interested. Thus, it is possible to identify a number of properties of the information on which nodes' context has a relevant impact. The consideration and analysis of these properties is essential to design and evaluate new communication solutions that can match applications requirements in challenging networking scenarios.

One such property is the *volatility of the information value*: with time, users tend to access content less and less, a fact that reflects the *obsolescent nature of information*. Modern social networks, where people are most interested in the newest pieces of content published by their friends and family, offer a good example of such aspect. Nonetheless, others services exhibit the same property: news, live event streaming, hazard/disaster alert, street traffic conditions, video-chat, etc. all rely on the quick distribution of new content, whose value (in the form of utility for the users) decreases quickly with time. In the same way, in tactical environments, the latest data on the enemy's movements and actions acquire much greater value than images or videos of the enemy's activity recorded in the past. Furthermore, some applications might only be interested in delivering the most recent data to the users, such as those produced by movement detectors placed on the perimeter of the battlefield or by traffic monitoring cameras installed in the proximity of dangerous street intersections. In those cases, the importance of data critically drops down as soon as more updated data becomes available. These characteristics call for the prompt identification of

outdated data in the network, so that resources can be managed properly to offer a better QoE to the users.

Another property of the information is its *priority* [9]. Many applications produce content with different priority levels. For instance, smart city applications that support mobility of people might want to disseminate data containing traffic jams or storm alerts as quickly and as broadly as possible, even at the cost of preempting the dissemination of lower priority data. As another example, new mission directives produced by troop leaders or generals from the CoC should reach the troops and other interested nodes at the edge of the tactical network with higher priority than, for instance, data coming from a WSN. Applications might also want to explore layered coding techniques to improve the QoE offered to their users, thereby assigning higher priorities to the messages containing reference information and lower priorities to the ones containing differential data, even if those messages conceptually belong to the same service. As an example, weather forecast applications usually provide a complete set of related information, such as temperature forecasts, rainfall, weather and barometric pressure maps, etc., but most of the time users will only be interested in the weather forecast for their current location in the next couple of days. Similarly, live event streaming applications or applications that enable the retrieval of videos with low resource consumption could provide multiple resolution levels for their video flows: basic frames would be transmitted in higher priority messages and delivered to all viewers, while higher resolution frames would be packed within lower priority messages and disseminated in a best effort fashion.

When used properly, prioritization functions allow applications to give nodes in the network suggestions on how to handle the data. For instance, different priority levels might be associated with different caching or dissemination policies. In general, higher priority messages should have access to a greater amount of resources when compared to messages marked with a lower priority level. However, it would become very difficult for an application to decide on the priority level of messages it produces with respect to those generated by other applications. In fact, the task of choosing the appropriate priority level should take into consideration several factors that are most likely unknown to applications and developers. In NGN scenarios, such factors include how important some service is for the user, given what he or she is doing or needs to do, what job the user does and how relevant the services provided by the application are relevant to his or her job, if the provided service is enabling the user to interact with other persons or devices and how important the outcome of that interaction is to the user, and so on. In a TEN, similar factors need to be considered. For instance, how important a service is to the node running it, given its role in the tactical environment (is the node a device owned by a soldier or a commander, is it a

UAV, is it some kind of sensor?), how critical the carried information is for the outcome of the mission, or how important the data might be for nodes located somewhere else in the network, e.g., the COC or soldiers close to the enemy lines. Furthermore, decisions about the priority level to be given to certain messages should also consider the priority and content of other messages (or at least any information made available by metadata associated to those messages) currently in queue for transmission.

For this reason, the concept of priority should be *application-specific*, thus affecting how resources are allocated for different messages produced by the same application, instead of the amount of resources to which each application has access. In fact, applications should have all the necessary knowledge to establish the most appropriate priority levels for the messages they generate. Application-specific priority also prevents that greedy applications keep raising their priority level to have access to larger amounts of bandwidth, memory, and processing power at the expenses of other applications.

2.2.1. Patterns of Communication

In the choice and evaluation of novel communication paradigms for NGN and tactical scenarios, the patterns of communication of applications will play an extremely important part. In fact, next-generation and tactical services and applications present communication patterns that differ significantly from the traditional client-server paradigm. In fact, nodes in next-generation networks and TENs will take on both the roles of content producers and content consumers, switching naturally from one to the other, or even playing both roles at the same time, in a very dynamic way. The role assumed, as well as the content's recipient(s), will depend on the applications running on one node, the content's nature, the node's current location, and the users' social context and habits (in the NGN scenario) or the soldiers' rank, duties, and mission objectives (in a TEN). These aspects outline a many-to-many communications pattern, as opposed to the one-to-one communications model of client-server applications [55].

In many cases, applications in NGN and tactical scenarios will not be interested in specific data, but more generally in any updated information concerning one or more topics. For instance, emergency notification applications employed during disaster recovery would be interested in receiving any messages carrying news about ongoing hazards, instead of only those that contain information about specific hazards. Similarly, social applications that allow people to rate and comment about their experiences in restaurants, bars, and other places would be interested in retrieving any messages with updated ratings or new

comments on nearby places, instead of only those that carry the data about the places specifically requested by the users.

These different patterns of communication open the opportunity to distributed optimization approaches, such as network-level caching (also referred to as in-network caching) and data delivery. The next Chapter will analyze them better when describing new communication paradigms that make an effective use of such techniques.

2.3. Communication Paradigms for Challenged Networking Scenarios

The poor performance and inadequate communication semantics that many solutions provide in mobile, wireless, and extremely dynamic environments, such as tactical and next-generation networks, motivated researchers to investigate new paradigms that better suit those scenarios. Furthermore, the need to support the resource sharing requirements of location-, social-, and context-aware applications also suggests the rethinking of the programming paradigms for communications. This raises the opportunity to adopt novel solutions, such as the *Opportunistic Networking* and the *Information Centric Networking* paradigms.

2.3.1. Opportunistic Networking

Opportunistic Networking techniques have recently emerged to face many of the challenges that naturally arises in heterogeneous, dynamic, and resource constrained scenarios: from TENs [36], deep-space communications [32], and disaster recovery scenarios [14], to smart cities and vehicular networks [56], and all those situations where a network infrastructure is not (always) available [57]. The Opportunistic Networking paradigm comes from the networking concepts that naturally emerged in the MANET and Delay Tolerant Network (DTN) research fields and evolved into a more complex and effective set of networking strategies and protocols.

Given the short-lived nature of links between nodes in dynamic wireless networks, routing is one of the most challenging and pressing problems in Opportunistic Networking. Opportunistic Networking advocates a completely different conceptual approach to avoid the set of problems that direct and end-to-end communication solutions pose on the design of applications [32]. First, it aims at supporting communications in highly dynamic scenarios with temporary contact opportunities between nodes by leveraging store-and-forward

techniques and by taking advantage of node mobility to support the information dissemination process [14]. In addition, this paradigm takes into account the information coming from both the application and the environment contexts. By analyzing the aspects in human interactions [16] or in mobility patterns [15], or both the context [58] and the content of exchanged messages [59], applications that rely on Opportunistic Networking can maximize the possibility to exploit resources offered by nodes that temporarily fall under the connection range, thereby increasing the effectiveness in reaching their goals.

This way, the Opportunistic Networking paradigm can be very effective in cases where the network is frequently subject to partitioning or when there is the need to offload congested networks [60] [61] [62]. For example, in a smart city, nodes generate traffic to support the needs of a very variegated set of users: from the citizens, to emergency and security units, the local government, and all sensors deployed on the urban territory. However, those nodes might not be able to directly access the wired network infrastructure, for instance because they lack an Ethernet NIC or because they operate in areas of the city not covered by the wired network; similarly, they might not be in range of any Wi-Fi or WiMAX AP. At the same time, as explained in Chapter 1, the 3G/4G cellular network might not be an option, since the generated traffic could be excessive, the network might be congested and the available bandwidth insufficient to accommodate all traffic generated by nodes, or sending data over it might be too expensive. In these and akin situations, nodes could take advantage of opportunistic communications with nodes occasionally roaming in their proximity. This would create temporary connectivity options that nodes can leverage to tackle partitioning or to avoid sending data over the cellular network.

The research literature on routing algorithms for Opportunistic Networking usually classify them into two main classes: single-copy and multi-copy [63]. In the former case, there is always a single carrier for each message; in the multi-copy case, instead, messages can be duplicated in order to increase their availability throughout the network and exploit multiple paths to reach destination. Therefore, approaches that belong to the multi-copy branch trade a higher resource consumption for an increased delivery ratio and reduced latency.

The scientific literature further divides routing algorithms for Opportunistic Networking based on the type and amount of knowledge they require for the routing process. In accordance with [64], I will distinguish between context-oblivious, mobility-based, and social context-based algorithms. Algorithms like Epidemic Routing (ER) [65] and Spray and Wait (SnW) [66] belong to the first category, PRoPHET [67], MaxProp [68], and Spray and Focus (SnF) [69] are examples of mobility-based algorithms, and finally HiBOp [70] and

SpatioTempo [71] are two examples of social context-based routing protocols. I invite the reader interested in a detailed description of any of those protocols to read the cited works.

Context-oblivious protocols have the advantage of simplicity, as they do not need to gather or build any knowledge on the environment or the neighboring nodes, and they can guarantee low packet delivery latency at the cost of large bandwidth and memory overhead. Typical approaches to reduce resource consumption include adding a Time-To-Live (TTL) field to the header of packets and limiting the number of copies of a message that can be redistributed into the network (ER uses TTL, while SnW uses both techniques). Mobility-based routing protocols try to reduce the overhead of context-oblivious protocols by exploiting historical data on past encounters with other nodes or the knowledge about other nodes' mobility patterns. Finally, social context-based routing protocols are based on the observation that nodes' movements mostly depend on their carriers, and so they extend the concept of mobility-based routing protocols by taking into account users' social relationships to predict nodes' mobility patterns.

Even if comparative tests show increased resource efficiency when comparing mobility- and social context- based approaches to context-oblivious protocols [67] [72], it is important to make a few observations. First, context-based routing protocols require gathering enough data before they can extract any useful knowledge from them. This type of requirement is very difficult to satisfy in presence of highly dynamic networking environments. Second, those protocols assume that devices can set aside a certain amount of memory to store information about each node encountered in the past. I can thus highlight two critical aspects: first, the need for a data collection phase whose duration cannot be underestimated nor predicted, since it depends on the characteristics of each pair of nodes; and second, the high amount of memory necessary to store social and mobility information about hundreds or thousands of people that populate urban environments and with which users can cross paths in a relatively short time. On top of these hindrances, context-based routing algorithms usually leverage their knowledge to find the best path to reach a single target; therefore, it would be hard to use them effectively to deliver data to multiple nodes in the same network [73].

The results presented in [74] further support this thesis. In their work, Barzan et al. use mobility data obtained from real traces of mass events to show how flooding-based protocols can perform better or show comparable results to more complex routing solutions, such as PRoPHET. Moreover, [75] backs up the observations made in the previous study by showing that the most complex routing solutions were designed to maintain end-to-end

connectivity between nodes in MANETs and opportunistic networks, but such property becomes superfluous, and sometimes even detrimental, in content-centric networks.

2.3.2. Information-centric Networking (ICN)

ICN has recently emerged as another very interesting communication paradigm to serve applications and services working in extremely dynamic and heterogeneous scenarios, like those of the next-generation [76] and TENs [77]. In fact, the interest-based routing (also called name-based in the literature [18]) implemented in ICN do not follow the direct end-to-end paradigm typically used by today's Internet applications, but relies on the publish/subscribe communication paradigm, which many research efforts have proved to be very effective in presence of node mobility or within dynamic environments [78].

ICN aims at shifting the focus of the Internet from hosts, i.e., the data's physical location, to the content that users want to access [17]. Each content is split in one or more information units, called Information Objects (IOs). IOs are univocally and globally addressable through their unique name (or identifier), and they can be retrieved by means of a request/reply communication model. Nodes that want to retrieve some specific content need to subscribe their interest to it by sending an interest message that gets distributed across the network and stored in the memory of all nodes it traverses. Any encountered node that has the requested content in its cache can serve as a rendezvous point between the publisher and the subscriber and forward the data back to the subscriber. Depending on the single implementations, with ICN applications can either request a specific IO or notify their interest towards all IOs of a given type. In ICN jargon, any node involved in routing of requests or forwarding of IOs is called Routing Node (RN).

By decoupling communications along both the dimensions of time and space, ICN permits IOs to be published at any moment in time and applications to notify their interests in those IOs at any point in time. This way, the communication is asynchronous and there is no need to set up an end-to-end connection between a content requester and its provider. In addition, nodes interested in some content do not require any knowledge about the location of the information.

The focus on IOs brings several advantages to ICN, such as the capability of supporting multi-party communications, enabling efficient resource utilization through IO caching, and finally of providing low-overhead local communications, which are essentially important for implementing social-, location-, and context-aware applications. In addition, the content-centric perspective of ICN enables the transfer of IOs from multiple sources and/or to

multiple destinations as well as the retrieval of different IOs (or even different portions of the same IO) from multiple nodes at the same time [79]. The result from the users' point of view is an increase in performances, while the network can benefit from the distribution of traffic across multiple sources and paths and, consequently, from a fairer usage of the available bandwidth.

Those characteristics also make the ICN paradigm naturally able to support communications middleware designed to take advantage of the wide range of network interfaces equipped on modern mobile devices, which goes from short-range, low-power communications, e.g., ZigBee or Bluetooth, to medium range communications, e.g., Wi-Fi and WiMax, to expensive, long-range communications, e.g., 4G/LTE/5G and satellite [80] [81] [76]. Next-generation applications will thus be able to keep operating seamlessly on mobile devices as the underlying ICN-based middleware provides always best connected services or even uses multiple links at the same time, exploiting the features of the ICN stack that permit to take advantage of multiple links for the retrieval of the requested IOs.

In addition, ICN-based communications middleware can seamlessly reap the opportunities provided by information locality and application-independent information exchange to optimize communications. In-network caching techniques in ICN minimize the amount of resources required to communicate and share information between nodes in topological proximity. At the same time, ICN enables cross-application in-network data caching, which increases information sharing between different applications and, in turn, it reduces the number of redundant copies of resources in the network and the topological distance between IOs and consumers. In addition, in-network caching can put to good use the unused storage that resides at the edge of the network [81] [82].

In-network caching also enables ICN to support disconnected/delay-tolerant communications and to reduce the impact of network partitioning significantly, in a manner similar to Opportunistic Networking. By caching the "trendiest" IOs in nodes that are located closer to subscribers, ICN increases the number of neighboring nodes that are able to serve requests for the most common contents [79]. So, even in case of network partitioning, where no paths from a subscriber to the publisher are available, it might still be possible to retrieve the requested content from other nodes within the reachable portion of the network.

A lot of research work has been done recently in the field of ICN caching [83] [84] [85] [86] [87] [88] [89]. Caching techniques in ICN can be divided in *on-path caching*, where nodes can exploit only the content cached along the path(s) from source(s) to destination, and *off-path caching* strategies, where nodes can also exploit the information of content cached on

nodes that do not belong to such path [83]. Implementations further vary according to the cache dimension, sharing mechanism, decision policy (what content is placed at which cache node), eviction policy [87], and the content popularity model on which the specific caching technique is based [85] [90]. Regardless of the different implementation choices, all approaches assume the availability of large, fast storage memory at all caching nodes in order to implement efficient routing decisions and to enable opportunistic content caching [87] [91]. This assumption does not hold in the NGN scenario.

Depending on the caching decision policy and the nature of the content, all, some, or none of the RNs that take part in data forwarding will also cache (part of) the forwarded IOs (on-path caching). Additionally, implementations of off-path caching policies have nodes notify the other nodes in the network, or a centralized content object registry service, about the IOs they have in their cache to improve the routing of interest messages [92]. However, off-path caching is more expensive than its on-path counterpart is and, especially in presence of dynamic network conditions and unreliable links, its benefits might not be enough to balance out the increased demands placed on network resources.

A case could be made for in-network caching in wireless environments, thanks to its particularly favorable properties in presence of unreliable links and unstable paths [75], and in heterogeneous networks with MANET and cellular communications. For those scenarios, the possibility of retrieving cached content directly from nodes at the edge of the network could significantly improve the efficiency of routing and alleviate congestion on the cellular network [93]. The consequence of ubiquitous in-network caching is that the competition for one node's memory resources will not only occur among the multiple applications running on that node [6], but also with IOs that have been opportunistically cached for the purposes of improving the overall network performance, making cache sharing a more complex matter.

Cache sharing is an aspect of ICN that so far has received a relatively low attention. The reasons is partially due to the high number of issues that are still open [94], but also to the different importance cache sharing assumes when the context switches from the wired Internet to the next-generation scenario. [87] is one of the few studies that focuses the attention on cache sharing. While the observations made in that paper rest upon the wired Internet environment, several concepts still stand when taken to the NGN scenario. The authors point out that the optimization of cache sharing mechanisms depends on the point of view (which can be either that of the users, of the contents, or of the providers), and that there are two ways to manage memory sharing: using either a fixed or a dynamic cache partitioning system.

We believe that the research on cache sharing mechanism will acquire a greater importance in NGN scenarios, thanks to its capacity to reduce bandwidth usage and latency in a scenario where most content requests originate in wireless networks composed of resource-constrained nodes. However, more efforts on this topic are necessary before ICN solutions can support next-generation mobile applications. To this end, the techniques developed by the research in the field of Opportunistic Networking also represent particularly attractive directions of investigation [95].

3. THE MIDDLEWARE-BASED APPROACH

The study and evaluation of novel communication paradigms that address the problems arising in next-generation and tactical networks is an important step ahead towards supporting applications in such challenging networking scenarios. However, there is also the need to develop solutions to provide applications with the necessary abstractions and methods to take advantage of those paradigms.

More specifically, communications middleware specifically designed to support applications in extremely challenging networking scenarios represent a very interesting approach. Besides adopting paradigms that better suit the operating environment, middleware-based communication solutions can also provide useful tools and techniques to promote the smart usage of the scarcely available resources and apply communication optimization strategies that operate contextually at both the application and transport levels.

The first part of this Chapter describes advantages and requirements of approaches based on communications middleware. In the second part, the Chapter introduces the Agile Computing Middleware (ACM), a communications middleware purposely designed for challenging networking scenarios. The ACM is the result of years of research in the field of communications in extremely challenging networking environments, such as TENs, performed by the Nomads research team at the Florida Institute for Human and Machine Cognition of Pensacola, FL, USA (http://www.ihmc.us/research/software-agents-agile-computing-distributed-computing/).

3.1. Advantages and Requirements of Middleware-based Solutions

Building smart applications and services for next-generation and tactical networks is a very challenging task [6]. In fact, a multitude of different applications with very different requirements and priorities will run in those environments, and they will have to compete to access the scarce network resources and cope with the challenges that arise from node mobility, link intermittency, and network partitioning. To operate effectively, those applications will need to leverage new communication paradigms, which can effectively take advantage of the capabilities of modern mobile devices to access the network through multiple communication technologies.

Writing applications based on novel communication paradigms, like Opportunistic Networking and ICN, without any support from the operating systems, frameworks, or libraries, and using traditional end-to-end paradigms as the basic building blocks would be extraordinarily arduous and likely to result in a very complex and time-consuming effort for software developers. In fact, almost the totality of modern Operating Systems offer primitives based on the Socket API to enable the development of networked applications and services and do not support other communication paradigms.

The Socket API allows applications to easily set up end-to-end communications based on TCP or UDP, given the IP address of the destination and a port number. However, as discussed in Chapter 2.1, those protocols do not fit well the requirements of applications that operate in challenged networking environments. Additionally, the Socket API tends to conceal underlying layers and the link conditions from the caller, to favor simplicity and reusability. These characteristics make it very hard, or even impossible, to use the Socket API to develop applications that are aware of network conditions or that are based on communication paradigms that differ completely from the abstractions provided by TCP and UDP.

Nonetheless, the dynamicity and heterogeneity of next-generation mobile networks and TENs call for the adoption of new communication paradigms and network-aware communication models specifically designed to consider the characteristics of mobile and tactical environments, with frequent and severe fluctuations in channel conditions and node mobility not being the exception but the rule. As a result, they should support applications in implementing a continuous adaptation process of the communication function to reach a trade-off between the application requirements and the current network conditions. This would significantly facilitate the development of applications that must automatically adapt their QoS to the current network conditions, promoting or demoting the level and/or the type of service offered to their users.

This situation calls for the design of solutions that support developers in writing applications based on new communication paradigms and network-aware programming models, without having to rely on low-level, often not portable primitives to implement those models, a task that is extremely complex and error-prone. More specifically, developers could take advantage of middleware that provide overlying applications with a set of tools designed for extremely challenging and dynamic environments. Network-aware middleware solutions that implement multiple communication paradigms and allow developers to choose the one that better fit the application context and operational conditions would represent an invaluable asset, as they would let developers focus their efforts on business logic instead

of wasting energy and time on the design and implementation of low-level networking features.

For instance, Opportunistic Networking opens a wide range of options to applications, which can choose between many routing/forwarding protocols, replications techniques, and communication semantics adaptively, and exploit multiple NICs on the nodes to maximize their effectiveness. To this end, applications need to be network-aware, taking into account both their goals and the state of the networking environment. Therefore, a middleware layer that provides applications with the access to Opportunistic Networking techniques, the support for multiple network interfaces, and the knowledge about the state of the network is a very promising solution. Such middleware could implement different information dissemination strategies and allow applications to choose the strategy that best fits the current conditions.

In order to support network-aware programming models, it is essential that the middleware performs a continuous monitoring of resource availability and utilization. By accessing historical data on communication links, e.g., RSSI, lifetime, throughput, or any combination of them, and on neighboring nodes, e.g., number, characteristics, and duration of proximity interval, the middleware could gather valuable knowledge about the environment and use it for multiple purposes. In addition to the rather straightforward use for best network (attachment point) selection, environment knowledge could be exploited for decision making in channel access prioritization between the competing applications running on the node (while also considering the user's preferences and the applications' priorities and characteristics). Preemption and reallocation of channel resources are both major requirements of next-generation and tactical mobile applications and a fundamental departure from the TCP/IP model, which is based on flow control procedures that tend to assign an equal share of bandwidth to each connection.

The middleware should share the knowledge gathered about the environment with applications that want to be informed about the available resources, to allow them to implement smart service tailoring strategies. Involving applications in the communication and QoS decision loop is essential, as their needs can be quite different. The effective support for QoS adjustments also requires a set of APIs that give applications advanced control of message queues. The API set should provide features to allow applications to cancel the transmission of a message, to modify its content and its priority, and to change the delivery semantics. These features would be extremely efficient in tackling the effects of packets building up in the transmission queue during temporary disconnections, as in the example with BFT applications in TENs given in Chapter 2.1.

The characteristics of NGN and tactical scenarios highlights another main requirement for next-generation communication middleware: the support for seamless mobility. More specifically, the middleware should enable mobile connections, that is, connections whose endpoints could be dynamically and transparently reconfigured (or even switched to another device) in case one device moves to a different part of the network without disrupting the related service session. Mobile connections also represent the basic building block for the realization of captivating location-aware services, like the one depicted in Fig. 5, where Susie uses a gesture to move the documentary from her tablet to the big screen in front of her without interrupting its viewing.

In addition, communications middleware for mobile and dynamic environments cannot assume either end-to-end or all-the-time connectivity, but they need to support disruption-tolerant communications. In fact, sometimes the recipients of a communication are nodes that periodically travel areas that are not covered by any (accessible) network, like when a UAV flies over a platoon and provide time-limited connectivity to the soldiers. In other cases, users may voluntarily choose to disconnect their devices from the network (temporarily) to save battery life. These situations demand effective mechanisms to withstand long-term disconnectivity periods and to improve reliability, such as in-network or node-level caching and periodic retransmission of important (non-obsolete) data.

Another fundamental requirement of communications middleware designed for challenging networking scenarios concerns resource usage and sharing. In situations where nodes are much heterogeneous, going from sensors and portable devices, up to servers on the cloud and in the smart city data center, adapting the consumption of resources to the node's characteristics and conditions is essential to improve the system's efficiency. The middleware should also implement techniques to discover available resources in one node's proximity and to share resources with other nodes, in accordance with user-specified preferences. Prediction techniques could further improve the decision making process of the middleware. By anticipating the contact windows with resourceful nodes, the middleware might decide to defer the forwarding of specific packets to favor, for instance, the use of cheap ad hoc links instead of more expensive 4G connections, in order to save resources and offload the cellular network.

With the progress and the diffusion of IoT scenarios and cooperative P2P resource-sharing models, and with the solid importance that location-aware services has already revealed in many contexts, peer discovery across multiple networks and the possibility for the nodes to form/join groups of interest dynamically will likely assume a strategic importance in NGN

scenarios and TENs. In such environments, communications middleware could leverage cooperative P2P models to enable local resource sharing and information dissemination, in order to improve resource availability and decrease energy consumption. Supporting the formation of overlay networks that link nearby nodes with similar interests, running similar applications, or that are willing to share (part of) their resources could further improve the communication performance.

3.2. The Agile Computing Middleware (ACM)

The Agile Computing Middleware (http://www.ihmc.us/research/software-agents-agile-computing-distributed-computing) is a research project that aims at answering the challenges of NGN and tactical environments to which I contributed during my career as a PhD student [20] [36]. The ACM has the goal of providing a valuable and comprehensive collection of components to support the development of efficient distributed applications. Each ACM component, as depicted in Fig. 7, is designed to address a set of commonly related problems that often recur in mobile wireless scenarios. Several components of the middleware are available under the open source GNU General Public License version 3 (GPLv3) at the address https://github.com/ihmc/nomads.

3.2.1. Mockets

The Mockets (for Mobile Sockets) [96] [10] framework is an application-level communications library specifically designed to support adaptive applications in extremely dynamic environments, such as TENs and NGN scenarios. The Mockets component offers a rich set of functionalities that, as several experiments in the field have confirmed (the reader can refer to Chapter 4.1 for an experiment of this kind), are very effective in addressing challenges that applications face in next-generation mobile environments [97]. Mockets provides a comprehensive, TCP-like API that implements message- and connection-oriented communications on top of UDP.

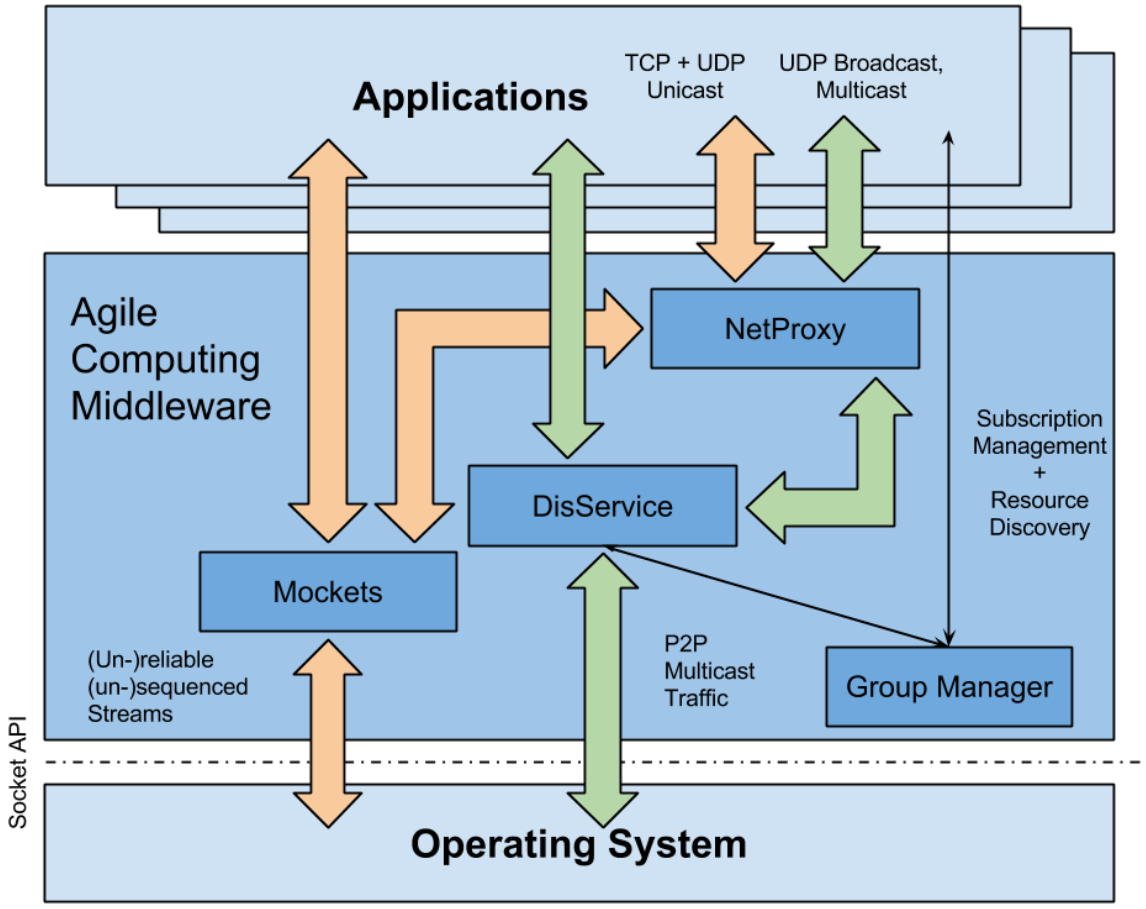

Fig. 7 Architecture and interaction between components of the Agile Computing Middleware

Fig. 8 shows the framework's architecture and its interface with applications and the network. Within each Mockets connection, applications can open many independent data flows and configure each one with different QoS and priority levels that can be changed at any time, in a dynamic manner. Each flow is then mapped into an independent queue in the middleware; this implies that QoS and prioritization settings affect all open connections on the node, but Mockets has the ultimate control over them. The middleware can therefore leverage user settings and information about current channel conditions to adjust the QoS and priority levels set by the applications.

Applications that rely on the Mockets framework can query it to have access to statistics on channels' conditions and the state of connections. In addition, Mockets enables the replacement of enqueued but outdated messages and provides numerous timeout options and policy-based bandwidth control. Finally, the framework guarantees complete orthogonality in data delivery semantics: each flow can be configured for any combination of sequenced/unsequenced and reliable/unreliable message delivery.

Mockets supports session mobility, which allows next-generation application developers to easily implement the functionalities that allow Susie (the reader can refer to the example in Section 1.3.1 and Fig. 5) to move the communication endpoint from her tablet to the flat-screen terminal in her office, without ever interrupting the streaming of the documentary. In fact, not only does Mockets enable seamless handovers in networks that do not support the IEEE 802.21 Media Independent Handover standard [98], but it also allows applications to suspend a service session, save the local endpoint's state, migrate it to another network interface or even to another node, and finally resume the suspended connection from there.

Finally, the Mockets component fosters a feedback-loop-based programming model, in which applications hand the middleware the data specifying the type of delivery they need (reliable/unreliable and sequenced/unsequenced), and, in turn, Mockets constantly provides them with information on the current channel conditions. This enables software developers to design adaptive applications that can choose the delivery semantics that best fit the type of traffic produced, while tailoring the amount of data produced to the network's current state (for instance, by increasing/decreasing the generation rate, compression level, or information resolution).

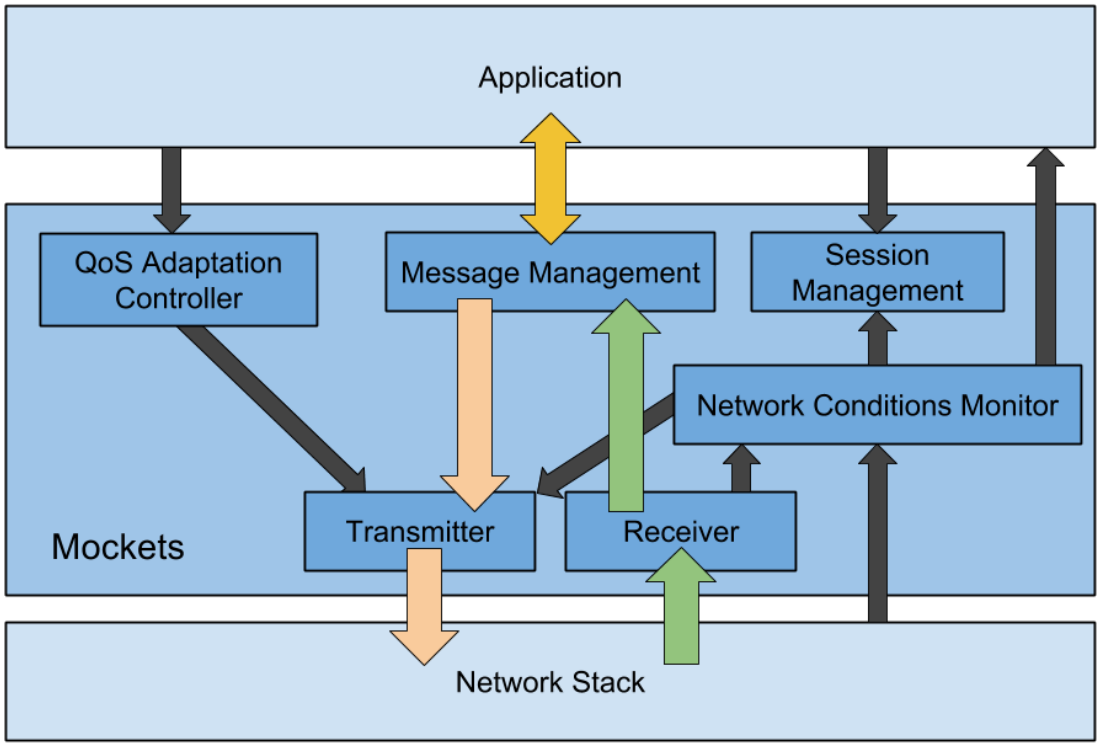

Fig. 8 Mockets Architecture

The Mockets framework is designed as an application-level library and so it is not part of the operating system kernel or network protocol stack. This allows easy porting of Mockets to many environments, and it is currently available for Win32, Linux, Android, and MacOSX platforms. This design choice supports easy deployment, platform independence, and phased utilization. It is also possible to have only a subset of the applications to use Mockets, while the remaining can continue to use TCP and UDP. This facilitates adoption and deployment, as applications can gradually migrate to Mockets from the sockets API.

3.2.2. DisService

DisService is the component of the ACM that implements robust and adaptive information dissemination between peers [36]. Nodes running DisService can exploit a publish-subscribe model designed to operate in dynamic network topologies with highly mobile nodes, like UAVs [8] [36] or public means of transportation [99], which allows applications to exchange messages via independent virtual communication channels, called “subscriptions”. By joining the same subscription, instances of the same or different applications running on any number of devices can set up a network of cooperating peers that are willing to share data.

The communication concepts implemented in DisService rely on the Opportunistic Networking paradigm and the principle of *opportunistic listening*. In accordance with them, DisService pushes to the next level the use of broadcast and multicast messages in next-generation environments. Through techniques like store-and-forward communications, the use of aggressive message-caching policies, and the possibility of prioritizing traffic on a per-message basis, multiple nodes can receive and store the data, thereby significantly increasing their availability and enabling disruption-tolerant information dissemination.

During this process, nodes keep track of each other’s contact history and resource availability, thus allowing each node to build a knowledge about its surrounding environment (namely, the “*World State*”). According to their specific (user-defined) policies, DisService instances can share information about node types (mobile, access point, sensor, vehicle, and so on) and characteristics (remaining battery life, computational power, storage, and so on). Social-, location-, and context-aware applications relying on DisService can access the knowledge collected in the World State to implement smart resource management policies.

3.2.3. Other components

NetProxy [11] [13] [19] is the component responsible for providing integration between SoA systems, COTS and legacy applications, and the ACM. Among its many features, the most notable include network protocol remapping, connection multiplexing, data compression, intelligent buffering, flow prioritization, and packets consolidation. Protocol remapping in NetProxy plays the most critical role, as it allows to forward (part of) the traffic generated by applications over Mockets and DisService transparently. This is a key step towards enabling the reuse of COTS and legacy applications in challenged networks, because it gives applications access to the features of ACM without making any change to their source code. NetProxy supports two operational modes, namely Host Mode and Gateway Mode, to fit into different network configurations and meet various user requirements. The ACM NetProxy is the result of one of my most important research strands. The reader can find more details on this component in Chapter 4.3 of this Thesis.

The ACM has a dedicated component, called Group Manager, that supports the efficient discovery of peers (and the resources and services they share) through an adaptable combination of proactive signaling and reactive search [36]. The discovery process is performed in the context of a group that peers need to join before they can take part in resource sharing or query for peers that can answer their needs. Group Manager was designed to operate effectively in a mobile environment, optimizing the discovery process to minimize bandwidth consumption, reduce latency, and favor the discovery of resourceful nodes. The Group Manager's discovery and management algorithms operate in a fully decentralized manner to withstand the high churn rate that characterizes next-generation mobile networks.

Note that, by combining the features provided by DisService (resource sharing) and the Group Manager (resource discovery), developers can easily implement functionalities like those that allow the SmartStream app on John's smartphone to retrieve missing frames from nearby nodes by means of D2D techniques (the reader can refer to the example in Section 1.3.1 and Fig. 6).

Finally, the ACM was designed to enable the policy-driven tuning of the communication function behavior. More specifically, the ACM allows users and applications to define policies that tailor the generated traffic according to their needs and current network conditions, thereby optimizing resource usage. To this end, the ACM integrates with the KaoS policy manager [100].

4. ENABLING THE SUPPORT FOR COTS AND SOA-BASED APPLICATIONS IN TENS

Military applications in TENs make a wide use of SoA-based and COTS software components to provide services to soldiers and other nodes in the system. However, those components typically rely on application protocols, e.g., HTTP, and middleware that hinder the adoption of performance optimizations techniques at the application level. SOA-based applications also make heavy use of verbose XML-based data representation protocols, which significantly increase bandwidth usage in the system. Finally, SoA and COTS components typically rely on traditional networking protocols such as TCP and UDP, which struggle to provide the necessary QoS to applications running in TENs (see Chapters 1.3.2 and 2.1).

The previous Chapter introduced the ACM, a communications middleware that provides a collection of components that address specific problems arising in extremely dynamic mobile networks. Mockets supports end-to-end communications in TENs and it is designed to perform well even in presence of frequent packet loss, temporary disconnections, and variable latency. Nonetheless, having access to the source code of SoA-based, COTS, and legacy applications is often impossible, and so rewriting them to take advantage of the features provided by the ACM might not be an option. Therefore, there is the need for a solution to bridge the gap between applications and the communications middleware specifically designed to support them in extremely challenging networking scenarios. This is essential to provide effective communication capabilities to soldiers, vehicles, UAVs, and other nodes on the battlefield.

This Chapter first presents the results of an experiment aimed at comparing the performance of several communication solutions with Mockets. Then, the Chapter discusses the need for a solution to give applications access to communications middleware, such as the ACM, in a transparent fashion. Finally, this Chapter describes NetProxy, a solution to enable the (re-)use of SoA-based, COTS, and legacy applications in extremely challenging networking scenarios, e.g., TENs. Experimental results show that the techniques implemented in NetProxy can make a more efficient use of the available bandwidth, reduce its consumption, and smooth the shape of network traffic.

4.1. Comparison of Different Communication Solutions

Researchers have developed several solutions to address the issues that traditional protocols face in challenged networking scenarios, for instance by tackling the problem of mobility at the network layer [101], or by developing strategies that aim at optimizing TCP performance on wireless networks [102] [103] [104]. Those solutions include the Stream Control Transmission Protocol (SCTP), the UDP-based Data Transfer (UDT), and Mockets (see Chapter 3.2.1).

I ran a set of experiments to compare the performance between traditional TCP-based communications, other transport-level protocols such as SCTP and UDT, and a communications middleware like Mockets over different tactical radio links [97]. In the comparison study, I used TCP CUBIC [105], available as the default TCP flavor in Linux kernels from version 2.6.19 to 3.1. It builds upon TCP BIC and changes the growth function of the congestion window from a combination of linear, logarithmic, and exponential curves to a cubic function, hence its name. TCP CUBIC shows a more efficient use of network resources in high bandwidth-delay product networks under a wide range of round-trip times and achieves better fairness with competing TCP flows [105].

4.1.1. Stream Control Transmission Protocol (SCTP)

SCTP [106] [107] is a transport layer protocol that relies on IP to provide message-oriented and connection-oriented end-to-end communications. Similarly to TCP, SCTP ensures reliable, in-sequence delivery of messages and provides flow and congestion control algorithms comparable to those implemented in TCP.

However, SCTP optionally provides order-of-arrival message delivery semantics. Unlike TCP, SCTP also supports multi-homing and multi-streaming. The former allows the protocol to take advantage of multiple IP addresses or network interfaces on the same endpoint, a feature that can improve connection survivability in case of node mobility or link disruption.

SCTP currently exploits multi-homing for redundancy purposes only, and so it does not permit to increase the maximum throughput. A “primary” address is chosen to receive data, and heartbeats are used to monitor the availability of alternate transmission paths and to test previously discovered paths. Multi-streaming, instead, ensures the concurrent transmission of multiple streams of data between connected hosts. This feature is important to avoid head-of-line blocking between independent streams.

Although Linux kernel natively supports SCTP since version 2.6, I installed the *Linux Kernel Stream Control Transmission Protocol Tools (LKSCTP Tools)* version 1.0.16 on the systems used for the experiments. LKSCTP provides a Linux user-space library that I used in the test utility I developed to access the SCTP-specific API that is not part of the standard sockets interface.

4.1.2. UDP-based Data Transfer (UDT)

UDP-based Data Transfer (UDT) [108] [109] is an application-level data transport protocol that builds on top of UDP to support distributed data intensive applications over wide area high-speed networks (WAN). The main design goal of UDT is to reach high data transfer throughput over the network, while also achieving fairness between multiple UDT flows and without starving TCP connections.

UDT is connection-oriented and provides both stream-oriented and message-oriented delivery semantics. Additionally, it is possible to configure UDT to perform reliable, partially reliable, or unreliable message transmissions, as well as sequenced or unsequenced delivery. The UDT API is very similar to the Berkeley socket API, a choice made to simplify the process of switching from TCP or UDP to UDT.

UDT supports user-defined congestion control algorithms and the multiplexing of multiple UDT connections over a single UDP flow (that is, all messages specify the same destination UDP port number). Additionally, applications using UDT can access an internal performance monitor to retrieve statistics about open connections. Finally, as an application-level library based on UDP, it is easy to port UDT-based applications to different machines and operating systems.

To develop the test utility I used in my experiments, I installed the *libudt-dev* package version 4.11. libudt-dev is a package for Ubuntu Linux systems that provides an implementation of UDT version 4 usable for application development.

4.1.3. Experimental Scenario

The primary purpose of this study is to evaluate and compare the performance of various transport protocols over some current tactical radios in a suburban outdoor environment. This is a significant, ongoing task and, in order to constrain the complexity of the problem, I made several simplifying assumptions in terms of the choice of protocols (which is by no means exhaustive), the selection of radio platforms (limited to three radios commonly

utilized in TENs: the Harris PRC-117G with the ANW2 waveform, the TrellisWare TW-400 CUB, and the Persistent Systems WaveRelay MPU4), the size of the network, and the deployment topologies.

Even with these assumptions, the results are sufficiently interesting and worth reporting. The evaluation is not comprehensive, as new aspects still have to be incorporated, and therefore future studies are needed. It is important to note that this study is not comparing the performance of different radios, and therefore I discourage the reader from using the results presented as a means to compare radio performance, since that was not my purpose.

The location where I ran the experiments is the Army Research Laboratory (ARL) Adelphi Laboratory Center (ALC) campus. The ARL-ALC campus consists of multiple buildings of varying construction, interconnected by paved roadways and pathways. This campus is considered to be representative of a suburban environment, and also provided a somewhat controlled area for this experiment. Though the elements involved in this test were stationary, there were multiple vehicles and pedestrians moving throughout the campus and the testing elements at any given time. In most of the testing topologies, trees and some foliage existed between the relay node and one of the end nodes. The tests conducted took place toward the end of March with most of the foliage not yet in bloom.

Scans of the spectrum before the tests did not reveal any interferers on the frequencies used. The antennas used during the tests were the standard antennas provided with the man portable units. In the case of the TrellisWare radios, the standard dual band antenna was used. For the TrellisWare and the WaveRelay radios, the antennas were attached to the radio and the radio was mounted to the roof of the trailer and vehicles used in the test. For the Harris 117G, the antenna was mounted to the roof of the trailer and vehicles, but the radio was remoted to the interior, using a low loss coax cable. Frequencies used were chosen so that any (even if very small) activity from idle radios would not interfere with the device under test.

Preliminary tests were conducted using the Harris and WaveRelay radios. Given the environment described above, these tests determined that a relay node would be required in order to get the desired throughput at the desired distances. This intermediate node also tested the protocol's response to a relay, which would be a common occurrence in a wireless ad-hoc network. End points for the topology were selected to test performance at various distances. Fig. 9 shows a map (courtesy of Google™ Earth) of the overall topology

for the test. I labeled and marked node locations with a circle. The distance between each marked location is measured in feet.

For all tests, the server node remained stationary at the trailer location. The relay node also remained stationary in the South Lot. The client node remained stationary during the tests, but then moved to each of the other locations to conduct each test. The VIP lot topology involved a server node located in the trailer, a relay node in the South Lot, and a client node in the VIP Lot. For the Flag Pole topology, the server and client remained in their previous locations while the client moved to the Flag Pole location for the test. During the Zahl Road topology, server and relay remained as before, while the client was located on Zahl road. In the K lot topology, server and relay remained in their respective locations, while the client node moved to the K Lot location for the test. Fig. 10 is an elevation graph of the terrain for the K Lot topology, provided to show the variability of the terrain.

Each of the tests consisted of three radio nodes and two computer nodes. The server node was always fixed at the location marked Trailer and consisted of a Dell Precision 6600 Laptop with an Intel Core i7 2820QM Processor with 8 GB RAM and a 256 GB SSD running Ubuntu Linux 14.04 Desktop 64-bit. Likewise, the client node was an identically configured laptop located at one of the four possible locations marked VIP Lot, Flag Pole, Zahl Road, and K Lot. The relay radio node, located at the position marked South Lot, was not attached to a computer node. The experiment scenario was chosen to be very simple: a bulk data transfer over a single connection from the client node to the server node. This also ensured that there was no other traffic being generated at the same time to interfere with the single data transfer.

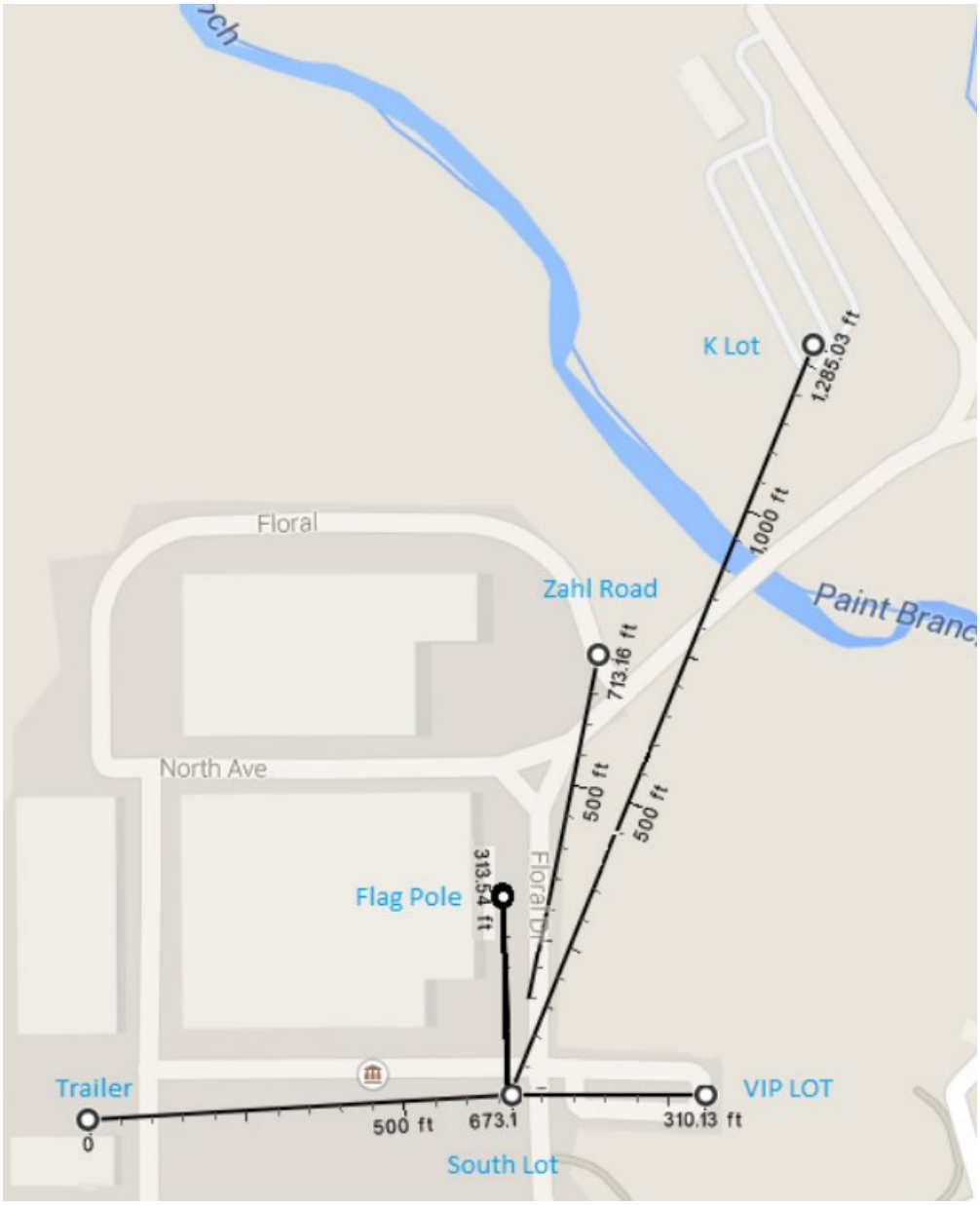

Fig. 9 Topology for the experiments

I implemented two custom applications, one for the server side and one for the client side, to exercise the selected transport protocols. The server side instantiated listeners (e.g., server sockets) for each of the protocols and simply waited for incoming connections. The client node connected to the server node using one of the four selected protocols, waited for a response, sent the size of the data to be uploaded to the server followed by the data. The client would then wait until the server acknowledged receipt of all the data (done by sending a single byte, the character “.”). The client determined the throughput by measuring the elapsed time starting after the size of the data was sent (but before the start of transmission of the actual data) and until the acknowledgement was received. The size of the data was 1024 KB for the K Lot and the Zahl Road topologies and 2048 KB for the Flag Pole and VIP Lot topologies.

I conducted each test with one of the three radios connected to the client and server nodes. The other radios were left on and idle (which should not have caused any interference given that the frequencies were deconflicted). In order to reduce temporal effects, the client cycled through each of the four transport protocols (Mockets first, followed by TCP, SCTP, and finally UDT). This comprised one iteration of the test and each test consisted of 10 such iterations.

Fig. 10 Terrain Elevation for the K Lot Test Topology

4.1.4. Experiment Results and Analysis

The results of the experiment are shown in Table 1. Each column represents one particular topology and radio combination. The radio names are abbreviated: H stands for the the Harris PRC-117G, TW stands for the TrellisWare TW-400 CUB, and finally WR stands for the Persistent Systems WaveRelay MPU4. Each row presents the results for a specific transport protocol for the topology and radio combination. For each result, the Table shows the average, maximum, and minimum throughput, expressed in KB/s, over 10 iterations, followed by the standard deviation. One exception is that, for the K Lot topology, the Table only reports the results for the Harris 117G, as the other two radios did not work reliably enough to collect data for multiple iterations. For example, when the nodes were connected to the TrellisWare radios, they were able to ping each other, but the client would time out connecting to the server, the data transfers would abort, or the data transfers would essentially remain stuck and had to be aborted after several minutes.

One of the most interesting results of these tests was the observed variability in performance across multiple iterations. I reported the maximum and minimum performance and the standard deviation to highlight this observation. Fig. 11 (a through d) shows the results, in graphical form, for each of the four topologies. The thin vertical bar shows the minimum and maximum observed performance. The larger rectangle shows one standard deviation above and below the mean performance. Note that Fig. 11a only shows the

performance for the Harris radio. As discussed earlier, the other two radios were not able to complete the test with this topology.

The variability observed was particularly surprising given four simplifications imposed on the experiment: first, there was no mobility involved whatsoever; second, all the frequencies were deconflicted prior to the test and none of the radios were being interfered with at the Radio Frequency (RF) level; third, there were only three radio nodes in the topology; and fourth, there were only two applications using the radio, with only one active data transmission. The observed performance was also surprising given the distances and the number of nodes in the topologies. As shown in Fig. 9, the distance between the server node and the relay node was fixed at 673.1 feet. The distance between the relay node and the client node varied from 310.13 feet to 1285.03 feet.

The next observation concerns the comparison in performance between the different transport protocols and communications middleware used. As Fig. 11a-d show, Mockets performed better than all the other protocols on the Harris. In particular, Mockets performed 45% better than TCP and 71% better than SCTP and UDT. This result was surprising for the Harris radio given that Harris includes a TCP accelerator built into the radio when using the ANW2 waveform. TCP did perform as the second best protocol on the Harris radio, slightly beating out SCTP and UDT. On the TrellisWare radio, Mockets significantly outperformed TCP by 109% and SCTP by 167%. It also outperformed UDT, but by a smaller margin of 22%. Thereby, the second best performing protocol was UDT, followed by TCP and finally SCTP. Finally, on the WaveRelay radio, Mockets, TCP, and SCTP performed just about the same, within 1% to 2% of each other. UDT was instead the worst performing protocol on the WaveRelay radio, with Mockets beating UDT by 48%, TCP beating UDT by 49%, and SCTP beating UDT by 51%.
Another surprising observation was that UDT did very well on the TrellisWare radio compared to TCP and SCTP, but did much worse than TCP and SCTP on the WaveRelay radio.

Table 1 Obtained results reporting Average, Maximum, and Minimum Throughput (in KB/s), and Standard Deviation for each combination of protocol, topology, and radio

		VIP Lot Topology			Flag Pole Lot Topology			Zahl Road Topology			K Lot Topology
		H	TW	WR	H	TW	WR	H	TW	WR	H
Mockets	Avg.	260,42	347,02	238,06	257,98	311,85	366,41	182,57	196,77	97,15	57,54
	Max	262,09	371,82	403,87	261,99	324,98	581,32	190,23	220,74	117,16	64,89
	Min	257,51	309,04	55,61	233,74	263,27	165,19	157,81	158,74	43,97	26,34
	St. Dev.	2,02	24,04	101,46	8,68	19,94	159,40	9,93	25,37	24,83	11,70
TCP	Avg.	194,15	175,56	219,49	191,03	147,99	385,46	97,29	85,82	103,86	57,43
	Max	199,09	227,53	293,96	201,63	226,45	489,60	126,61	100,83	154,22	100,90
	Min	180,00	99,91	158,77	177,87	55,28	267,22	43,01	70,58	64,27	25,52
	St. Dev.	7,56	44,38	39,61	10,86	53,30	68,66	28,95	10,75	27,70	21,11
SCTP	Avg.	157,01	116,98	255,01	156,53	131,80	365,72	96,38	70,86	97,77	38,90
	Max	161,67	165,29	374,27	161,60	164,39	568,26	118,79	103,81	136,12	67,78
	Min	117,65	63,27	129,40	121,52	76,56	165,52	36,12	38,55	36,08	17,01
	St. Dev.	13,84	40,50	77,37	12,43	33,82	125,85	28,76	21,07	30,23	16,87
UDT	Avg.	154,43	264,69	139,37	127,58	303,91	260,06	128,02	132,70	75,52	58,48
	Max	196,15	369,94	265,04	180,28	450,61	436,49	198,03	194,16	104,38	94,74
	Min	92,88	147,24	50,51	34,93	86,25	116,33	42,16	49,71	10,73	16,32
	St. Dev.	28,11	70,00	67,39	54,45	120,29	98,90	49,70	49,26	27,62	24,91

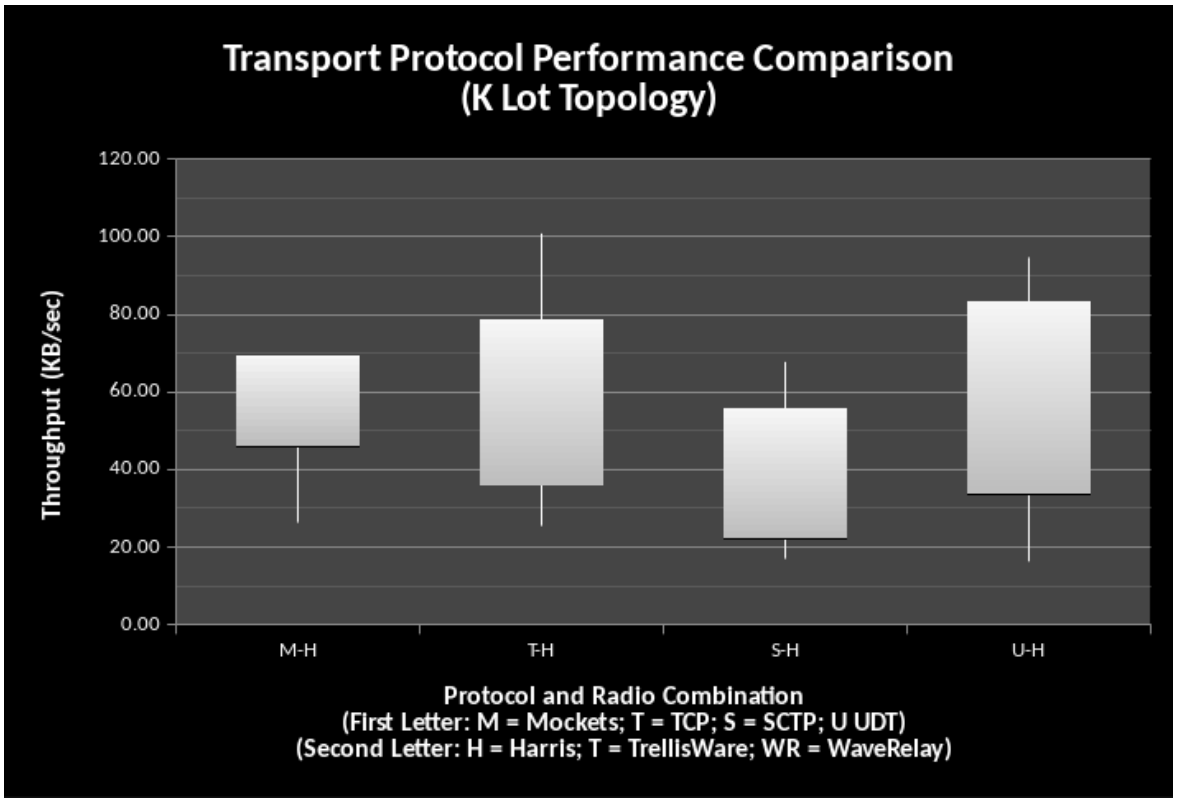

Fig. 11a Performance Results for K Lot Topology Showing Minimum, Maximum, and Deviation in Throughput (KB/s)

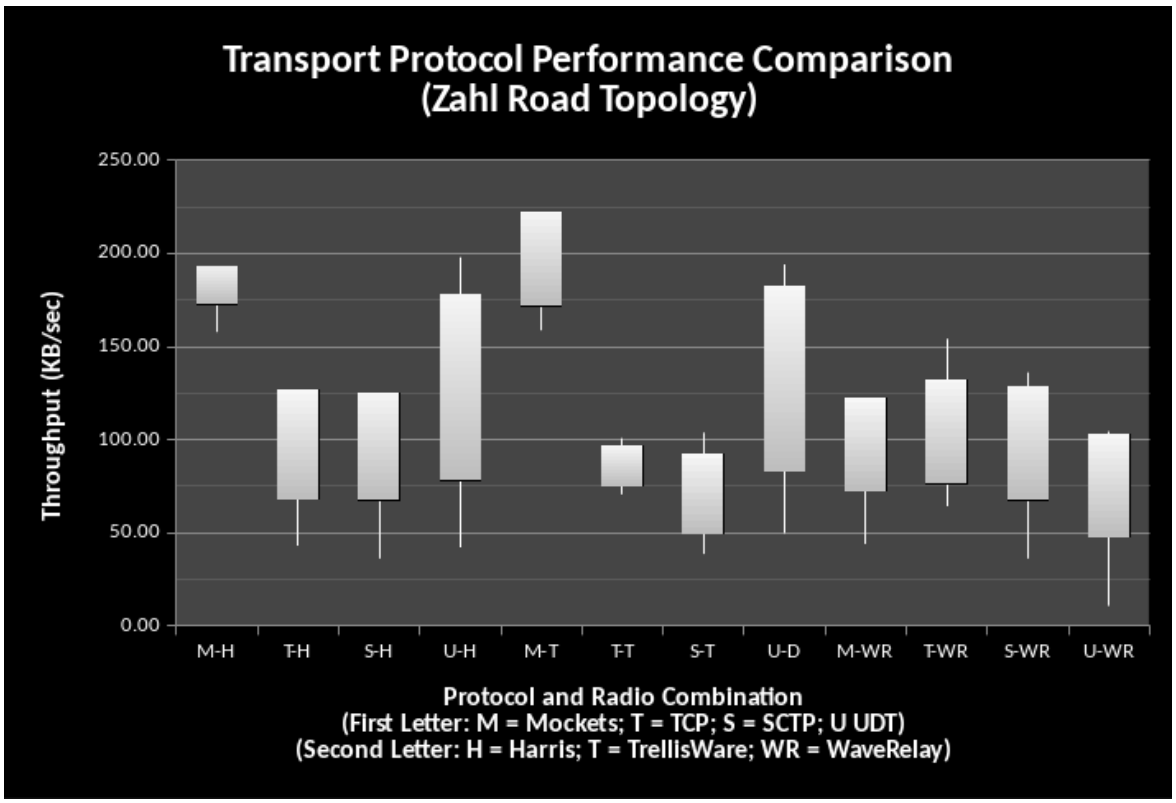

Fig. 11b Performance Results for Zahl Road Topology Showing Minimum, Maximum, and Deviation in Throughput (KB/s)

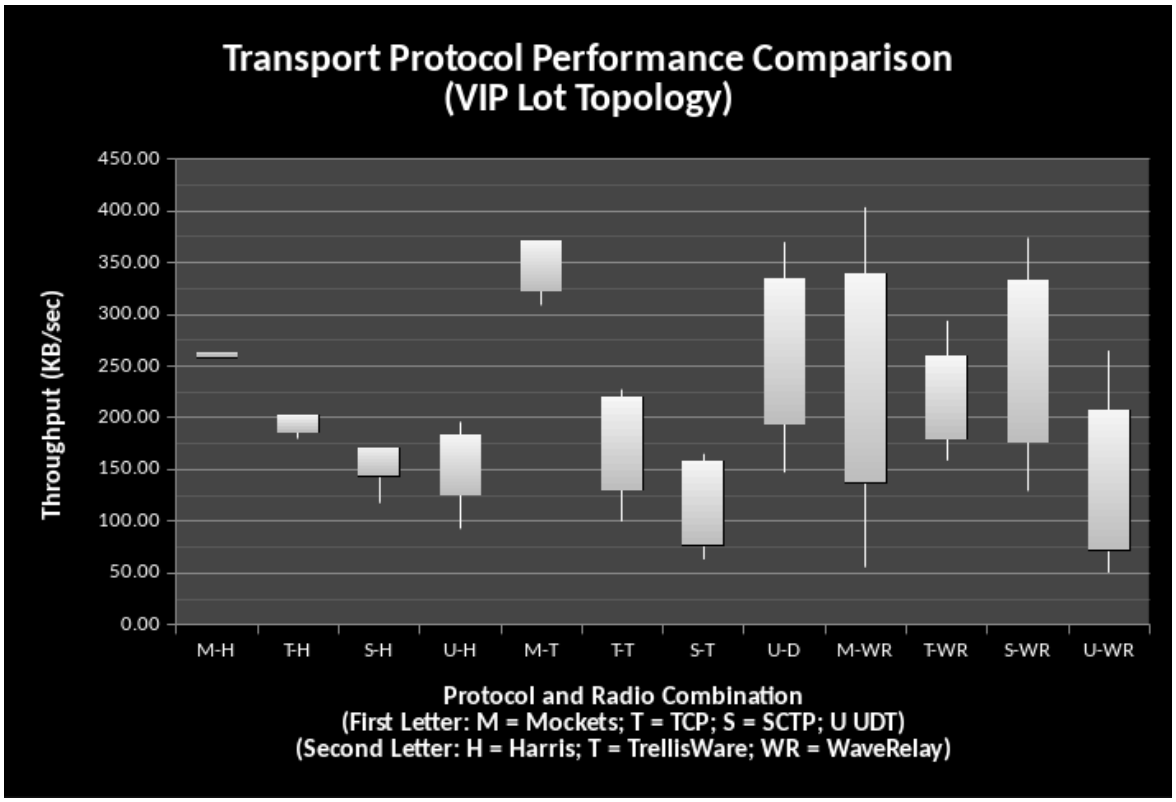

Fig. 11c Performance Results for VIP Lot Topology Showing Minimum, Maximum, and Deviation in Throughput (KB/s)

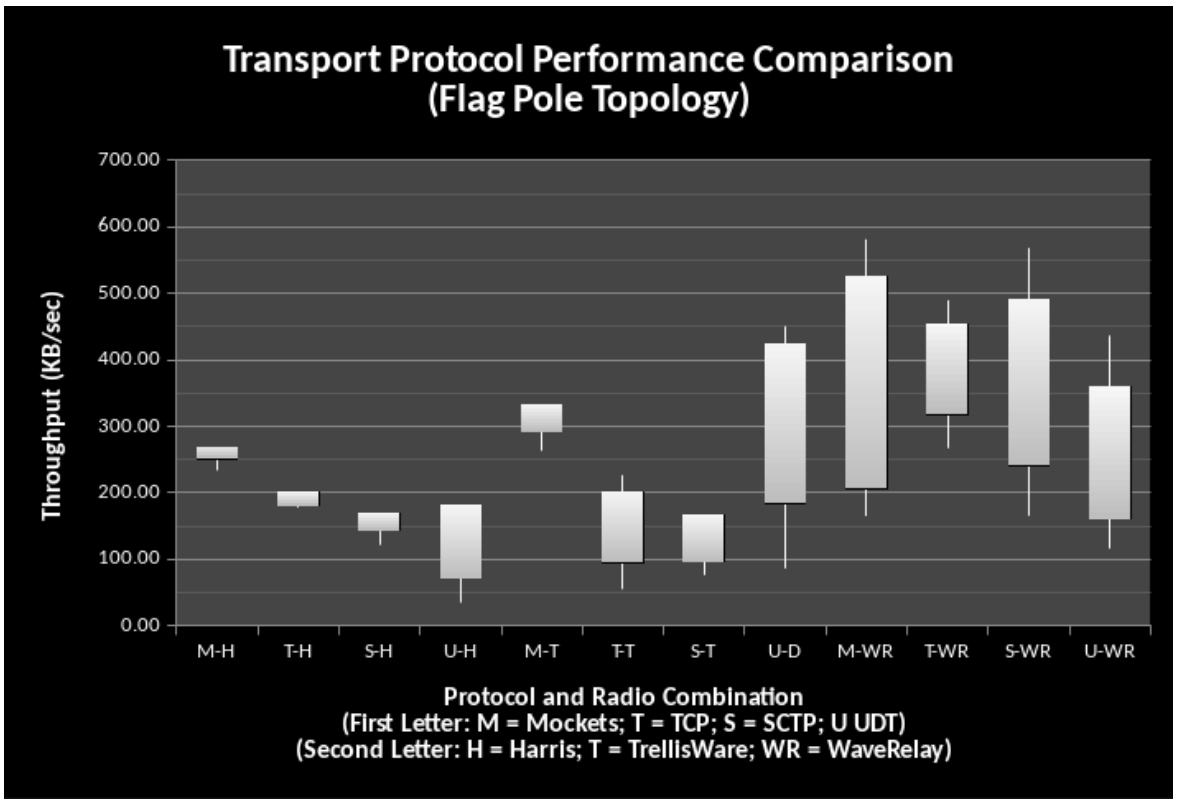

Fig. 11d Performance Results for Flag Pole Topology Showing Minimum, Maximum, and Deviation in Throughput (KB/s)

The observed variability lends support to the argument for adaptive solutions to help applications address such fluctuations in the network performance. In this particular evaluation, all the protocols were compared using the TCP model – namely a reliable and sequenced stream of bytes. However, the TCP model is highly limiting because it does not allow the transport protocol to distinguish between message boundaries, and the only type of service provided is reliable and sequenced delivery of bytes. One consequence is head-of-line blocking, which prevents subsequent messages from being delivered even if they were received in their entirety. Another consequence is that the application cannot specify different requirements for different messages. The other three solutions evaluated do provide support for message-based abstractions, which address some of these concerns. However, many legacy applications still use TCP, and rewriting them to use an alternate protocol would be difficult, expensive, and/or impossible. This condition call for alternative approaches that enable the reuse of legacy applications by enabling the access to other protocols or the communications middleware in a manner transparent to applications.

4.2. Bridging the Gap between COTS and SoA-based Applications and TENs

COTS and SoA-based applications typically perform long service sessions and adopt transport-level communication semantics based on the reliable stream model provided by TCP. Such 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 are usually equipped with more than one NIC (like 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) and, in order to achieve the best performance, applications need to be able to switch dynamically 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 and restart services 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 [110], 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 that characterize this kind of networks.

Nonetheless, the mismatch between the semantics offered by the transport-layer protocols upon which COTS applications and SoA-based services are built and those that could be reasonably provided over TENs is an even more pressing issue. At the transport protocol level, COTS and SoA-based 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.

The performance of TCP and UDP badly deteriorates in tactical environments (see Chapter 2.1), and so their limitations have a major impact on COTS and legacy tactical applications that typically leverage TCP for reliable end-to-end communications 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. This produces a critical mismatch between the semantics required by applications, those offered by transport protocols such as TCP and

UDP, and finally the QoS that such protocols can provide in challenged scenarios like a TEN.

Instead, applications running in TENs 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 strictly necessary. The importance of providing customizable delivery mechanisms in wireless environments is widely recognized, as demonstrated by the research efforts on protocols like SCTP, which supports multiple streams, multiple associations, and partial reliability mechanisms [106]. 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 [10].

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 and SoA-based applications are typically designed for wired Internet environments with the assumptions of having frequent, 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 [111]. These assumptions do not hold in TENs, thus leading to very poor performance results.

Running COTS applications on top of communication solutions, such as middleware or software frameworks, specifically designed for TENs is a very interesting solution. However, pursuing this approach is often impossible or impractical, as it requires modifications to the applications' source code. Clearly, with third party or legacy software, changing the source code is normally impossible, because either it 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 and SoA-based services over TEN-specific communication solutions [11] [19] [51]. This approach follows a school of research, dating as far back as 1995, with proposals such as I-TCP [112], Mobile-TCP [113], and the Remote Sockets Architecture [114], 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 [115], 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 improvements of application performance instead of 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 gap between application QoS requirements and the QoS levels that middleware specifically designed to support applications over TENs can provide.

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

Such approach has the advantage of being application-transparent. It works with any COTS or SoA-based application 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, legacy, and SoA-based applications in TENs.

4.3. A Proxy-based Approach: The ACM NetProxy

An important part of my research work during my PhD focused on the development and design of the ACM NetProxy. NetProxy, resting on the ideas and concepts described in the previous Chapter, transparently intercepts any (TCP- or UDP-based) network traffic generated by 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 components of the ACM, which I discussed in Chapter 3.2. Thereby, the ACM NetProxy is the component of the Agile Computing Middleware that provides transparent integration between COTS, SoA-based, and legacy applications on one side and the middleware on the other side.

Unlike TCP, the Mockets middleware does not assume to operate over low error-rate channels, but implements an adaptable congestion control mechanism that was devised specifically for the characteristic challenges of MANETs. One of the targets of NetProxy is enabling COTS and Legacy applications to benefit from Mockets without having to modify their source code. DisService, on the other hand, is a P2P information dissemination solution that enables disruption-tolerant information dissemination by relying on techniques coming from the research on Opportunistic Networking (opportunistic discovery of communications, storage, and processing sharable resources, and prediction of future availability of resources) to improve the performance of the information dissemination process in MANETs.

Mockets and DisService represent complementary solutions that address a large number of the communication requirements in TEN environments. In turn, NetProxy, whose high-level architecture is depicted in Fig. 12, relies on Mockets and DisService to provide COTS applications with communication solutions well suited for TENs. 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 final delivery to the destination. At the receiver side, another instance of NetProxy performs the inverse task. This feature is called *protocol remapping* and it is arguably one of the most important functionalities of NetProxy. In NetProxy jargon, any solution such as Mockets of

DisService, on top of which it is possible to remap application traffic, is called *"NetProxy connector"* or simply *"connector"*. Beside protocol remapping, NetProxy also provides resilience to temporary disconnections and link disruptions, stream compression, intelligent buffering, traffic filtering and forwarding, packets consolidation, flow prioritization, connection multiplexing, and network activity logging.

Users can configure NetProxy with policies that target specific communications, which can be distinguished based on the type of traffic, the protocol used, or the source/destination address pairs. Users can then update those policies dynamically whenever required, without affecting running applications. This provides a standard and centralized configuration point for all the functionalities provided by the proxy, allowing for faster and easier management of the communication flows in the network. All of these capabilities do introduce an overhead into the communication pathway, as a result of the time that the proxy spends analyzing and processing the packets. This overhead is intrinsic to any proxy-based approach. However, 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.

The ACM NetProxy is open source (available as a component of the ACM at the address https://github.com/ihmc/nomads) and was designed for easy extensibility. These properties make it possible for developers to enhance the protocol remapping functionality of NetProxy with additional connectors, such as SCTP [106] or UDT [109], with low effort. The objective of making NetProxy an extremely flexible solution while maintaining complete application transparency underlies all the design choices. By giving applications access to the features of the ACM without requiring any changes to their source code, NetProxy becomes the keystone to enable the reuse of SoA and COTS components in TENs.

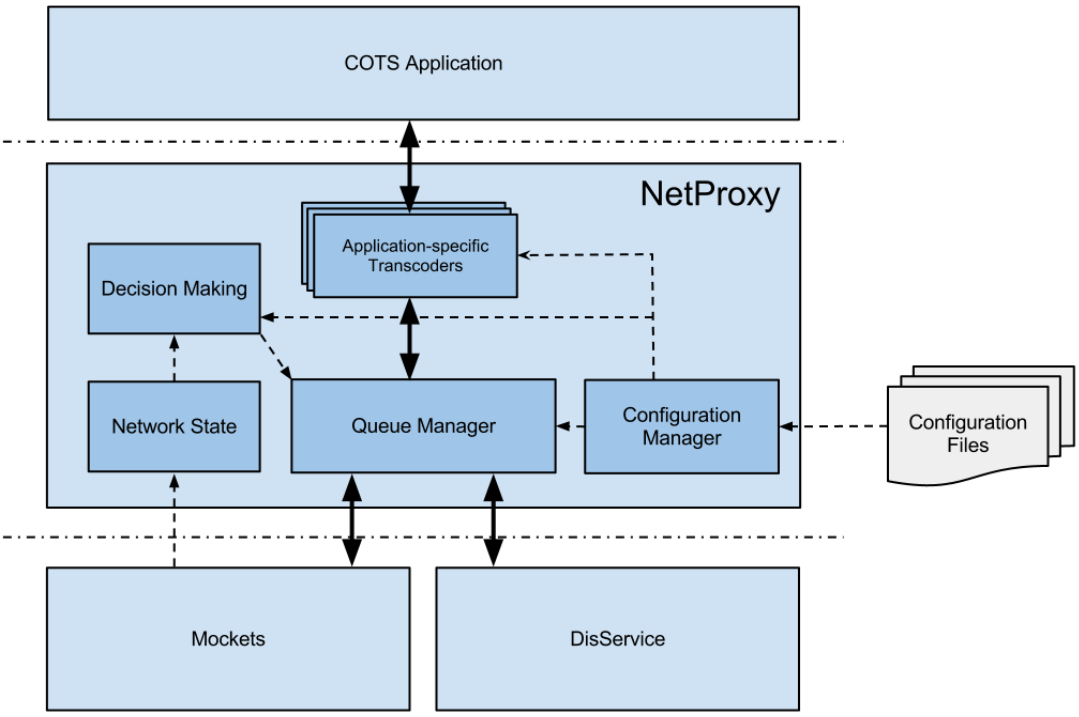

Fig. 12 NetProxy architecture and interface with applications and lower level-communication solutions

4.3.1. Design of the ACM NetProxy

The ACM NetProxy transparently intercepts network traffic generated by applications, analyzes it, and then makes decisions regarding its handling. The actions it takes are transparent to the parts involved in the communication, thus making this approach suited for legacy and COTS applications. Its most relevant features include QoS improvement and adaptation mechanisms such as network protocol remapping, traffic characterization, data compression, intelligent buffering, flow prioritization, connection multiplexing, and packets filtering and consolidation. In addition, NetProxy supports two operational modes, to adapt to various requirements and network setups.

The primary objective of NetProxy is to convey the data carried within TCP and UDP packets over the Mockets and DisService ACM components. In order to fulfill this target, NetProxy captures TCP and UDP packets, extracts their content, wraps extracted data and other necessary information in *proxy messages*, and finally passes these messages to the ACM components that will handle the delivery to destination, in accordance with the user's setting preferences. I will commonly refer to this operation as "protocol remapping". In order to maintain the transparency with communicating applications, the receiver side needs to

set up another instance of NetProxy to perform the inverse operation. Therefore, the design comprises pairwise communication between proxies located at the endpoints.

It is also possible to configure NetProxy to remap TCP over UDP and vice-versa, or to perform no protocol remapping whatsoever. Remapping TCP over UDP could be useful when there exist underlying mechanisms and functionalities that can guarantee that no packet loss or packet reordering will ever occur at the transport level, to replace the header of TCP packets with the smaller header of UDP messages. Instead, remapping UDP over TCP could be useful in presence of highly congested, long BDP links, where sending packets at rates that do not consider the network congestion level might severely affect other traffic flows. Finally, NetProxy can be configured so that it performs no remapping, which can be useful if there are specific flows that need to be forwarded unmodified. Nonetheless, it is still possible to apply other optimization techniques, such as stream compression, to those flows, thereby increasing performance even with no protocol remapping.

While running, NetProxy can also acquire knowledge on the traffic flowing in the network 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 it intercepts new packets, NetProxy analyzes their content to extract information such as 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 framework to learn about the characteristics of end-to-end connections and the available links. 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.

This way, the network traffic characterization provides 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. Network-awareness enables the selection of the network protocols and/or the components of the ACM that better match status and characteristics of the network, in terms of bandwidth availability, average latency, link reliability, nodes'

mobility, etc. Similarly, application-awareness enables 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.

The configuration of the decision-making module of NetProxy is currently read from 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.

4.3.2. Host Mode and Gateway Mode

NetProxy supports two operational modes, namely Host Mode (HM) and Gateway Mode (GM). Each operational mode allows NetProxy to be used in different network configurations and satisfy different requirements. Briefly, when NetProxy operates in HM, applications that want to take advantage of it have to run on the same node, while, when NetProxy operates in GM, applications can reside on any node of the IN.

Note that when performing traffic manipulation operations such as packet consolidation, protocol remapping, or data compression, NetProxy requires a second instance to perform the inverse operations at the other end of communications to preserve transparency. However, it is not necessary that the operational modes chosen for the two instances of NetProxy are the same. Hence, the operational mode can be chosen uniquely to satisfy other requirements, such as those placed by limited nodes' capabilities, the network topology, or any policies enforced by one or more tactical organizations, thereby keeping the flexibility of the solution high.

4.3.2.1. Host Mode

Fig. 13 shows NetProxy running in HM on a node where two applications, App1 and App2, are running and generating different types of traffic. NetProxy intercept this traffic using a virtual Tun/Tap network interface. After processing, NetProxy typically forwards the payload extracted from intercepted packets via Mockets, DisService, or other protocols to other nodes in the network. On those nodes, another instance of NetProxy will inject received data on a local instance of the virtual Tun/Tap interface for ultimate delivery to applications. NetProxy currently supports remote communications based on TCP, UDP (on which Mockets and DisService rely), and serial, but other connectors could be easily added if needed.

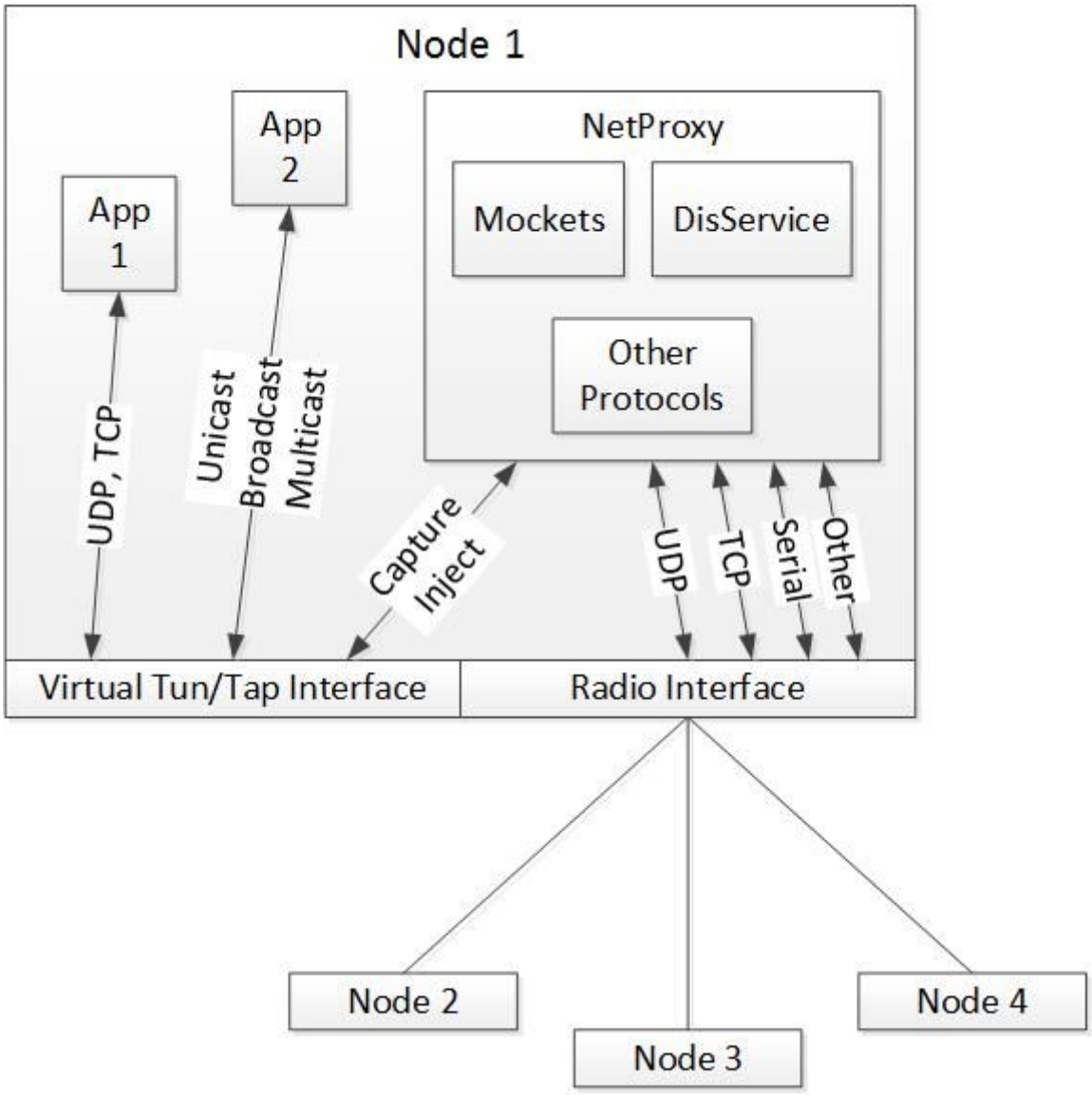

Fig. 13 Design of the NetProxy Host Mode

As an example, let us consider a simple scenario where App 1 on Node 1 needs to establish a TCP connection with App 2 on Node 2, a server application that is waiting for requests. Two instances of NetProxy are running in HM on Node 1 and Node 2, respectively, and are configured so that TCP connections are remapped over Mockets. At the beginning, App 1 attempts to open such connection with the remote host and sends a SYN packet that the local NetProxy intercepts. NetProxy responds to the SYN by injecting a SYN ACK on the local virtual Tun/Tap interface as if it was coming from the remote application. This sets up a new "*local connection*" between App 1 and the NetProxy running on that node (NP1), as opposed to the "*remote connection*" that NP1 will open with the NetProxy running on Node 2 (NP2). Because of the given configuration, Mockets provides the remote connectivity between NP1 and NP2. In case a Mockets connection was already available between the two endpoints, NP1 and NP2 would reuse the existing connection. On top of that remote connection, the two NetProxy instances initialize a new "*logical connection*" between App 1 and App 2. The latest abstraction is necessary to distinguish the traffic that belongs to different applications, e.g., between App 1 and App 3, or App 3 and App 4, running on the same nodes.

When NP2 receives the request to initialize a new logical connection from NP1, NP2, in turn, establishes a local TCP connection with App 2 over the virtual Tun/Tap network interface by sending a SYN packet as if it was coming from App 1. Once all connections are setup, NP1 can finally send the data coming from App 1 to NP2 over the remote Mockets connection. On the other side, NP2 receives data from the Mockets middleware and forwards it to App 2 over the local TCP connection. From this moment on and until applications close their endpoint of the local TCP connections, communication is full duplex. It is relevant to note that remote connections are completely independent from the connections opened and closed by the applications, which means that NetProxy can keep them open even when all logical connections have been closed, or even create them beforehand to reduce the delays due to connection establishment. Finally, it is also possible to have a different connector for each direction to carry traffic between two NetProxy instances. This might be useful when the specific network configuration causes the path from A to B to be very different from the path that packets follow to reach A from B.

Therefore, HM requires all nodes hosting SoA-based or COTS applications that need to benefit from components of the ACM to run a copy of NetProxy locally. In addition, HM also necessitates the installation of a virtual Tun/Tap network interface on each node and a proper configuration for each instance of the proxy.

4.3.2.2. Gateway Mode

GM differs from HM in the way the ACM NetProxy intercepts packets and in the role that the node running the proxy needs to assume in the network. When operating in GM, the ACM NetProxy needs to run on nodes equipped with at least two network interfaces. The proxy uses one of them, named "*Internal Network Interface*" (INI), to intercept all packets generated by nodes within a portion of the network, called "*Internal Network*" (IN), and addressed to nodes that do not belong to the same subnetwork. NetProxy instead uses the other interface, labelled "*External Network Interface*" (ENI), to send the captured traffic to remote destinations (in the "*External Network*", EN) after processing is complete. The naming of the two operational modes comes from the network configuration to which they adapt the best: due to its requirements, it is convenient to run NetProxy in GM on network gateways, while NetProxy in HM can run on any node of the network. Because of this, in the rest of Chapter 4 I will sometimes refer to the nodes that run NetProxy in GM simply as "proxy gateways".

Fig. 14 shows the schema of two nodes running NetProxy in GM: NP1 and NP3. The IN of NP1 contains three nodes connected to Node 1 via Ethernet or Wi-Fi; the IN of NP3, instead, has only one node also connected to Node 3 via Ethernet or Wi-Fi. The EN could be any WAN, MANET, or SATCOM network. Another node, Node 2, is connected to the EN and running another instance of NetProxy, NP2. The figure does not specify if NP2 is running in HM or GM and, for the correct functioning of the system, it is not relevant.

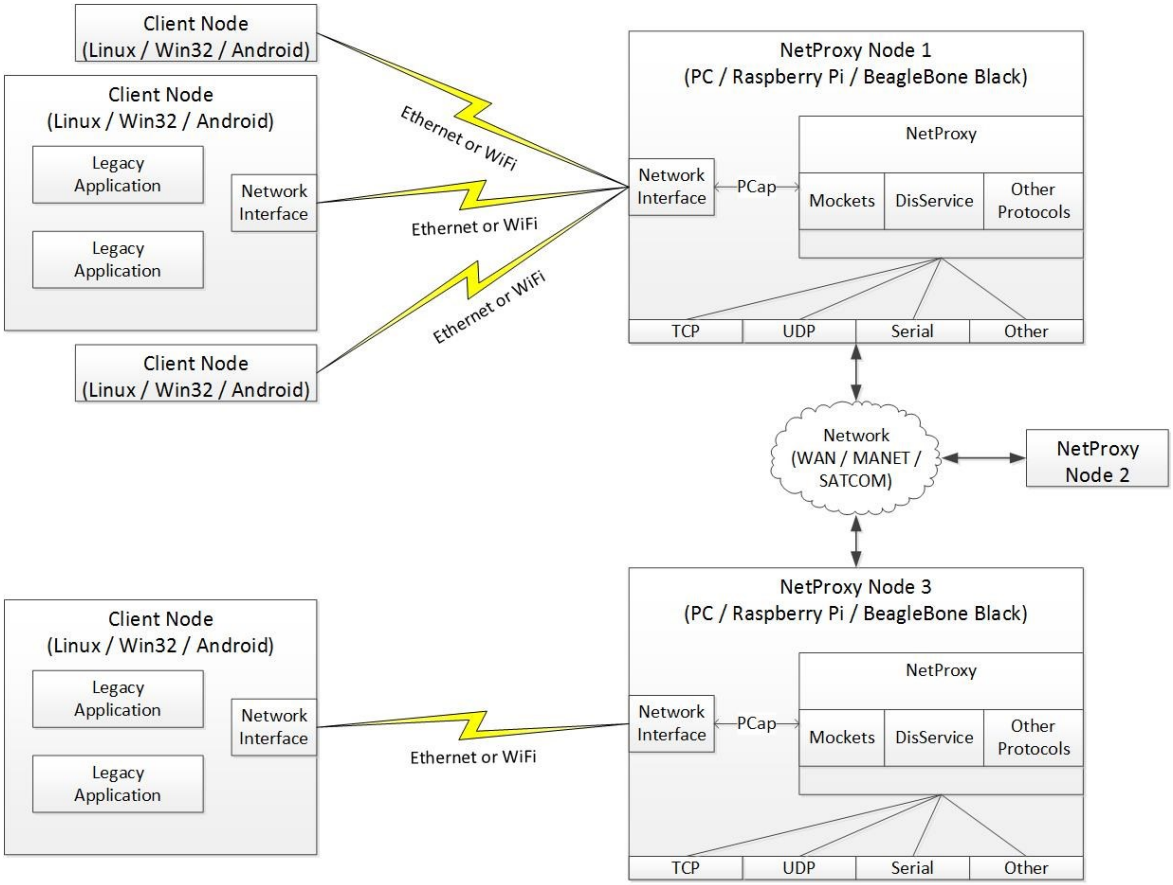

Fig. 14 Design of the NetProxy Gateway Mode

Compared to HM, GM does not require the installation of any virtual network interface, nor does it need to run a copy of NetProxy on each node that hosts SoA services or COTS applications that require access to the features provided by the ACM. Furthermore, GM reduces the amount of configuration required, since there is only one instance for each sub-network, as opposed to one for each node. When running in GM, the ACM NetProxy uses libpcap (http://www.tcpdump.org/) to sniff packets on both the internal and external network. To intercept and process network packets transparently, the ACM NetProxy also needs to implement a customized version of the Address Resolution Protocol (ARP).

Another very important difference is that GM allows NetProxy to build a comprehensive picture of the traffic that nodes in the IN generate, what services they request, and what kind of data they transfer. On the other hand, HM only permits the proxy to learn about local services, thus limiting the efficiency of the decision-making process. Moreover, often the ENI of gateway nodes in TENs is very different from the INI, providing links with completely different characteristics 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 links more

easily; this information is extremely useful to 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 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 it could be directly integrated into the wireless device for complete transparency.

Fig. 15 represents the setup of a TEN where two nodes take on the role of proxy gateway and run NetProxy in GM and other two nodes run 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 wheeled vehicle, the INI is connected to the MANET and the ENI is connected to the SATCOM terminal. Consistently with the design principle of achieving the highest level of transparency, it is not required to run NetProxy or apply changes to the configuration on any node residing in the INI of the wheeled vehicle or the OC.

Three different cases might occur. The first case is very simple, and it refers to two nodes in the IN or EN that need 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 sniffing 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, which is neither located behind another proxy gateway nor on a node that runs NetProxy in HM. When this event occurs, NetProxy intercepts the ARP request sent by the node that wants to start the communication, 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 specified in the header of frames (level 2, or MAC level, of the ISO/OSI reference stack) is that of the ENI;
2. the IP destination address matches that of some node in the other subnetwork.

The reason I chose to modify the SHA field in ARP packets is twofold: for clarity's sake, as it makes it straightforward to identify packets that NetProxy has to forward 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 (I will also refer to it as "source") in the IN that needs to send data to a remote host (also, "destination"). If the 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 the ACM NetProxy in HM, or it is located behind another proxy gateway running NetProxy in GM. For brevity, in my example I will only consider the 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. Following a proxy message sent from the NetProxy at the source side to the proxy at the destination side to inform it about the new logical connection, the latter NetProxy will generate an ARP request to obtain the MAC address of the destination node's NIC. Once the address is resolved, the end-to-end connection can be established and the nodes involved can take advantage of the features of NetProxy and the ACM, in accordance with the settings written in the configuration files. 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 would simply forward packets from the IN to the EN, where the network gateway can take care of their delivery. This allows complete transparency from the point of view of the applications and reduces the amount of configuration necessary to ensure communications between remote nodes.

The NetProxy running in GM requires that only the ENI has an IP address assigned to it, and it will make use only of the MAC address of the ENI when sending ARP packets. This design choice reduces the consumption of IP addresses, which might be a limited resource in some complex network configurations, especially when many different parties are involved, and, at the same time, it avoids that the proxy gateway OS replies to ARP requests generated in the IN, thereby leaving the control to NetProxy.

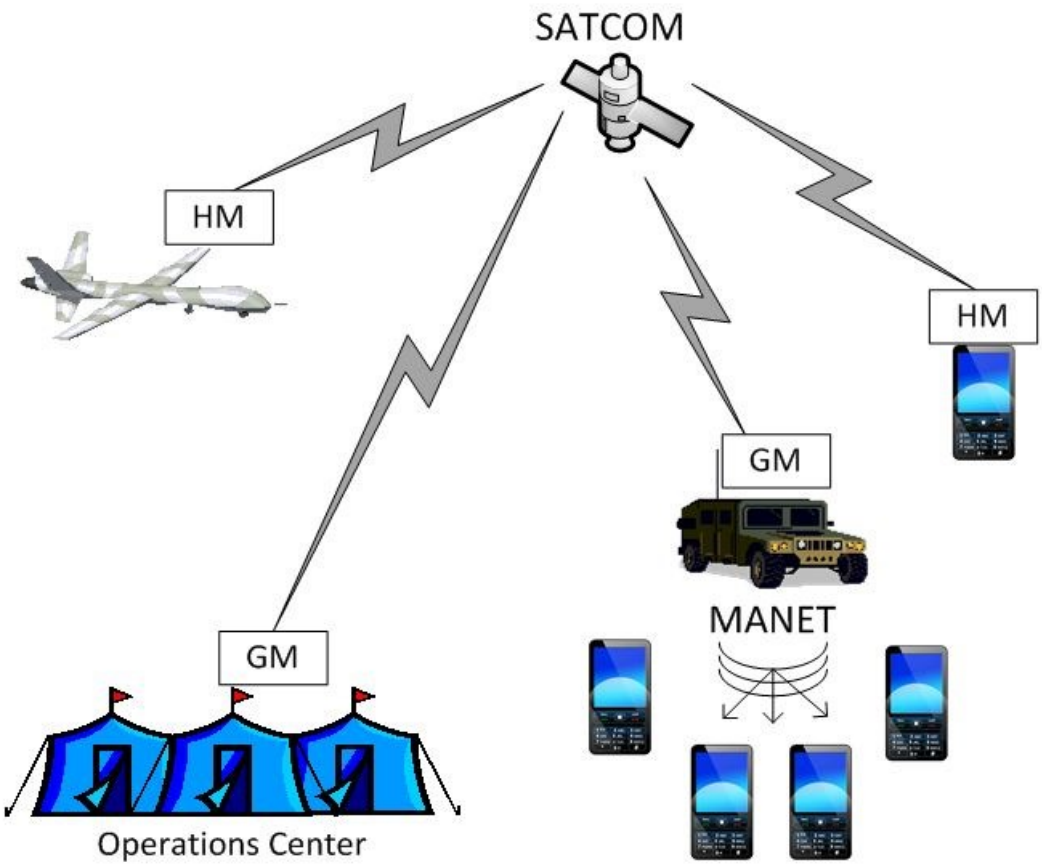

Fig. 15 Tactical network scenario with NetProxy running in Host Mode (HM) and Gateway Mode (GM)

4.3.3. Architecture and Implementation Details

The ACM NetProxy executes in user space and may be installed as a service or a daemon process. Depending on the operational mode in which it runs, NetProxy relies on either a virtual Tun/Tap network interface (when operating in HM) or a libpcap interface (when operating in GM) for packet capturing/sniffing.

In case the system is configured to run NetProxy in HM, the virtual Tun/Tap interface needs to be assigned an IP address (henceforth, I will refer to the IP address of a virtual network interface as “virtual IP”, or “virtual address”) and a subnet mask, which identifies a virtual network. The virtual IPs of NetProxy nodes that operate in HM have to belong to the same virtual network for applications running on those nodes to communicate through NetProxy. This makes sure that packets addressed to remote virtual IP addresses are sent out on the virtual Tun/Tap interface. Similarly, to communicate with nodes in the IN of a NetProxy

operating in GM, the virtual IP address of nodes running NetProxy in HM needs to belong to the same virtual network of the nodes in the IN.

An application that wants to open a connection to a remote instance going through NetProxy needs no modification to its source code; it can use standard TCP/UDP system calls to generate packets addressed to destination. NetProxy intercepts those packets, processes them, and extracts the data. If protocol remapping is configured, NetProxy will forward extracted data over the new protocol or middleware component, such as Mockets or DisService, which will in turn undertake the task of delivering data to destination. Packet capturing occurs on either the virtual Tun/Tap interface or the libpcap interface, depending on the operational mode chosen. When configured in GM, nodes in the IN might produce ARP requests before sending the data over TCP or UDP packets; in this case, the NetProxy replies to those packets as described in Chapter 4.3.2.2.

Fig. 16 shows the architecture of NetProxy with its main components. The Local Receiver (LR) is a thread responsible for capturing packets from either the virtual network interface or the libpcap interface. The LR continuously listens on the virtual interface for new packets and processes them. Depending on the type of packet captured and the configuration settings, the LR will take different actions.

First of all, if NetProxy is operating in GM, it sniffs a packet addressed to a node in the EN, and no processing is configured to take place for that type of traffic, then NetProxy simply forwards the packet on the EN. This guarantees complete transparency for traffic that does not need any processing. Differently, if the proxy intercepts a UDP unicast packet, the LR buffers it and a second thread will take care of packet consolidation, remapping, compression, and delivery according to the configured options. Instead, in case of UDP multicast/broadcast packets, the NetProxy can be configured to leverage DisService for delivery, or it is possible to specify a list of remote nodes to which send the data using Mockets, TCP or UDP unicast. Alternatively, when running in GM, NetProxy could also be configured to simply forward multicast/broadcast UDP packets onto the ENI. NetProxy can also handle pings and the most common ICMP messages; in this case, protocol remapping is necessary and common choices are UDP or unreliable and unsequenced Mockets.

Finally, when NetProxy receives TCP packets, the LR processes them according to the settings in the configuration files and the options specified in the TCP header. Usually, data is buffered in memory and managed by the Flow Manager (FM). In order to do this, the LR implements a version of the TCP protocol adapted to the features implemented by NetProxy. In addition, when applications try to establish a new TCP connection, the LR is

responsible for starting the procedure that will generate and send a message to the remote NetProxy to inform it that a new logical connection needs to be established. That message will contain all necessary information to identify an application that reside on a node in the IN of the remote proxy or directly on the proxy node, depending on the operational mode.

The FM maintains the status of connections with local applications and remote NetProxy instances in a Flows Table. For every TCP connection open with a local application, the Flow Table contains an entry identified by a unique pair of integers (each one called “connection identifier”, or “connection ID”) that couples the local connection to the respective one at remote side, where a symmetric pair of integers is replicated in the local Flows Table. Each pair of identifiers identifies a single logical connection over a remote connection between two NetProxy instances. In fact, the number of connections opened between two proxies does not necessarily mirror the number of TCP connections opened with local applications, because NetProxy normally multiplexes multiple TCP flows over the same connection, which might be provided by Mockets, TCP, UDP, or other connectors. Every time two instances of NetProxy exchange packets in the context of a logical connection, data get wrapped within proxy messages where two fields contain the identifiers for a specific pair, to discriminate data belonging to different TCP logical connections. In addition to connection identifiers, entries of the Flows Table store all necessary information to keep the state of the local TCP connections (i.e., timeouts, counters, etc.), incoming/outgoing buffered data, remote proxy addresses, and local and remote IP addresses and port numbers.

The main task of the Remote Transmitter (RT) is to read data stored in entries in the Flows Table and send it to remote NetProxies, through the Protocol Adapter (PA), for final delivery to applications. The PA implements all the functions to encapsulate data in different protocols, depending on the configured options. Therefore, the PA is the only component of NetProxy that interacts with the OS Socket API or other ACM components, like Mockets and DisService. Note that data is not taken as are from the Flows Table and sent directly to remote NetProxies. If compression is enabled, NetProxy compresses data before handing them over to the OS or another ACM component for transmission. Additionally, if NetProxy is handling data that were carried within UDP messages and the proper configuration settings are enabled, NetProxy performs packet consolidation prior to transmission. Finally, when an application closes its half of the TCP connection, the RT is also responsible for notifying the remote proxy of the completed transmission by sending a dedicated proxy message. Another dedicated proxy message serves to inform a remote NetProxy of local applications that reset any open TCP connection.

The Remote Receiver (RR) is responsible for receiving proxy messages and performing different actions depending on the type message it received. Possible actions include opening new TCP connections with local applications, closing one half of a TCP connection, resetting a connection, and so on. The RR also appends actions to the corresponding entry in the Flows Table; this ensures that, if TCP packets get lost for any reason in the IN or in the virtual Tun/Tap interface, retransmission is possible. Instead, when the RR receives proxy messages that contain data, their content is decompressed (if any compression was applied) and appended to the corresponding entry in the Flows Table, but the RR does not take care directly for its delivery.

The Local Transmitter (LT) is the component that takes care of sending data and retransmitting lost TCP packets to local applications. Its main task is to forward data to local applications reliably and in a sequenced manner. For this purpose, the LT generates and sends TCP packets in accordance with the timeouts and rules of the protocol. Any necessary information to handle the local TCP connections with applications is stored in the Flows Table, which the LT updates and accesses through the FM.

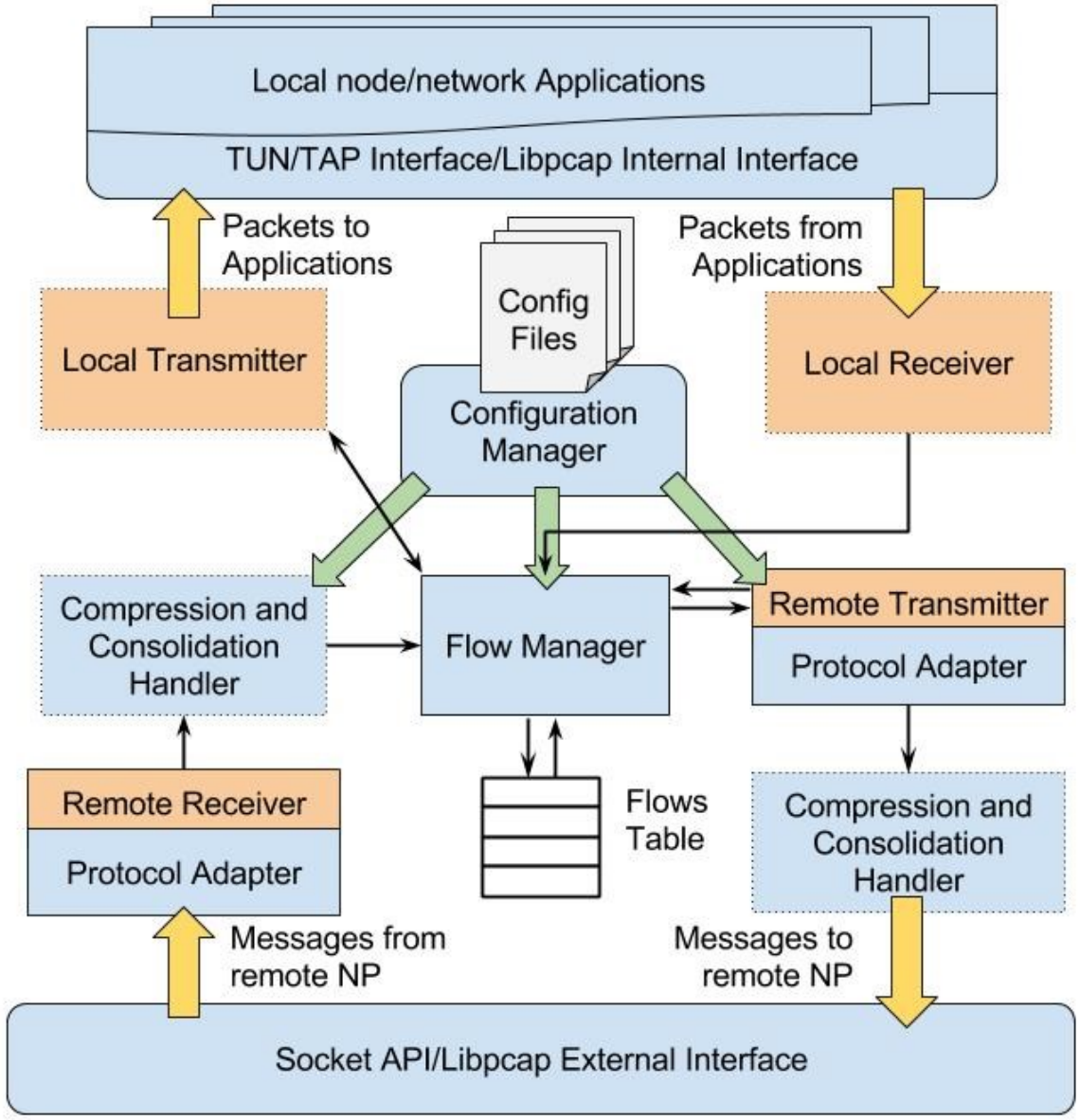

Fig. 16 Architecture of NetProxy

4.3.4. Experimental Results

To demonstrate the effectiveness of NetProxy in supporting the reuse of COTS and SoA-based applications in TENs, this Chapter presents the results obtained from three different experiments. The experiments are designed to reproduce and illustrate common issues that COTS applications exhibit in TENs and reflect typical network configurations and traffic loads.

For the first two experiments, whose goal was to evaluate the performance improvement gained by remapping TCP connections over Mockets using NetProxy, I ran several experiments in an emulated environment that permits to reproduce the characteristics of

TENs. More specifically, I used an enhanced version of the Mobile Ad-hoc Network Emulator (MANE) (http://www.nrl.navy.mil/itd/ncs/products/mane), a tool designed to reproduce the characteristics of unreliable environments such as TENs and set up and configure links between the nodes involved in the tests.

The nodes are part of the NOMADS testbed, which comprises 96 servers connected through a 100Mbps Ethernet LAN. The hardware configuration of 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 of each link, thus allowing the evaluation of different systems and protocols in a reproducible, laboratory-controlled environment. The reliability parameter is complementary to the Packet Error Rate (PER): a 90 percent reliability value is equal to a 10 percent PER value.

4.3.4.1. Remote Web Service Experiment

For the first experiment, I wrote a simple client application that generates an HTTP-based SOAP request, sends it to a Web Server located on another node of the testbed, and waits for the response, whose size for the purposes of the tests was fix to 2611 Bytes (~2.55KB). The client application is also responsible for measuring the throughput. NetProxy was running in HM on both the client and server nodes. As for the emulator configuration, I kept the bandwidth of the link stable at the value of 1 Mb/s throughout the experiment, while I varied the reliability values between 87, 90, 93 and 95 percent. I configured the client application to repeat the SOAP request 50 times with the same link conditions before I changed the configuration of MANE to set the next reliability value for the link. For the first batch of tests, only the protocol remapping feature of NetProxy was enabled, to remap TCP over Mockets. For the second batch of tests, I also enabled the QoS enhancement feature of stream compression, to compare performances between Zlib (http://www.zlib.net) and LZMA (http://www.7-zip.org/sdk.html), two highly efficient, lossless compression algorithms available for free as open source, and to evaluate the performance improvements that can be achieved with data compression.

Fig. 17 shows the results obtained when running the experiment described above and connecting client and server using TCP, TCP via NetProxy, or remapping TCP over Mockets using NetProxy (“Mockets via NetProxy”, in the figure). The higher throughput achieved by the third solution clearly stands out from the graph. It is also worth noting that TCP via NetProxy performs better than plain TCP; the reasons are manifold. First of all, several traditional SOAP implementations, such as Apache Axis 2 (http://axis.apache.org/axis2/java/core/), by default send open a new TCP connection to for

each SOAP request. 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 when the traffic flows over a TCP connection opened between two instances of NetProxy, because NetProxy multiplexes all traffic directed to the same proxy onto the same connection, which is kept open between consecutive requests. Furthermore, NetProxy buffers the content of multiple TCP packets into a smaller number of larger segments. Thanks to this feature, NetProxy typically sends less packets to transmit the same amount of data, thereby reducing the protocol overhead.

Fig. 18 shows the results obtained running the same experiment after enabling compression in NetProxy. Again, I configured the client application to repeat the SOAP request 50 times, this time fixing Mockets via NetProxy as the connector, but varying the compression algorithm. The figure shows that enabling compression in NetProxy produced high gains in the measured throughput. This is due to the verbosity of the HTTP and SOAP protocols, which causes compression to be extremely effective. Note that, despite the higher compression ratio of LZMA compared to Zlib, using Zlib with NetProxy resulted in higher throughput. This is due to the greater computational resources that LZMA requires.

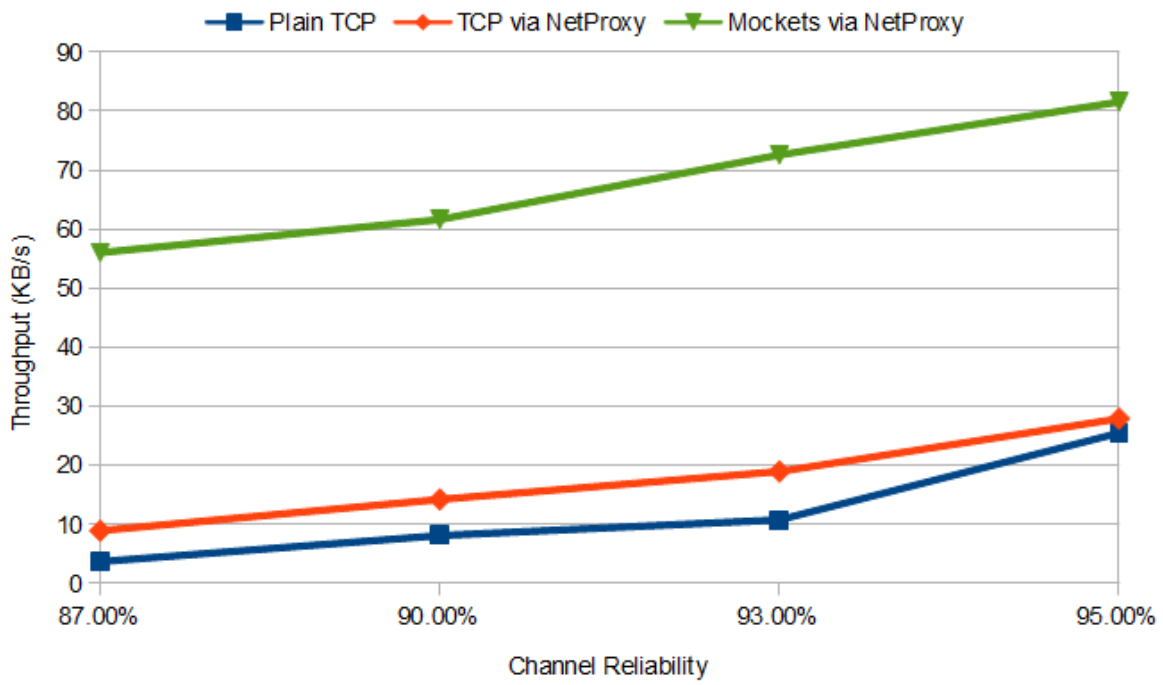

Fig. 17 Measured throughput when running the remote web service experiment with TCP (in blue), TCP via NetProxy (in red), and Mockets via NetProxy (in green)

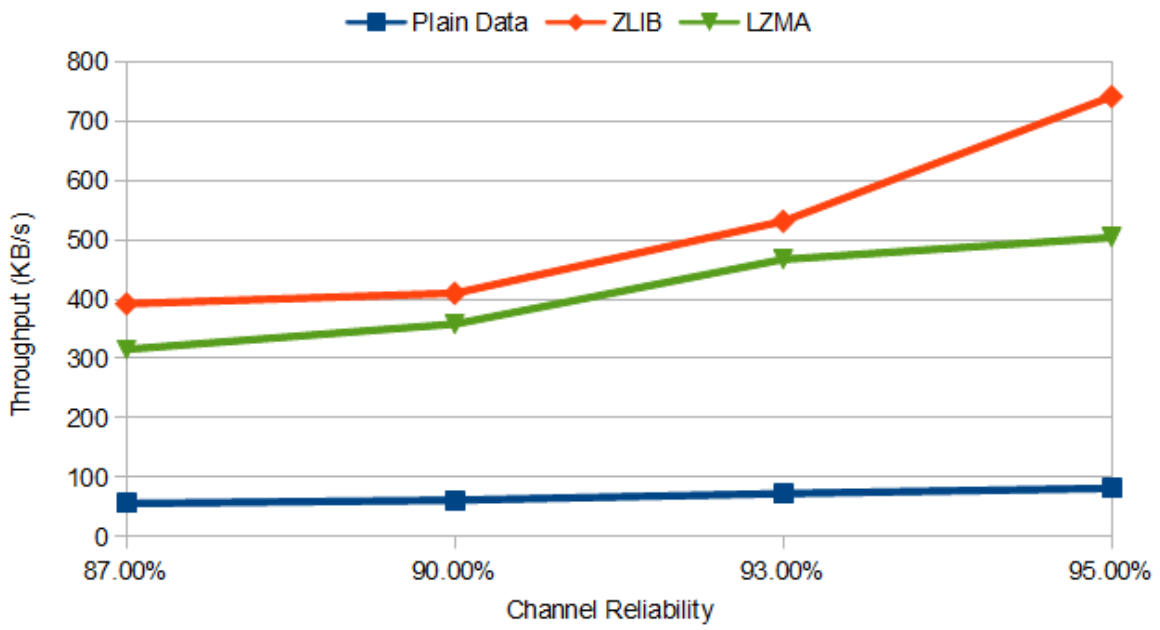

Fig. 18 Measured throughput when running the remote web service experiment with Mockets via NetProxy and no compression (in blue), Zlib compression (in red), and LZMA compression (in green)

4.3.4.2. Publish-Subscribe Service Experiment

For the second experiment, I used Apache Qpid (http://qpid.apache.org/) to build a basic scenario based on a service designed in accordance with the publish-subscribe model. Fig. 19 depicts the configuration I used for the experiment. Two instances of Qpid (Qpid A and Qpid B) were running on two nodes of the testbed and two applications, publisher and subscriber, were running on a third node (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 deliver the published messages to the subscriber, Qpid A set up a connection with Qpid B. NetProxy was running in HM on the two Qpid nodes, so that communications between Qpid A and Qpid B could go through the proxy.

The publisher, subscriber, and the two Qpid instances used TCP connections over the 100Mb/s Ethernet LAN provided by the NOMADS testbed; MANE controlled the link between the two Qpid nodes. The nodes I used for this second experiment are CentOS 6.2 Linux virtual machines (VM), each VM running on an Ubuntu 8.04 LTS server. Again, I used MANE to fix the link bandwidth to 1 Mb/s vary the reliability value between 85, 90, and 95 percent. Instead, MANE did not affect the links between the two Qpid nodes and the node running publisher and subscriber, which therefore kept their intrinsic features (100Mb/s Ethernet links). In order to collect results, the publisher application published 10 copies of

messages of 100, 256, and 512 KB in size for each reliability value, totaling 30 messages for each value of the reliability parameter, and then the subscriber application measured the throughput upon reception of each message. I configured the proxy only to remap TCP traffic over Mockets; no additional QoS enhancement feature was enabled.

Fig. 20 shows the results of this second experiment. Each figure shows the average throughput obtained with messages of 100 KB (a), 256 KB (b), and 512 KB (c) in size against all reliability values set by MANE. While the performance of both solutions decreases with a reduction in link reliability, this trend for TCP is much steeper than for the solution where TCP is remapped over Mockets using NetProxy. Table 2 contains a more detailed analysis of the experiment's results. As shown, in almost all tests, the throughput variability measured when using Mockets and NetProxy, expressed as the throughput standard deviation, was lower than the one measured when TCP was used.

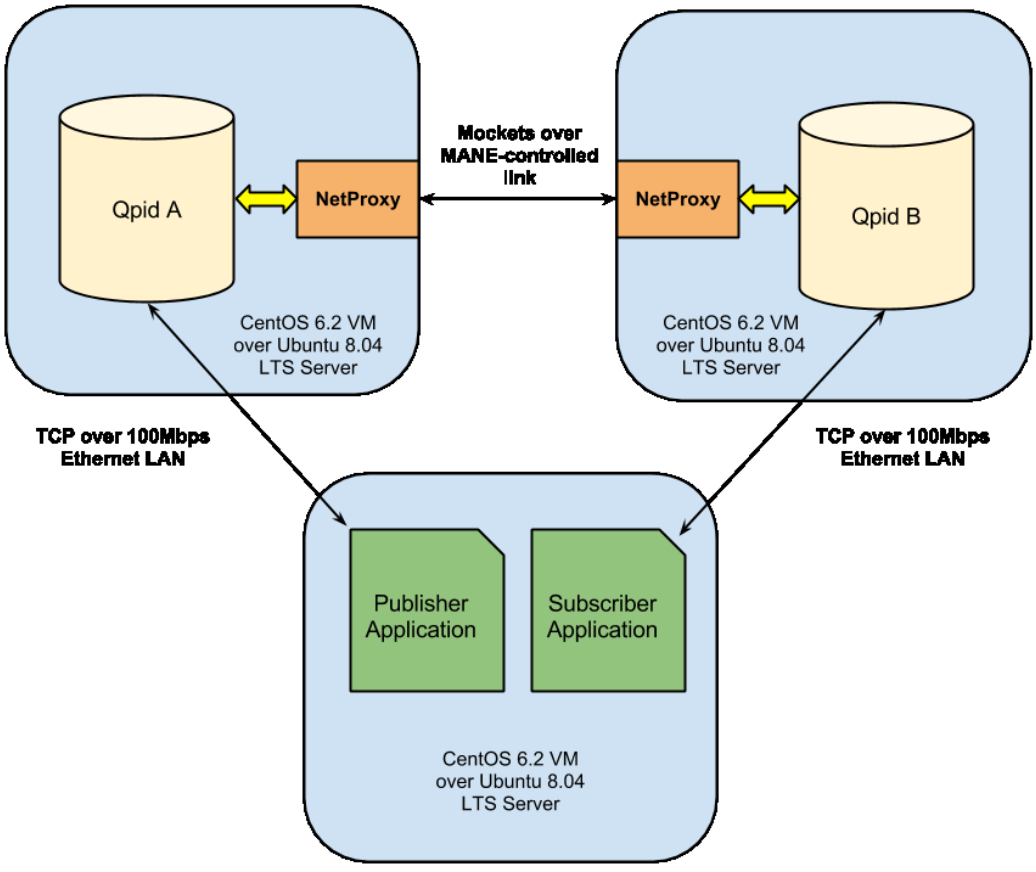

Fig. 19 Configuration of the Publish-Subscribe experiment with Apache Qpid

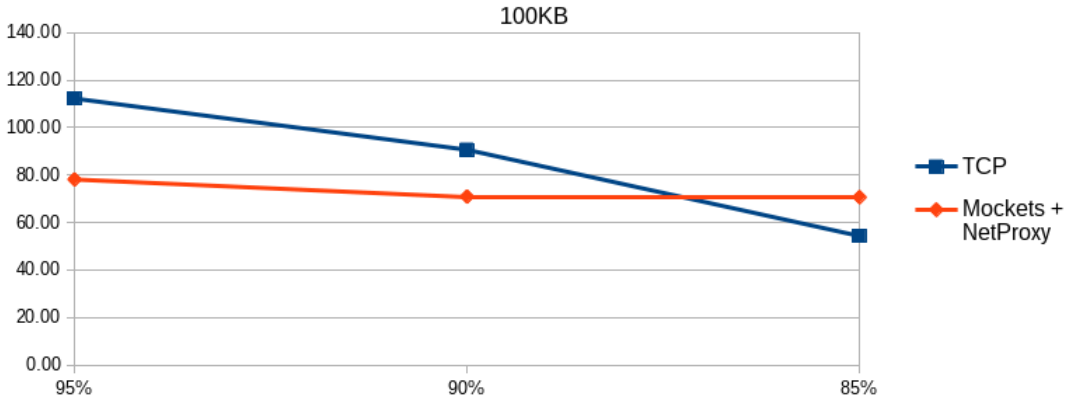

Fig. 20a Average measured throughput (in KB/s) for messages of 100 KB and channel reliability set to 95, 90, and 85 percent. Tests run with TCP and Mockets via NetProxy

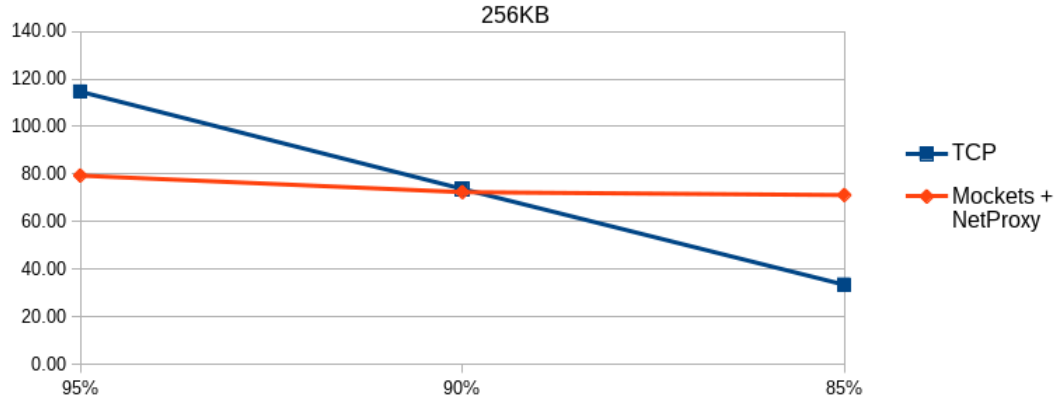

Fig. 20b Average measured throughput (in KB/s) for messages of 256 KB and channel reliability set to 95, 90, and 85 percent. Tests run with TCP and Mockets via NetProxy

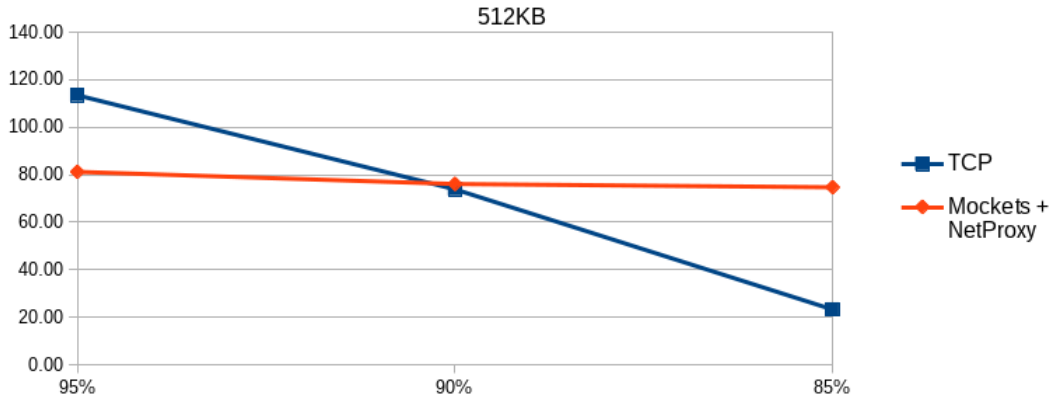

Fig. 20c Average measured throughput (in KB/s) for messages of 512 KB and channel reliability set to 95, 90, and 85 percent. Tests run with TCP and Mockets via NetProxy

Table 2 Average Throughput and Throughput Standard Deviation measured during the Publish-Subscribe experiment performed with Apache Qpid with messages of size: a) 102412 bytes; b) 256012 bytes; c) 524300 bytes

a)

Message Size (Bytes)	102412					
Reliability	95%		90%		85%	
Connector	TCP	Mockets + NetProxy	TCP	Mockets + NetProxy	TCP	Mockets + NetProxy
Avg. Throughput (KB/s)	112.19	78.10	90.61	70.89	54.29	70.64
St. Dev.	21.55	4.42	26.95	10.77	26.15	9.00

b)

Message Size (Bytes)	256012					
Reliability	95%		90%		85%	
Connector	TCP	Mockets + NetProxy	TCP	Mockets + NetProxy	TCP	Mockets + NetProxy
Avg. Throughput (KB/s)	114.68	79.39	73.76	72.41	33.42	71.19
St. Dev.	9.40	6.88	23.97	10.28	21.59	10.37

c)

Message Size (Bytes)	524300					
Reliability	95%		90%		85%	
Connector	TCP	Mockets + NetProxy	TCP	Mockets + NetProxy	TCP	Mockets + NetProxy
Avg. Throughput (KB/s)	113.45	81.21	73.82	76.09	23.20	74.69
St. Dev.	5.56	10.45	12.88	9.88	17.45	11.66

From these results, it appears clearly that there is still room for improvements for NetProxy and Mockets, for which the additional overhead of the proxy-based solution and the shortcomings of Mockets' congestion control emerge when operating with reliable links. However, my experience with the use of NetProxy and 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. The reason lays in the relatively high number of packet retransmissions occurring during the experiments, which corresponds to an average PER level higher than 10 percent. Moreover, as the experiment presented in the next Chapter will show, NetProxy can significantly reduce the amount of traffic flowing from a network to another and give it a much smoother shape, which entails a better utilization of the available bandwidth.

4.3.4.3. Tactical Environment Experiment

In my third experiment, I evaluated the impact of ACM NetProxy in a reference scenario based on the 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 occurring in TENs, 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.

During the experiment, several military hub vehicles could connect to the Operations Center (OC) using SATCOM communications links, which have a bandwidth of 256Kb/s and an average latency of 2 seconds. The purpose of each hub vehicle was 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 the ACM NetProxy was running in GM on a node in the OC network, from 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 going over the SATCOM links to destination.

I configured all NetProxy instances to remap outgoing UDP and TCP transmissions over a reliable Mockets connection opened on the SATCOM link. I then enabled the QoS

enhancement options to compress data on all streams using the Zlib library and consolidate all UDP messages addressed to the same destination. Note that NetProxy always performs traffic buffering 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 tried 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 handhelds devices in the MANET). However, all NetProxy instances were configured to perform the same operations on data, whose type and magnitude were comparable across the different teams deployed in the scenario. Therefore, I believe that the narrower focus of this analysis does not affect the validity and the purpose of this discussion.

For my analysis, I divided the whole experiment duration in time intervals of 0.1 seconds and allocated each network event in its corresponding slot. Table 3 below presents a statistical summary of collected measurements. It compares the generated traffic (in bytes) and the number of packets sent before and after NetProxy processed the traffic. Reported statistics include arithmetic mean, standard deviation, and maximum number of bytes and packets sent over the network in a single interval. All traffic labelled as “before” only considers packets addressed to nodes in the EN, so that it is possible to evaluate the impact of NetProxy fairly.

From the results reported in Table 3 it is clear that, after going through NetProxy, the amount of traffic coming out from the ENI is substantially less than the one generated by nodes in the IN. Looking at the mean, the effects of data compression and packets consolidation are evident, with a reduction of 30.6 percent in the average number of generated bytes and 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 Fig. 22 below illustrates better.

Reducing burstiness is essential to enable NetProxy to provision the required QoS. First of all, it avoids many packets being lost at the bottleneck 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.

Fig. 21 shows in more detail the effects of the data compression and packet consolidation features of the ACM NetProxy. The two 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. Fig. 21a represents, with blue bars, the traffic (in KB) flowing in the IN, and with a red color the traffic sent out on the EN by NetProxy. Similarly, Fig. 21b 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. This in turn 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 [116].

Table 3 Mean, standard deviation, and maximum value of traffic (measured in bytes) and packets sent over the network in each 0.1s interval before and after NetProxy's processing

	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

Fig. 21a Difference in traffic (expressed in KB/s) sent between IN and EN before and after NetProxy processed the traffic

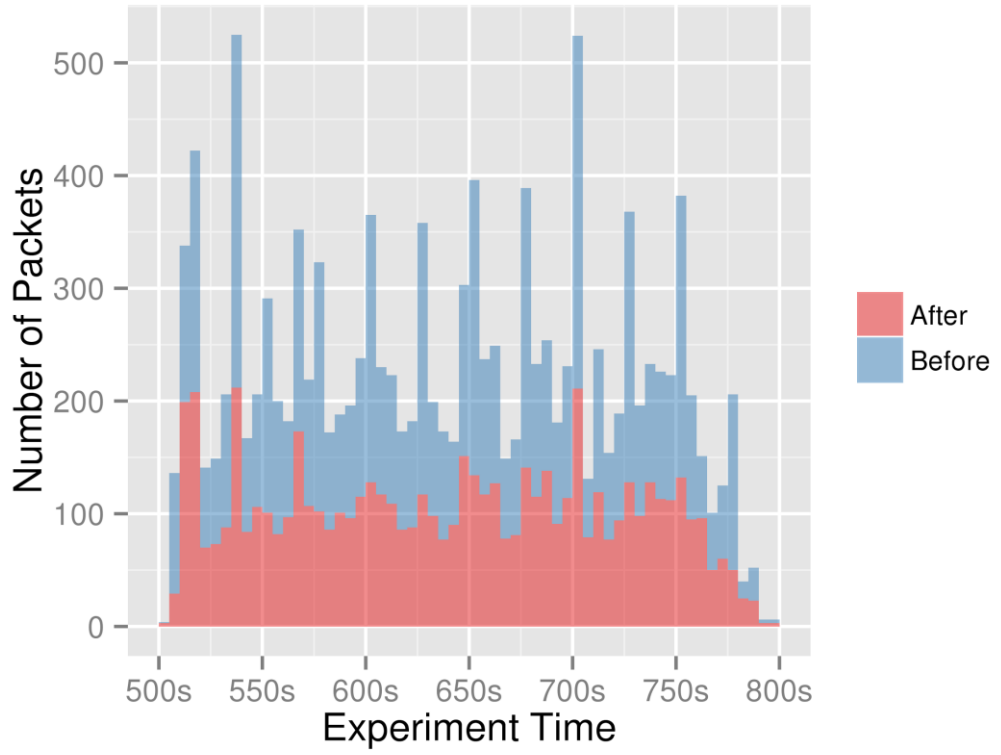

Fig. 21b Difference in the number of packets sent between IN and EN before and after NetProxy processed the traffic

Fig. 22 depicts the empirical density distribution of the number of bytes and packets, respectively, sent over the IN and EN in each of the 0.1s long intervals in which I partitioned the experiment for this analysis. The two 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 performance that is more predictable. The data reported on the X axis of the two graphs was limited to 20 KB and 20 packets, respectively, to better show the differences in shape between the two density distributions. In Fig. 22a, 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, Fig. 22b 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, note that the tail of the curves marked as “Before” would reach almost 200 KB in Fig. 22a, and go beyond 300 packets in Fig. 22b. I chose not to include all data in the graphs to obtain intelligible figures.

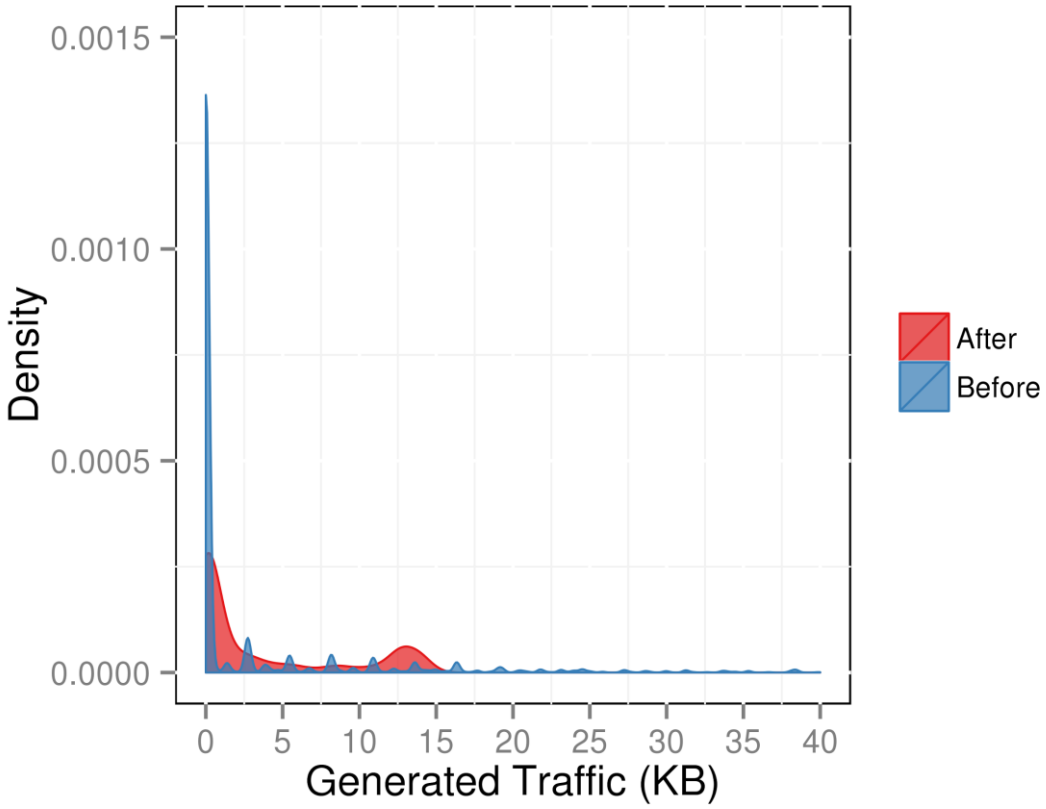

Fig. 22a Density distribution of the number of bytes (plot limited to 40 KB) before and after NetProxy processed the traffic

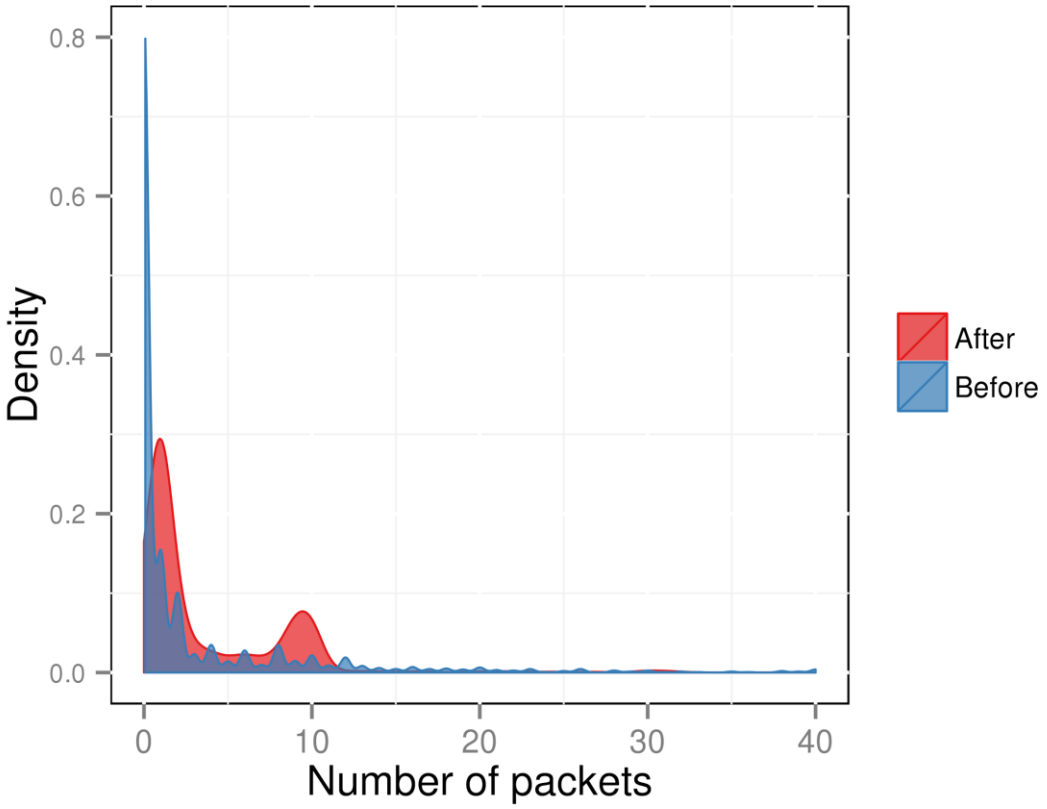

Fig. 22b Density distribution of the number of packets (plot limited to 40) before and after NetProxy processed the traffic

5. SMART DISCOVERY AND EXPLOITATION OF AVAILABLE RESOURCES

Smart cities and other NGN scenarios, characterized by extreme dynamism of nodes and heterogeneity, will suffer a scarcity of resources mainly due to the congestion and costs associated to the use of the cellular network and the inability of mobile nodes to access the wired network infrastructure. Thus, nodes will have to rely increasingly on alternative connectivity solutions that can naturally take advantage of node mobility and heterogeneous network interfaces.

Ad hoc networking enables nodes to establish connections between each other directly, without the support from any network infrastructure, and so it seems a very interesting approach to provide connectivity between nodes in NGN scenarios. However, in MANETs, connections between nodes are mostly unstable, partially because of mobility and partially because of the higher packet loss rate over links. This represents a challenge for applications that need to access the network to satisfy their requirements, which differ based on the class of service they implement. For instance, emergency and danger alert applications need to minimize latency and maximize dissemination of the alerts, most likely with a priority that depends on nodes location and users' social relations. Differently, traffic-monitoring services require higher bandwidth, but they usually do not have severe requirements in terms of latency and the capability of spreading messages across the network (nodes' reachability). Therefore, depending on their service class, next-generation applications will need to be provided with enough bandwidth and solutions that enable the quick and effective discovery and exploitation of new connectivity resources.

This and the next chapter presents two different, but complementary, approaches to provide smart resource management in smart cities and other NGN scenarios. The first solution combines features from the research in the fields of Opportunistic and Information-centric Networking and proposes ICeDiM, a middleware for NGN scenarios that takes advantage of heterogeneous connectivity solutions and in-network caching to improve network performance. ICeDiM provides high message delivery ratios when compared to other solutions from the research literature on Opportunistic Networking, while it can keep overhead and latency low in the network. Moreover, part of my research work on this topic led to the definition of the concept of Application-level Dissemination Channels (ADC), which addresses the issue of managing nodes' resource sharing in next-generation networks and tries to engage users in sharing their devices' resources.

The next Chapter presents an extension of DisService that leverages Opportunistic Networking techniques to predict contacts with resource-rich nodes and base decisions regarding packet delivery on such predictions. In fact, by analyzing the history of past contacts with other nodes, my work shows that it is possible to *discover a wide spectrum of complex periodic patterns* and build predictions on future contacts based on the recurrence of those patterns. This allows the middleware to choose whether to send data immediately using the cellular network, or wait for the arrival of a node according to a prediction and hand packets over to it via an ad hoc link (for instance, using Bluetooth or Wi-Fi), in order to promote the offloading of the cellular network.

5.1. ICeDiM: a Communications Middleware for Next-generation Networking Scenarios

The challenges that extremely dynamic and heterogeneous networking scenarios, such as those that will arise in the future with next-generation networks, will hinder the correct functioning of applications. In fact, direct communication solutions and schemes, based on the setup of end-to-end channels as in the client-server paradigm, will have a hard time to try to keep connections open to specific destinations when facing high nodes mobility and packet loss. As a result, applications in NGN scenarios might not be able to offer the desired QoE to their users.

Instead, applications and services running in the NGN scenario would be very well served by communication solutions based on the *Information Centric Networking* (ICN). By relying heavily on the publish-subscribe model and shifting the focus of communications from the content's location to the content itself (IOs) [17], ICN seems a very promising solution to pursue in order to tackle the problems introduced by high nodes mobility and frequent disruption of end-to-end communication paths. In addition, the ICN paradigm is particularly well suited to support communications in heterogeneous networking environments and it naturally takes advantages of information locality. Finally, in-network caching allows ICN to support delay-tolerant communications, reduces the impact of network partitioning on information availability, and enables subscribers to retrieve content from multiple sources [79] [80].

Unfortunately, while ICN seems to fit very well into the operating context of next-generation communications scenarios, it was essentially devised for the wired Internet, to provide an architecture that can respond efficiently to the users' requests by decoupling IOs from their originators and by bringing the trendiest contents closer to the users through in-network

caching on powerful router nodes [87] [117]. In fact, several studies suggest that ICN-based approaches could work very well within reference scenarios that focus on static and social content exchanged over the wired Internet [83] [118]. However, the methodologies and tools adopted by the most common ICN implementations are not very well suited for wireless communication environments and require substantial modifications to perform effectively in next-generation environments.

Additionally, the smart exploitation of the usually scarce and heterogeneous communication resources has not yet been explored extensively by the research literature on ICN. One of the targets of the work presented in this Chapter is indeed to reduce this gap, by investigating the feasibility of the ICN paradigm in the NGN scenario and analyzing advantages and difficulties that arise when developing an ICN-based information dissemination middleware.

To this end, this Chapter presents ICeDiM, a communications middleware that takes advantage of concepts developed in the research fields of both Opportunistic Networking and ICN. I designed ICeDiM building on top of two fundamental concepts to support ICN-based communications in next-generation environments: *Application-level Dissemination Channels* (ADCs) with *tunable permeability levels* and the capability of *leveraging the broadcast nature of wireless channels* in order to improve the effectiveness of in-network caching and the information dissemination process. The chapter also discusses the results of a thorough and in-depth experimental evaluation of ICeDiM in a next-generation environment realistically simulated using ICeONE. ICeONE is an extension of the well-known ONE simulator (http://www.netlab.tkk.fi/tutkimus/dtn/theone/), which represents the state of the art of Opportunistic Networking simulation, that I designed and implemented to enable the simulation of ICN-based communication solutions in next-generation environments. The results obtained show how the concepts implemented in ICeDiM can keep resource consumption on the network nodes under control while also increasing the delivery ratio of IOs compared to other solutions developed in the context of research on Opportunistic Networking.

5.1.1. Taking ICN to the Next-generation Scenario

Researchers have dedicated a significant attention to ICN recently. Part of their work, such as that on naming, is directly applicable to NGN scenarios. For this reason, the research efforts conducted in the context of my work on ICeDiM did not focus on those topics.

Instead, routing of IOs becomes one of the most relevant topics when moving ICN from a wired Internet environment to the NGN scenario. The research work on routing in ICN can be further decomposed in two subcategories: routing of interest messages and forwarding of data back to the subscriber. While researchers have devised several routing algorithms for ICN in the wired Internet, those solutions highly depend on a backbone of router nodes equipped with large, fast storage modules and dedicated processing power to support caching and scalability [87] [94] [117]. However, such assumptions cannot hold in next-generation communication environments, where network heterogeneity and the major presence of portable devices and sensors cannot provide the necessary memory and computational resources.

Still, the ICN paradigm seems to adapt particularly well to the communications patterns that will arise in NGN scenarios, as discussed in Section 2.2. In fact, ICN allows users to be content producers by publishing some content, and content consumers by notifying their interest in certain IOs and requesting their delivery. This way, ICN supports many-to-many communications by allowing users to subscribe to multiple types of content and retrieve IOs from multiple sources at the same time [119] [120]. In addition, the concept of subscriptions in ICN can model well that of a set of recipients that hold a certain relation to a user, such as the group of his/her friends, family members, or coworkers, or with a common interest or characteristic, such as the set of people who live in the same neighborhood, who take the same train to commute to work every day, who like the same TV show, or who are interested in news about science and technology [121]. This property could be a great advantage for next-generation mobile applications implementing social- and location- based services.

Assuming a wired, mostly static network infrastructure at the basis, most ICN implementations use a pull-based content delivery model, distinguish only between on-path and off-path caching [83], and implement simple cache eviction policies, such as Least Recently Used (LRU) or Least Frequently Used (LFU) [92] [122], or probabilistic IO replacement policies [91]. However, in dynamic scenarios where nodes move frequently and fast, and where they are potentially interested in any IOs that carry information on certain themes, the pull model typically implemented by ICN solutions needlessly increases delivery latency, thereby reducing the length of connectivity windows between nodes and harming the efficacy of IO dissemination [77] [123].

Instead, studies and experiments proved push-based models to be more effective under very dynamic conditions [36] [124] or when one-to-many and many-to-many communication schemes are involved [80]. Thus, the system would benefit from a model where interest messages do not need to be routed towards the closest content provider available to issue

a request for a specific IO, but instead nodes use them to notify their neighbors about the IOs on which they are interested. After that, neighboring nodes are left to decide for the forwarding of each single IO they have in their cache. This shifts the push model in the direction of a more hybrid one, according to which nodes forward cached IOs towards their neighbors based on the interest messages they received from them. In addition, IO caching techniques could also leverage the knowledge on nodes' interests to improve their effectiveness, for instance by giving higher priority to IOs that nearby nodes will less likely cache.

Finally, It is important to note that almost the totality of the research done in the field typically assumes that nodes in the network are willing to share (a part of) their resources to deliver other nodes' messages [125]. This assumption is motivated partially by the need of simplifying the system model and partially by the consideration that, in many specific contexts, nodes are deployed and configured by the same party, hence the possibility of uncooperative behaviors can be ruled out. However, this last case does not apply to next-generation scenarios, where nodes include private smart devices and vehicles. Nonetheless, nodes in the network need to cooperate and invest significant amounts of resources to ensure data delivery, regardless of the type of routing solution used (context-aware routing schemes need to build up a thorough knowledge of the network in order to enable smart routing decisions, while context-oblivious schemes aim at flooding the network with multiple copies of the same message to increase the probability of one copy reaching its destination). Therefore, it is essential to devise a compelling resource management system that can give applications the tools to control resource sharing while, at the same time, preventing that malicious applications can exploit other cooperating nodes without contributing with part of their node's resources in the routing effort.

In light of these considerations, I propose the Information-Centric Dissemination Middleware, or ICeDiM, to address the need for an effective ICN-based solution in next-generation communications scenarios. ICeDiM offers overlying applications the necessary set of services to enable the delivery of IOs in ICN-based heterogeneous networks. The middleware combines achievements from the research on Opportunistic and Information-centric Networking (the Thesis gives a detailed review of both research areas in Section 2.3) into a solution that supports the communication process of applications in NGN scenarios. Thus, ICeDiM exploits the store-carry-forward principle, typical of Opportunistic Networking, to provide delay-tolerant communications to overlying applications. In addition, it takes advantage of in-network caching and the capability of retrieving IOs from any carrier, two concepts coming from the research on ICN, to improve the dissemination process, reduce latency, and limit the effects of network partitioning on information availability.

Furthermore, if any NICs on the node support broadcast transmissions, ICeDiM can also overhear broadcast packets sent by other nodes to improve caching opportunities and further tune the dissemination process.

The middleware also addresses the problems of resource sharing in NGN scenarios, which include engaging people in sharing their devices' resources and the management of shared resources, by introducing the concept of Application-level Dissemination Channels (ADCs) as a unit of resource allocation and communication. ADC-based IO dissemination relies on the broadcast nature of radio communications and extends naturally to heterogeneous network technologies and environments. All these functionalities contribute to the composition of a system that enables the distributed smart management of available resources for communication purposes.

5.1.2. Application-level Dissemination Channels

Application-level Dissemination Channels are the most innovative building block for communications in ICeDiM: they represent both an application-level thematic attribute and a node-level commitment to sharing resources within that channel. Thus, ADCs regulate resource consumption and nodes' collaboration in the effort of delivering messages in next-generation networks. Examples of ADC themes include textual news, video stream news, social networking, wiki, smart city access, information from nearby sensors, and so forth.

More specifically, the thematic nature of ADCs allows to group similar applications and services to achieve three objectives: reducing resource usage at the network level, overcoming nodes' and users' reluctance in sharing their resources, and exploiting potential optimization opportunities. Since only nodes interested in a given theme will participate to the corresponding ADC, resource consumption for IOs that do not belong to any theme in which a node is interested, e.g., IO retransmissions and storage and processing of IOs and metadata, is reduced significantly. In addition, I expect that applications and users will be more willing to share resources on their devices if they are consumed to sustain the same kind of services they use. Finally, if nodes cache messages produced by applications of the same thematic classes, there is a higher chance that future requests can be resolved directly from the cache.

Applications can join one or more ADCs dynamically, either by publishing IOs within that (those) channel(s), or by explicitly subscribing to IOs published within that (those) channel(s). When an application joins some ADC, ICeDiM ensures to trade part of the node's network bandwidth, storage capacity, and computational power in exchange of the

support in the message delivery process from other nodes that belong to the same channel. This way, all nodes belonging to an ADC will take part in the common, shared, and distributed effort of disseminating and delivering any IO published within that channel.

ADCs are not restricted to a single application, so various applications at the same time can join the same ADC. When this happens, applications that are member of the same ADC will share resources. As a result, if the applications running on a node subscribe their interest to multiple themes, that node would participate in the dissemination process associated to all corresponding ADCs at the same time. Just as applications and nodes can join multiple ADCs, the same way IOs can belong to multiple ADCs. Unsubscribing from the messages published within some ADC is equivalent to leaving that channel.

In the highly dynamic next-generation environment, ADCs will have an ever-changing set of resources available, corresponding to the overlay network formed by the nodes subscribed to the channel (and the links that are created and disrupted as those nodes fall in and out of each other's communication range) and all the resources shared by those nodes. Within the resource constraints introduced by the ADC, the dissemination strategy remains in charge of making decisions on message routing and forwarding, as discussed in Section 5.1.1.

Finally, the thematic nature of ADCs allows application designers to explore trade-offs between the effectiveness of communications within the subscribed ADCs and the preservation of nodes' resources, by modulating the granularity of the dissemination channels' themes – and hence the corresponding number of participating entities. Building applications on top of a high number of ADCs with finer granularity themes would lead to a lower resource consumption on the nodes, but also to a lower collaboration between them and, consequently, diminished support from the network. In contrast, applications using a low number of ADCs with coarser granularity themes would limit the constraints enforced on resource consumption, to the point where all IO forwarding and caching decisions would be left entirely to the dissemination strategy (when a single ADC is used for all applications).

To illustrate how ADCs work, Fig. 23 gives a visual representation of the overlay networks generated by three different ADCs in a mobile scenario, represented with three different colors. The channels' names are "Green", "Yellow", and "Violet", respectively, from the colors used to distinguish them in the figure. Nodes A, B, and C are subscribed to ADC "Green", nodes D and E are subscribed to ADC "Yellow", and nodes G, H, I, and J are subscribed to ADC "Violet". However, node A has also joined ADC "Violet" and, similarly, nodes E and F have also joined ADC "Green". In this example, node K has not subscribed

to any channel. This might happen for several reasons, such as in cases where no applications are running on one node, or under request of the user, for instance in order to save the remaining battery life on his smartphone.

Black, continuous lines in Fig. 23 represent links between nodes that belong to the same ADC. ICeDiM can thereby exploit such links to transfer IOs within the context of that ADC. Transfers will take place as long as both the destination and the IO belong to the same ADC and the decision-making component within ICeDiM decides to begin transmission, in accordance with the underlying dissemination strategy. We call this type of transmission “within-channel transmission”. Instead, red dotted lines are used to represent links between nodes that do not belong to the same ADC. Therefore, even though some connections are available at the link layer, ICeDiM ignores them for the purpose of IO forwarding. This way, for example, communications between nodes A and F are possible only by means of the store-carry-forward paradigm, i.e. by caching IOs and waiting for the two nodes to move under transmission range.

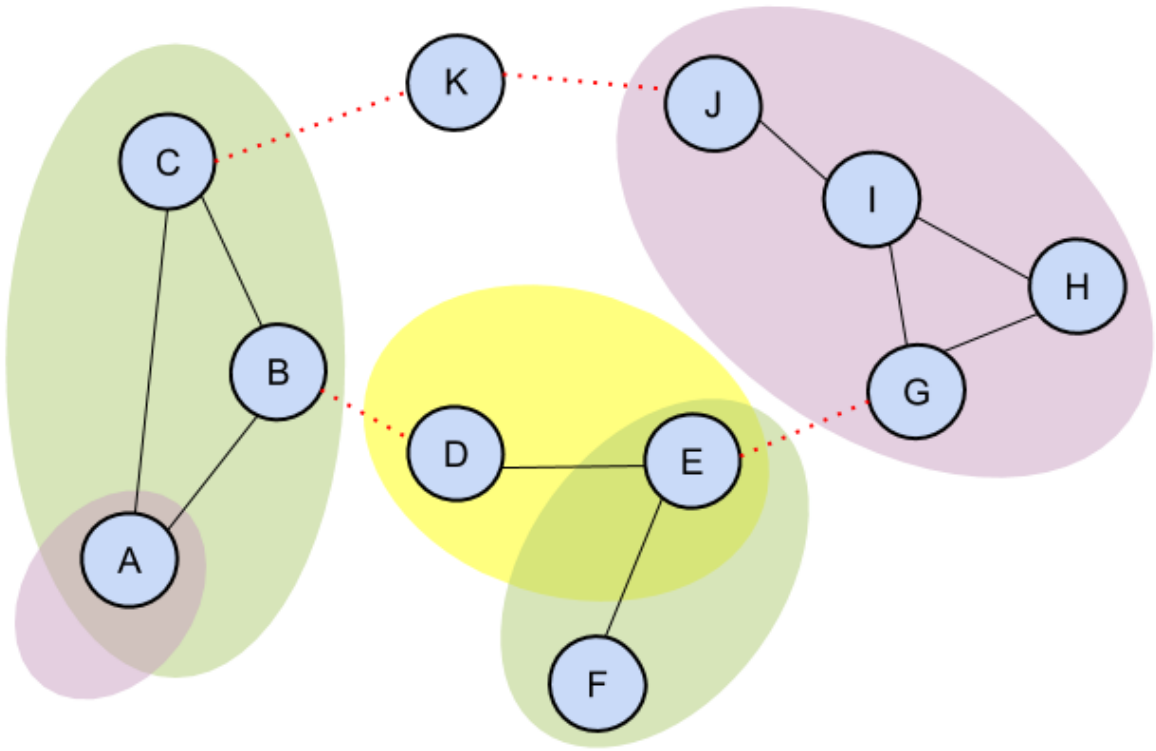

Fig. 23 ICeDiM nodes communicating over Strict Application-level Dissemination Channels

5.1.2.1. Permeability of Application-level Dissemination Channels

Many different types of nodes, owned by various parties (public organizations, private companies, people, etc.), with different and always changing goals and constraints, will operate in NGN environments. For instance, the type of network activity performed by personal devices such as smartphones, tablets, and laptops highly depends on the user's behavior, as well as the screen size and resolution of their devices, remaining battery life, computation power and memory available, and so on.

In these situations, assuming that all nodes will always cooperate to the IO dissemination process would likely be too optimistic. As a result, strictly enforcing the dissemination of IOs within the corresponding ADC boundaries might lead to missed connectivity opportunities, prevent message delivery to interested parties, and cause partitioning phenomena in the overlay networks associated to the ADCs, even if a communication path at the link/network level connects those partitions. A situation like the one described is illustrated in Fig. 23, where the overlay network established by the ADC "Green" is partitioned into two parts: one composed of nodes A, B, and C, and the other composed of nodes E and F.

Borrowing from the biological metaphor of "*membrane passing*" diffusion processes, ICeDiM addresses these issues by introducing the idea of dissemination channels with *tunable permeability*. More specifically, ICeDiM enables the partial relaxation of the constraints on the dissemination of IOs imposed by ADCs, thus allowing a (configurable) portion of IOs to pass through the boundaries of an ADC in order to favor their dissemination.

The goal of SP-ADCs is to maintain nodes' resource consumption and their participation in the IO dissemination process under control while, at the same time, increasing the options for communication. The level of permeability is a fundamental and system-wide attribute of an ADC, which must be selected at the moment of ADC creation and must be respected by all nodes participating to that ADC. ICeDiM supports three ADC permeability levels: *strict*, *semipermeable*, and *unconstrained*.

Strict ADC (S-ADCs) have impassable boundaries, as described in Section 5.1. As a result, within an S-ADC, ICeDiM will only cache IOs that belong to corresponding thematic channel. Also, a node can disseminate an IO "M" over an S-ADC "C" to a second node only if the latter node has subscribed to "C", i.e., S-ADCs only allow within-channel IO transmissions.

Instead, *Semipermeable ADCs* (SP-ADCs) permit a fraction of IOs to seep through the boundaries of their own channels and, thus, reach portions of the network that would be inaccessible using S-ADCs. In addition to within-channel transfers of IOs, semipermeable channels implement a “membrane-passing” process that comes into play to determine if an IO can be transferred anyway. Thereby, differently from what happens with S-ADCs, ICeDiM nodes might store in their cache IOs received over an SP-ADC that do not belong to any of the ADCs they joined. In any case, within-channel transmissions have higher priority than those generated by membrane-passing events, i.e., ICeDiM favors the transfer of IOs that can be delivered to nodes in the same ADC over transfer of IOs that do not belong to any of the ADC the neighbors have joined.

Finally, in *Unconstrained ADC* (U-ADC) no boundaries whatsoever are enforced on the dissemination of IOs. This represents a degenerate case, in which all nodes in the network have joined the same, single dissemination channel to which all IOs belong, independently from their corresponding thematic attribute. The U-ADC mode does not enforce any constraint on resource allocation and consumption in the nodes, and IO forwarding and caching decisions depend entirely on the dissemination strategy applied.

Fig. 24 shows the same scenario of Fig. 23 but, in this case, ADCs are semipermeable instead of strict. The main difference is highlighted by the use of black dashed lines to represent links between nodes that do not belong to the same dissemination channel, instead of red dotted lines, to show that IO transmission across different ADCs is permitted when the SP-ADC mode is used. Let us consider, for example, node A in Fig. 24. That node hosts an application that has published an IO within the context of the ADC “Green” (depicted as a green, unopened letter in the figure). ICeDiM transfers that IO to node B, since nodes A and B both belong to the “Green” ADC. At this point, node B can try to transfer that IO to node D, which might decide to cache it, even if not interested in “Green IOs”. If this happens, the IO has effectively passed through the channel’s membrane to be received and stored by node D. At this point, node D knows that one of its neighbors, namely node E, has subscribed to the ADC “Green”, and so it will start delivery of the IO to E. This operation is not subject to any probabilistic decision, as node E is a potential destination of “Green IOs”. After reception of the IO is completed, node E can proceed and transfer it to node F by means of a within-channel transmission. Note that the transmission between nodes D and E is within-channel because both the IO and node E belong to the ADC “Green”, and not because nodes D and E both joined the ADC “Yellow”.

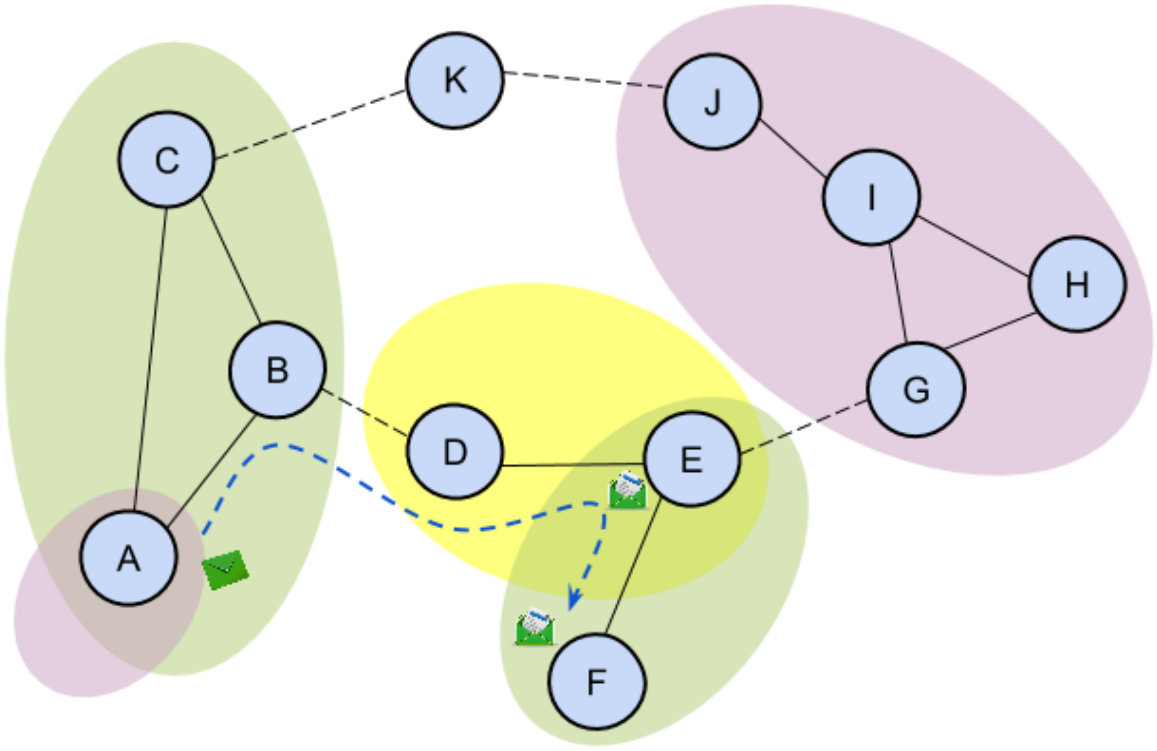

Fig. 24 ICeDiM nodes communicating with Semipermeable Application-level Dissemination Channels

5.1.2.2. Controlling the Permeability of Application-level Dissemination Channels

ICeDiM uses a probabilistic model to control the “membrane-passing phenomenon”, i.e., the process of one or more IOs seeping through the boundaries of an SP-ADC and, potentially, being received and cached by another nearby node that did not join the same channel. The model is based on two probability parameters: FP (which stands for Forwarding Probability) and CP (for Caching Probability).

The FP and CP parameters respectively control the forwarding and storing, or caching, of IOs, in an independent fashion. Thus, whenever a node has the possibility to forward an IO to a neighbor that has not joined the same ADC of that IO, transmission will be performed with probability FP. Similarly, upon reception of a new IO that does not belong to any of the ADCs for which a node has made a subscription, that node will cache the received IO with probability CP. Whenever an IO is generated or cached for the first time by a node, ICeDiM runs the forwarding decision-making procedure for that IO before transmitting it. With reference to our biology-inspired metaphor, this mechanism is similar to having membranes

that permit copies of an IO to pass through them with a permeability coefficient FP, and then to enter a cell, i.e., a node, with a permeability coefficient CP.

ICeDiM keeps track of all the forwarding and caching decisions made by a node concerning a certain IO, and re-evaluates them only when network conditions change in a way that could affect the dissemination process. In fact, naïvely rerunning the probabilistic forwarding or caching decision-making procedure every time ICeDiM needs to decide whether to forward or cache an IO would severely distort the impact of FP and CP, effectively increasing their value of a factor that depends from the number of contact opportunities, the duration of those contacts, and the dissemination strategy employed.

To understand this problem, let us assume that two nodes, X and Y, ran a naïve implementation of the SP-ADC mode and that node X had an IO M in its cache that does not belong to any of the ADCs joined by node Y. Let us now consider the case of node Y entering the transmission range of node X and, consequently, of node X choosing to forward M to node Y, following a successful outcome of the stochastic IO forwarding decision-making procedure controlled by FP. Let us also assume that node Y decided to drop the IO, following a negative outcome of the stochastic IO caching decision making process controlled by CP. Suppose that the two nodes then proceeded towards their destinations, lost their connection, and, after a while, node Y entered the transmission range of a third node, Z, which also happens to have M in its cache. Similarly to what node X did before and following a successful outcome of the stochastic IO forwarding decision-making procedure, node Z could forward M to Y, which in turn this time might decide to cache the message, following a re-run of the stochastic IO caching decision-making process that returned a successful outcome. The result is that message M was forwarded twice before node Y cached it, relaxing the control of caching decision, doubling bandwidth consumption, and increasing latency. In real-life scenarios, with a much larger numbers of nodes and messages involved, this might lead to very significant performance losses.

To avoid this issue, we adopted a more sophisticated decision-making process that reevaluates past decisions with respect to the membrane-passing phenomenon only when appropriate. More specifically, in ICeDiM all forwarding-related decisions concerning an IO M (that is, whether to transmit it or not) made by a node X are kept valid until a new neighbor Y that has not subscribed to any of M's ADCs enters the transmission range of node X, whereas nodes leaving its transmission range are ignored. This way, a node will decide on IO transmission only when the conditions of the surrounding environment change in a meaningful way. Instead, the decision of caching or discarding a specific IO is made only once, the first time that IO is received, and the node complies with it indefinitely. Thorough

experimental tests confirmed that the enhanced versions of the IO forwarding and caching decision-making processes prevent situations like the one described in the example above from happening.

5.1.3. Dissemination Strategies for Wireless Communications

Wireless technologies such as Wi-Fi allow nodes to broadcast messages to all neighbors at the same time. In terms of bandwidth usage, the cost of broadcasting a packet is (almost) the same as sending the same packet to a single node via unicast, but broadcast can reach multiple nearby nodes in a single transmission, thereby potentially increasing spectrum efficiency [36]. This fact is widely known in literature and many techniques designed to improve performances in MANETs and other wireless networks rely on it [126] [127].

The broadcast nature of the wireless medium represents a significant potential also for the ICN paradigm. In fact, while nodes on the delivery path of an IO will normally cache and forward the data towards the subscribers, other nearby nodes can exploit the broadcast transmission to know what IO is being delivered and any useful pieces of information that can be extracted from the metadata sent along with the IO. Additionally, broadcast transmissions can ease the delivery of IOs by enabling the opportunistic caching and redistribution of overheard objects to other nodes in the network. One could see this strategy as something in between on-path and off-path caching, and it is particularly effective in those cases where multiple nodes in one network have subscribed to the same content, since it enables the network to take a distributed, proactive effort to deliver the content to all subscribers [118].

This strategy can significantly reduce resource consumption when, for instance, many customers sitting in a coffee bar have connected their tablets and smartphones to the free Wi-Fi to watch a news channel that is streaming breaking news, or a live game of the national football team. In such cases, the AP can broadcast messages to the devices in the ICN-enabled network of the coffee bar, thereby saving resources and allowing nodes within range to cache transmitted messages. Live video streaming applications could implement features like, for example, a reliable, even if lower-quality, video stream, or the generation of a limited amount of retransmission requests to retrieve frames missing from its internal buffer, in order to increase the offered streaming resolution. The combination of the availability of frames cached by nodes of the local network and the content-centric delivery mechanism of ICN allows these and any applications with similar features to retrieve missing data from nearby nodes, consequently reducing server-side load. Moreover, resolving retransmission requests locally ensures more timely data retrieval, which is critical

for services like live video streaming. [128] proposes and analyzes a similar approach to increase efficiency of video streaming over ICN-enabled networks.

Nodes running ICeDiM can take advantage of broadcast-enabled network interfaces by periodically sending HELLO messages to notify neighboring nodes about their presence. HELLO messages contain information about all the ADCs the sender has joined (said otherwise, HELLO messages contain all sender's interests) and all the IDs (each one being a string of variable length) of cached IOs. During forwarding, ICeDiM uses the information contained in HELLO messages received from neighbors to select which IOs to send and passes them to the underlying dissemination strategy. IO selection is done by crossing the ADC joined by neighbors with those of cached IOs. If a message forwarding order prioritization strategy is present, ICeDiM applies it to the list of IOs available for forwarding before feeding it to the dissemination strategy.

Normally, ICeDiM caches all messages it receives that belong to one of the channels the node has joined or that were probabilistically accepted if running in the SP-ADC mode. In addition, ICeDiM can take advantage of connectivity options and features that are only available on a subset of the interfaces. For instance, if a node is equipped with a network interface that supports packet broadcast, ICeDiM applies an opportunistic caching strategy that permits to cache overheard IOs broadcasted by neighbors, provided that they belong to one of the ADCs joined by the overhearing node or that they managed to traverse its semipermeable channels.

ICeDiM includes two broadcast-capable dissemination protocols: the Epidemic Broadcast Routing Protocol (EBRP) and the Spray and Wait Broadcast Routing Protocol (SnWBRP). These strategies implement the broadcast versions of the Epidemic and the Spray and Wait routing protocols, as documented in [65] and in [66]. The versions of EBRP and SnWBRP implemented in ICeDiM support ADCs by subscribing to one or more channels and generating/forwarding messages in the context of one or more of them, and can be used in combination with any one of the three ADC modes.

The original versions of the Epidemic and Spray and Wait routing protocols have an anti-entropy phase to exchange information about what messages each node has in its cache, so that nodes can learn which ones should be transmitted and to which nodes. I extended the anti-entropy phase of both the EBRP and SnWBRP to support ADCs and then integrated it in the ICeDiM HELLO message. This way, nodes are capable of notifying their neighbors about the decision of not caching certain messages, as it might happen when

S-ADCs or SP-ADCs are used. This avoids performing pointless transmissions, thereby saving bandwidth and enabling a better utilization of the communication opportunities.

Another very interesting technology that can further increase efficiency in next-generation networks is Device-to-Device (D2D). Some of the D2D techniques that have been proposed exploit the cellular spectrum to establish direct communications between two mobile devices without the support from base stations or the core of the cellular network. The advantages of this technology include higher throughput, better energy efficiency, lower communications delay, and increased fairness [4]. Several types of services could benefit greatly from it, e.g., video communication, cellular network offloading, supporting communications during disaster recovery, proximity-based services, etc. D2D is currently a very active research topic, but many problems still need a resolution and many design choices are yet to be made before devices will support this technology.

Nonetheless, ICN-based communications middleware such as ICeDiM would benefit greatly from D2D techniques. In fact, the availability of an additional network interface to perform local communications would broaden nodes' communication opportunities, with positive consequences on the quality of routing and forwarding decisions. More specifically, D2D techniques would enrich the heterogeneity of connectivity technologies available to ICeDiM with one that provides one-to-one node communications characterized by low latency, low energy consumption, and high throughput. [129] shows that the combination of D2D with other wireless short-range technologies, like Wi-Fi, can be extremely effective to reduce power consumption and service latency in multicast-based applications. The article also describes how nodes could use D2D techniques to support content sharing between neighbors and extend the network coverage. These features are key blocks of ICN-based communications in next-generation environments, and so I believe that D2D will become an enabling technology for ICN in those scenarios.

5.1.4. An improved Version of the ONE Simulator

Given the difficulty of running large scale experiments in a real scenario due to the great number of devices and people that would involve, I opted to use simulation and reproduce the conditions of NGN environments. The importance of simulation in the area of wireless and heterogeneous networks is widely recognized in the literature [74] [130] [131] [132]. I evaluated several different options to choose the simulator that better satisfy the requirements for testing ICeDiM in a realistic and accurate networking environment.

5.1.4.1. Choice of the Simulation Engine

NS-2 (http://nsnam.isi.edu/nsnam/index.php) and NS-3 (http://www.nsnam.org/) provide very accurate and comprehensive simulation environments that supports many standard protocols and network interfaces. However, neither NS-2 nor NS-3 provide direct support for realistic mobility models, but require other tools to generate traces from such models to feed the simulator. Similarly, GTNets (http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/) provide an accurate network simulation environment that aims to reproduce the structure of actual networks, but the support for mobility models is poor. Furthermore, NS-2/NS-3 and GTNets are not designed for wireless ad hoc networks or the information dissemination: this makes the set-up of a smart city scenario and the implementation of the required dissemination algorithms extremely time-consuming and error-prone processes.

Unlike the previous solutions, TraNS (http://lca.epfl.ch/projects/trans/) is a GUI tool that integrates real vehicle mobility traces coming from SUMO (a road traffic simulation package, http://sumo.dlr.de) with the network simulator NS-2, to allow the realistic simulation of Vehicular Ad hoc Networks (VANETs). Similarly, VanetMobiSim (http://vanet.eurecom.fr/) can generate movement traces of vehicles in different formats, thus supporting the integration with different network simulation/emulation tools. Nonetheless, both TraNS and VanetMobiSim focus exclusively on the simulation of VANETs, while smart city scenarios require a more diverse and complex combination of nodes' mobility models, as people and infrastructure are also valuable resources. In addition, besides the support for mobility models that can capture the heterogeneity of nodes in a smart city, the accurate analysis of the features of ICeDiM require a proper networking simulation engine.

After having considered the alternatives listed above, I opted for a modified version of the ONE Simulator (http://www.netlab.tkk.fi/tutkimus/dtn/theone/) [133]. ONE is open source, written in Java, and it is designed to simulate wireless opportunistic networks. These characteristics mainly drove my choice, as I was able to extend ONE with the missing features needed to implement ICeDiM and set up the experiments fairly easily. Another value of ONE is that it comes with a set of Router classes that implement a number of routing algorithms for MANETs: Epidemic, Spray and Wait, PRoPHET, MaxProp, and others. Moreover, it supports several types of nodes (pedestrians, cars, trams, buses, etc.) and mobility models. ONE allows both the pseudo-random generation of events, such as the generation of new messages, and the parsing of files containing event traces. In addition, setting up realistic smart city scenarios, using real maps such as that of the city of Helsinki, included in the simulator's bundle, is quite an easy task to do with ONE.

Notwithstanding the many useful characteristics of ONE, the accurate simulation of Application-level Dissemination Channels and broadcast wireless transmissions, as used in ICeDiM, require a few, yet very important, extensions. I built upon version 1.4.1 of ONE to introduce all the necessary missing features. I will hereinafter refer to the extended version of the ONE simulator as the ICeONE (Information-Centric ONE) simulator, or simply ICeONE. ICeONE is available for download at the link http://endif.unife.it/dsg/iceone.

5.1.4.2. Additional Features of ICeONE

Arguably, the most important features I added are the support for communications based on the Publish-Subscribe paradigm and for Application-level Dissemination Channels. In accordance with the Publish-Subscribe paradigm, applications can publish messages in the context of a subscription, and they can subscribe to one or more subscriptions to receive messages published within them. In addition, nodes (and the applications running on them) can generate messages and mark them with a specific channel ID. As a result, marked messages will belong to the relative ADC and will have access to all the resources set aside for such a channel.

Another important improvement is the support for broadcast transmissions, which the original version of the simulator does not natively provide. I implemented broadcast transmissions in the lower level components of the simulator. The motivation is twofold: to avoid polluting higher-level units, which implement the routing and forwarding policies, and to allow the design of other functionalities on top of the transmission mode (unicast or broadcast), but independently from it. These functionalities include an extensible interface for the *management of collisions* between multiple nodes transmitting at the same time and a *collision avoidance mechanism* similar to the CSMA/CA mechanism described in the IEEE 802.11 specifications [116].

To manage packet collisions, the ICeONE Simulator requires any simulated network interface to be associated with an implementation of the Collision Model (CM); this allows the configuration of different interfaces with the CM that better reproduces the characteristics of collisions for that particular technology. ICeONE already comes with two simple CM implementations. Moreover, a Java Interface enables users of ICeONE to develop and use their own CMs. In the experiments, I used the Synchronized Packets Collision Model (SPCM). SPCM permits the correct reception of messages only when the receiver remains within the sender's coverage during the whole transmission. In addition, SPCM marks as collided any message whose transfer overlapped another message reception or transmission, even if only partially. Finally, users could easily extend ICeONE

with more sophisticated CMs that could consider additional parameters when computing the probability of a collision, such as the signal energy or the presence of obstacles in between.

The CSMA/CA algorithm added to ICeONE is based on the algorithm described in the IEEE 802.11 specifications. According to it, the wireless network interface senses the channel whenever the node needs to start a transmission. If the channel is sensed busy, the transmission is backed off for a fixed amount of time after which the channel is sensed again; otherwise, the node sends the entire message. CSMA/CA still leaves open the possibility of incurring the hidden or the exposed node problems [134], which can lead to collisions or reduce network capacity. Note that, while IEEE 802.11 provides for the use of collision avoidance before performing broadcast transmissions, the Request-to-Send/Clear-to-Send (RTS/CTS) mechanism is enabled only for unicast [116] [135].

Another feature that ICeONE adds when compared to the ONE Simulator is an augmented cache management capability to support the IO forwarding and caching mechanisms based on the FP and CP parameters that SP-ADCs implement. The MessageQueueManager class provides an API that enables the refined management of nodes' caching memory through the operations of fetch, store, and delete of single messages. Moreover, two other classes allow tuning the behavior of the MessageQueueManager class by specifying the policies that control message-caching prioritization and forwarding order. The former also defines the cache eviction strategy, by selecting the "least important" messages to be deleted from memory when a new store operation hits the memory limit, whereas the latter affects the dissemination strategy, by suggesting which message to forward next. These classes permit the implementation of message forwarding order prioritization policies in the context of ICeDiM, as discussed in Section 5.1.3.

5.1.5. Experimental Results

This Section presents the sets of experiments I ran to assess the effectiveness of ICeDiM in improving the process of information delivery in NGN scenarios. I used ICeONE to simulate a smart city environment; in fact, I believe that the characteristics of smart cities, in terms of network heterogeneity and nodes' mobility, provide one of the most compelling test cases for the evaluation of ICeDiM. I performed three different experiments, each one aimed at evaluating the impact of different features of ICeDiM.

5.1.5.1. The Smart City Scenario

To evaluate the concepts and ideas presented in this Chapter in the most realistic conditions, I used the scenario provided by the map of downtown Helsinki, included in the original ONE Simulator bundle. Nodes' movements are limited to streets. In the experiments, I differentiate walking nodes (pedestrians, identified by the letters "p" and "w"), car nodes (identified by the letter "c"), and tram nodes (identified by the letter "t"). 80 pedestrians walk along the streets with a random speed, uniformly chosen in the [0.5, 1.5] m/s range. Similarly, 40 cars move with a speed that ranges from 2.7 to 15.3 m/s, while 6 trams have a speed ranging from 7 to 11.1 m/s, for a total of 126 nodes. Pedestrians and cars choose a traveling speed and a random destination point in the map, reach that point through the shortest path, and then stop for a time randomly chosen according to a uniform distribution ranging from 0 to 120 seconds. On the contrary, trams drive predefined routes back and forth, stopping for a time that ranges from 10 to 30 seconds whenever they reach an end of the route, which simulates stops at a bus stop. Fig. 25 shows a screenshot of the nodes' positions at the beginning of the simulation.

Each node is equipped with a short-range, low-speed, and low-power wireless network interface, whose specifications are compatible with those of Bluetooth® 2.0 + EDR; trams also have a wireless interface comparable to IEEE 802.11 Wi-Fi installed on them, which allows for longer-range connections and higher bandwidth. Bluetooth-like interfaces have a coverage of 100 meters and a transfer rate of 2.1 Mbps, whereas Wi-Fi covers a range of 150 meters with a transfer rate of 31.4 Mbps (nominal net throughput of 802.11g with CSMA/CA enabled). Green circles around the nodes in Fig. 25 represent coverage ranges.

All nodes reserve 5 MB of their memory for the purposes of message caching, with the exception of trams. In the first two experiments, tram nodes use 50 MB of memory to cache messages, while I repeated the third experiment multiple times varying the cache size of tram nodes. The values used for the third experiment are 5 MB, 10 MB, 15 MB, 25 MB, and 50 MB.

In the simulations, I considered five different applications and, for simplicity, I assumed that each application belongs to a different ADC and that they subscribe only to that one channel (that is, applications produce and consume messages only in the context of one single ADC). This is equivalent to saying that there is a 1:1 mapping between five different applications and five different ADCs. Each node simultaneously runs a variable number of randomly chosen applications, ranging from one to five, which is set according to a negative exponential random distribution. To maintain constant the number and type of applications running on each node, I fed a pseudorandom number generator with the same seed values

across all simulations. Thus, the tuple [68, 25, 15, 10, 2] describes the number of nodes per number of applications running on each node in all the experiments. This means that 68 nodes run one application chosen randomly among the five available applications, 25 nodes run two randomly chosen applications, and so on. Nodes do not change the set of ADCs to which they subscribe during the simulations. Note that the sum of all elements in the tuple is 120, because the six trams do not take part neither in message generation nor in its consumption. Said otherwise, in my simulations, trams are not eligible sources or destinations for any message, they do not join any ADC, but they might still contribute to message routing, forwarding, and caching according to the ADC mode and the dissemination and caching strategies employed. Thereby, when using the S-ADC mode, nodes will not take advantage of connection opportunities with tram nodes, while with U-ADCs nodes will exchange IOs with tram nodes as if they joined the same set of ADCs. Finally, in case of SP-ADCs, even if tram nodes do not subscribe to any specific channel, nodes can still have access to part of their resources when the membrane-passing phenomenon takes place.

Message creation occurs at each node with a period that varies between 20 and 40 seconds, according to a uniform random distribution. The same type of distribution controls message size, which ranges from 500 KB to 1 MB. I assumed that each IO is contained within a single message and so I will refer to IOs and messages interchangeably in the remainder of this Section. I ran all simulations for 46800 seconds (13 hours) of simulated time. Statistics for events that occur in the first hour were not collected, in order to warm up the simulation environment and to ensure that all caches are primed at the beginning of statistics collection.

I seeded the pseudo-random number generators with the same values in every test to reproduce the same events across all simulations. This ensured that nodes moved towards the same destinations, at the same speed, and in the same order, in each run. Similarly, nodes generated messages of the same size and at the same simulation time in each test.

Fig. 25 A screenshot of the networking scenario at the beginning of the simulation with ICeONE; nodes are in blue and their transmission range in green

5.1.5.2. First Experiment: Parameter Determination

I measured the performance in each test by collecting statistics and computing the network delivery ratio, delivery delay, overhead, the number of dropped messages, and the number of discarded messages. Messages are classified as dropped if they were first stored in one node's caching memory and then deleted in a future occasion, in accordance with some cache eviction policy, whereas I use the term discarded messages in case they were immediately rejected at the moment of reception, as it might happen when S-ADC or SP-ADC are used.

The network delivery ratio d_N is defined as follows:

$$d_N = \frac{D_N}{\sum_n \sum_m s_{n,m}} \qquad (3),$$

where D_N is the number of messages correctly delivered to destination, and the term at the denominator is the sum over all nodes of all messages to which they subscribed during simulation time (excluding the warm-up time). In compliance with this definition, the value

of the single term $s_{i,j}$ is 1 if and only if node *i* has subscribed to the content carried within message *j*, or 0 otherwise. Defined this way, the relation $0 \leq d_N \leq 1$ holds.

Similarly, I define the network overhead o_N as follows:

$$o_N = \frac{T_N - D_N}{D_N + 1} \qquad (4),$$

where T_N is the total number of transmissions performed by all nodes in the network, including collisions, and D_N is the same as above. Thus defined, we have $o_N \geq 0$.

The first experiment aims at identifying the optimal values of FP and CP for the SP-ADC mode in the simulated scenario. Their optimal values depend on several parameters, including the specific network configuration, the number of nodes, their mobility, and their cache size. Determining a network-level optimal pair for FP and CP is very important for at least two reasons. First, it allows fixing those variables to well-determined values and thus comparing SP-ADC against S-ADC and U-ADC. Equally importantly, the process of discovering the best values of FP and CP gives the chance to investigate their impact on the communications performance in the simulated scenario.

In order to identify the optimal parameter values, I fixed the dissemination strategy to SnWBRP and the ADC mode to SP-ADC. Then, I varied the values of FP and CP across all possible combinations of FP (accepting one value among 0.1, 0.3, 0.5, 0.7, and 0.9) and CP (whose value could be one among 0.1, 0.3, 0.5, 0.7, 0.9, and 1.0). Note that assigning the value 0 to both FP and CP is equivalent to setting the S-ADC mode, while assigning the value of 1.0 to both the parameters corresponds to using U-ADCs. Investigating the comparison between the SP-ADC and the other modes is the target of my second experiment. The first experiment required 30 simulation runs to be carried out completely. After data collection was completed, I analyzed the network delivery ratio and network overhead metrics obtained with each simulation.

Fig. 26 shows the trend of delivery ratio (a) and overhead (b) for all simulation runs. Interestingly, the simulation that scored the highest delivery ratio (0.8055) was configured with FP equal to 0.1 and CP equal to 0.5, while the one with both FP and CP set to 0.1 showed the lowest overhead, as one might expect. However, what is most important is to find pairs of values that allow reaching delivery ratios close to the optimum and, at the same time, maintaining the overhead under control. For this purpose, we designed the following score function Sf_N:

$$Sf_N = \frac{{d_N}^2}{\sqrt[3]{1 + o_N}} \qquad (5),$$

where d_N and o_N are defined as in (3) and (4), respectively. I added the unit to the network overhead so that the radicand is always higher than or equal to 1. Additionally, I chose the power of two at the numerator and the root of degree 3 at the denominator to give a higher importance to differences in the network delivery ratio than in network overhead. In Fig. 27, I plotted the value of (5) for all pairs of FP and CP obtained in the simulations run in the first experiment.

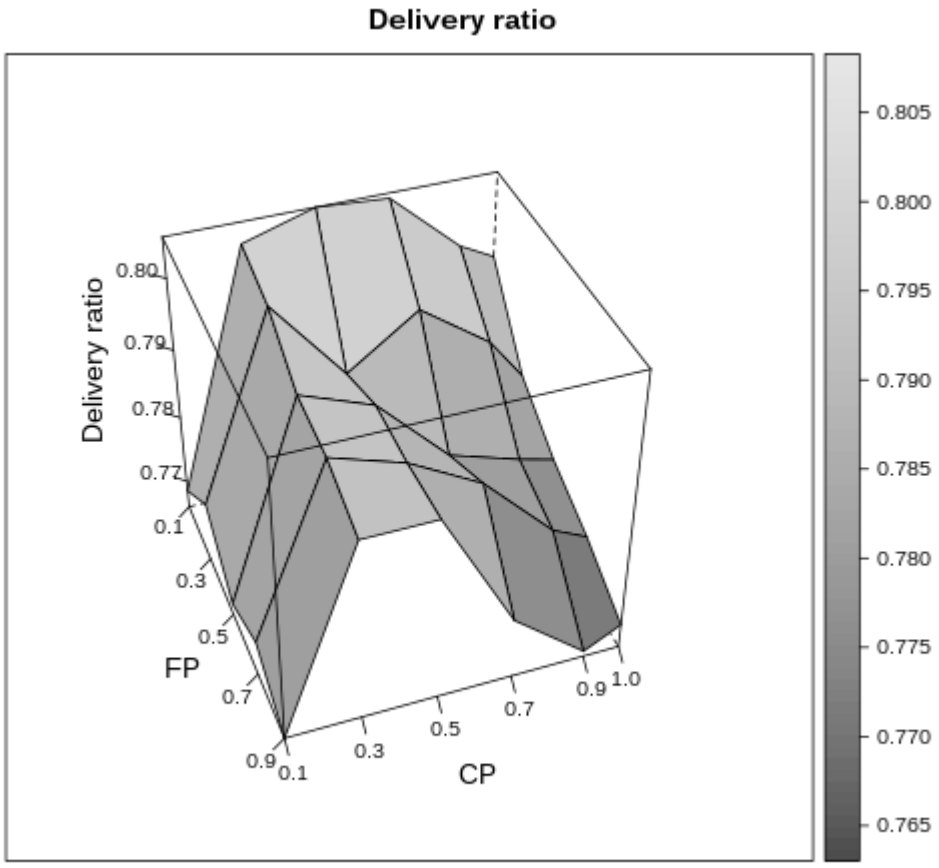

Fig. 26a Trend of delivery ratio varying the FP and CP parameters in our first experiment

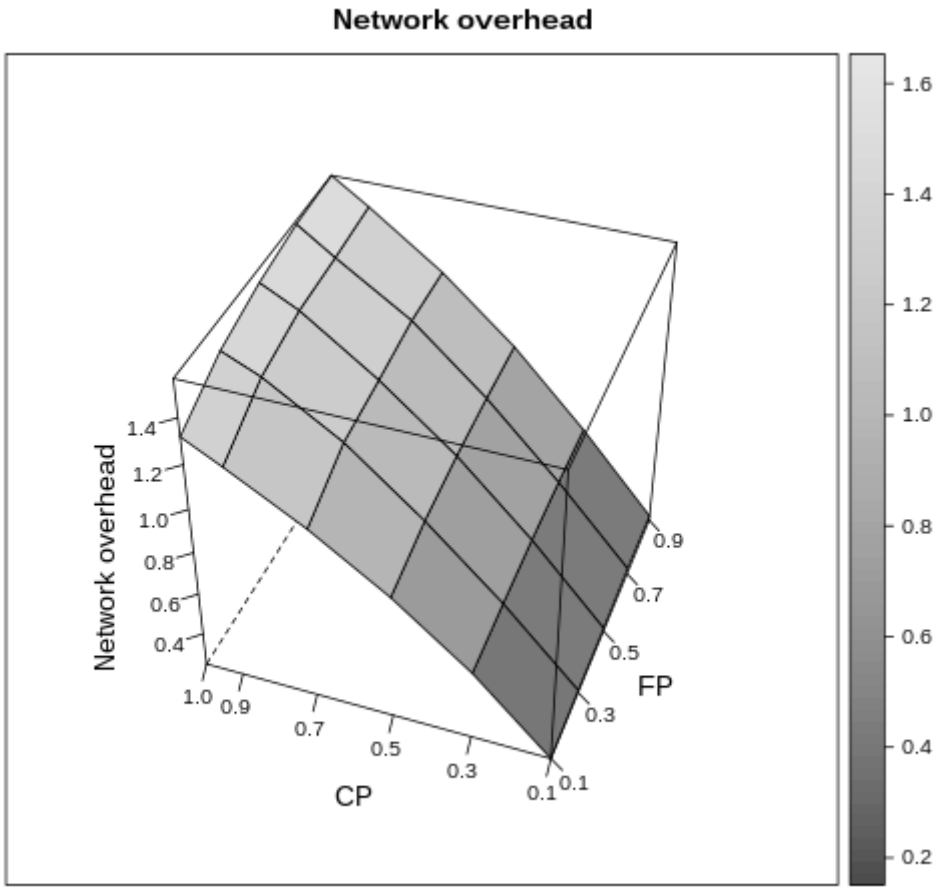

Fig. 26b Trend of the measured network overhead varying the values of the FP and CP parameters in our first experiment

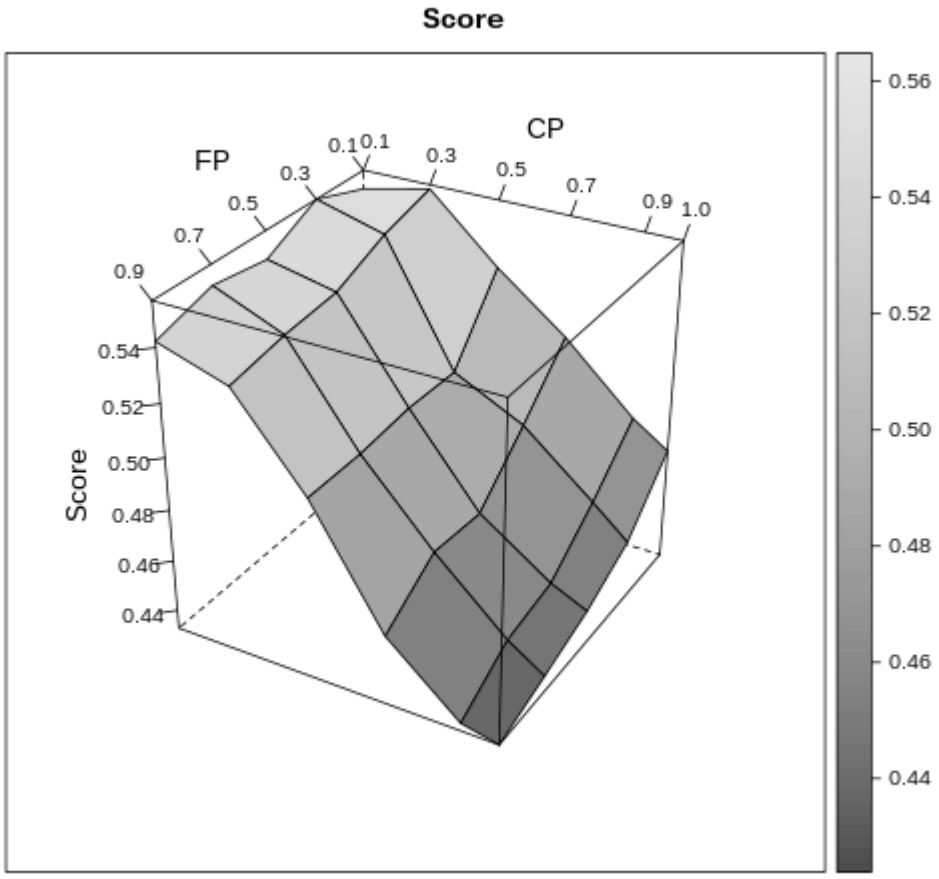

Fig. 27 Values of the score function Sf_N with the variation of the FP and CP parameters

The simulation configured with FP set to 0.3 and CP set to 0.1 is associated with the highest score, Sf_N= 0.5547, and produced a very high delivery ratio (0.8026) while keeping the overhead down to 0.5661. In the next experiments, I will fix the FP and CP parameters to these values. The measured metrics also show the strong impact of FP and CP on the performance of communications and first exhibit the potential of SP-ADCs. Analyzing the other statistics, one can notice that the average network latency stays comparable across all simulations. The lowest peak measured was 1173 seconds, which corresponds to values of 0.7963 and 1.2317 for d_N and o_N, respectively, and that was obtained setting FP to 0.1 and CP to 0.9. On the other hand, the highest latency found was 1313 seconds, achieved by setting FP to 0.9 and CP to 1.0, which corresponds to a network delivery ratio of 0.7693 and a network overhead of 1.5610.

5.1.5.3. Second Experiment: Comparison of the ADC Modes

In the second experiment, I ran 6 new simulations using the scenario illustrated in Section 5.1.5.1 and all possible combinations of dissemination strategy (EBRP and SnWBRP) and

ADC mode (strict, semipermeable, and unconstrained). As stated above, I configured the two simulations that used the semipermeable mode with the values 0.3 and 0.1 for FP and CP, respectively. The reader might note that the results obtained with EBRP/S-ADC and with SnWBRP/S-ADC correspond: this is the direct consequence of restraining message dissemination only among their destinations, against which both routing algorithms behave the same way.

Table 4 compares the network delivery ratio (as defined by eq. (3) above, in percentage), the median of the delivery delay (in seconds), the network overhead (as defined by eq. (4) above), and the total number of transmissions, message drops, and discarded messages for all six simulations. As one might expect from the characteristics of the two dissemination strategies, the values of delivery ratio and average latency obtained using the EBRP are slightly better than those obtained using the SnWBRP, whereas the SnWBRP achieved the lowest overhead. However, the most interesting aspect is analyzing their trend when varying the ADC mode. In fact, SP-ADC reaches the highest delivery ratio, which is counterintuitively much better than that measured using the unconstrained strategy. At the same time, SP-ADC keeps the overhead ratio under control and it does not affect the average latency significantly.

I believe that the explanation lies in the more frugal use of the available resources. In fact, reducing the number of total transmissions enabled a much better exploitation of the scarce bandwidth and decreased collisions significantly: results show -68.0 and -59.2 percent of collisions against a total number of message transmissions of -46.1 and -36.8 percent using the EBRP and the SnWBRP, respectively, when comparing semipermeable and unconstrained ADC strategies. In addition, SP-ADC reduced the number of message drops (-42.9 percent for EBRP and -34.2 percent for SnWBRP) compared to their unconstrained variants, hence a more efficient usage of the cache. At the same time, moving from S-ADC to SP-ADC increased the delivery ratio of 9.56 and 9.21 percent and reduced the expected delivery delay by 21.6 and 20.4 percent, using the EBRP and the SnWBRP respectively.

Table 4 Summary of the results obtained from all simulations performed during the second experiment

	Delivery Ratio	Delivery Delay (s)	Network Overhead	Num. of Transmissions	Num. of Collisions	Num. of Dropped Messages	Num. of Discarded Messages
EBRP + S-ADC	71.05%	975.2	0.07	47335	5927	43681	29215
EBRP + SP-ADC	80.61%	764.7	0.73	86557	15915	78447	34159
EBRP + U-ADC	77.1%	628.9	2.35	160653	49657	137377	0
SnWBRP + S-ADC	71.05%	975.2	0.07	47335	5927	43681	29215
SnWBRP + SP-ADC	80.26%	776.6	0.57	78227	16701	72842	27581
SnWBRP + U-ADC	76.86%	839.4	1.57	123028	40790	110251	0

5.1.5.4. Third Experiment: Evaluation of the Contribution of Tram Nodes

So far, ICeDiM was treating tram nodes exactly like any other node with respect to abiding by the rules enforced by the ADC mode. The goal of the third and last experiment is to appreciate better their potential for the purposes of message dissemination when used in combination with ADCs. In fact, given their high mobility, greater network and storage resources, and the access to an external, virtually infinite, power source, tram nodes represent a strategic resource to improve performance in NGN scenarios [99]. To achieve this goal, I collected statistics running multiple simulations where tram nodes have a special '*' subscription, which specifies their interest in messages that belong to any channel. This way, trams can take a more active part in message routing, regardless of the routing algorithm and the ADC mode chosen. In addition, with this last experiment I tried to achieve a more profound understanding of the impact that the size of the cache equipped on tram nodes has on the overall system performance. Therefore, I fixed the routing algorithm to SnWBRP and ran 15 simulations to test all possible combinations of the three ADC modes and five different values for tram nodes' cache size: 5 MB, 10 MB, 15 MB, 25 MB, and 50

MB (as explained in section 5.1.5.1). Again, I set the values of FP and CP in the SP-ADC mode to 0.3 and 0.1, respectively.

Fig. 28a compares the trends of the delivery ratio obtained by each ADC mode with the varying of trams' cache size. As one could easily expect, greater memory resources correspond to higher delivery ratios regardless of the ADC mode used. The semipermeable mode achieved the best result yet again, with a ratio of 0.8446, while using the strict and unconstrained modes yielded the values 0.8154 and 0.8026, respectively. Note that 0.8446 is the highest delivery ratio measured across all simulations performed, with an increase of 4.78 percent with respect to the value obtained using SP-ADC on top of the EBRP in our second experiment, which was the highest score reached so far.

On the other hand, Fig. 28b shows that the cache size does not seem to have a strong impact on the overhead ratio, which largely depends on the ADC mode used. The highest values measured are 0.0894 for the S-ADC, 0.4576 for the SP-ADC, and 1.3183 for the U-ADC. However, the comparison of these results with those measured in our previous experiment is very interesting. In fact, the increase in the delivery ratio shown above is associated to a reduction in the average network overhead: the SP-ADC and the U-ADC scored -19.72 and -16.03 percent, respectively. Contrarily, the S-ADC showed a growth of 27.71 percent in the measured network overhead. This suggests that exploiting tram nodes can also reduce bandwidth consumption effectively, when either the SP-ADC or the U-ADC is used.

Fig. 28c plots the average message delivery latency measured while varying the cache size equipped on the tram nodes. All curves are U-shaped; those obtained using the S-ADC and the SP-ADC have their minimum with a cache size of 15 MB, while the U-ADC yielded the lowest latency with a cache size of 10 MB. For small caches, the U-ADC mode shows better results, but the latency grows very quickly with the memory size. On the other hand, the SP-ADC mode shows higher latency values with small caches, but it grows more slowly than the unconstrained mode, thereby leading to lower latencies with larger memories. Moreover, it is important to note that the average values are computed considering only those messages that reached their destination. The significant difference in terms of delivery ratio between the S/SP-ADC modes and the U-ADC mode means that the curve for the U-ADC mode gets plotted from a significantly smaller data set. In particular, part of the missing values is composed of messages whose TTL expired before they could reach their destination, whereas the S/SP-ADC most likely succeeded in the delivery with high latencies, sometimes close to the TTL. Therefore, we have reasons to believe that the

latency values measured for the U-ADC is underestimated compared to those measured with the other two ADC modes.

The results presented in this Section show that trams have a great impact on the performance of the dissemination process. Based on these observations, Chapter 6 of this Thesis will investigate this matter further.

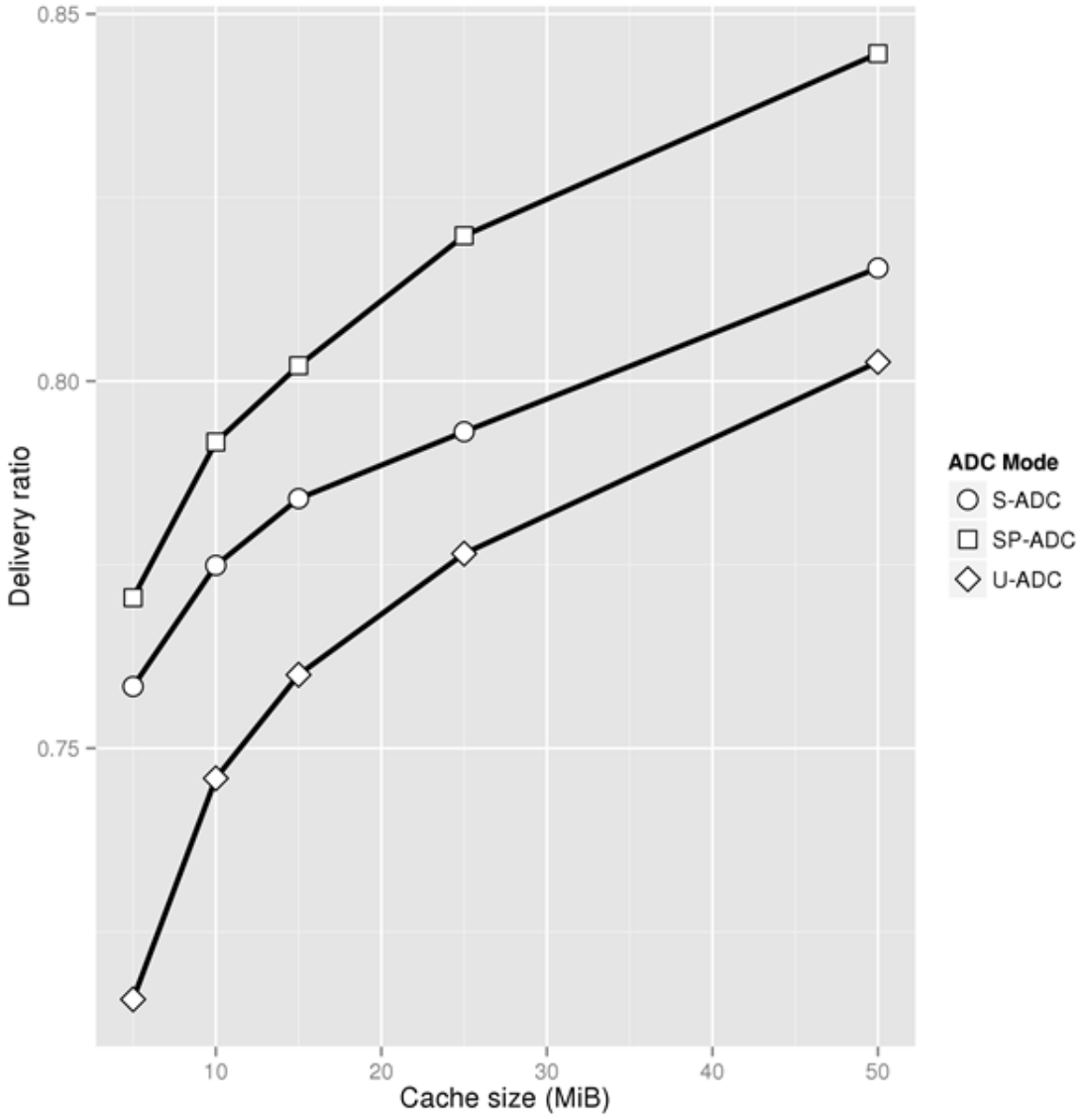

Fig. 28a Delivery ratio scores for each Application-level Dissemination Channel mode, varying the cache size on tram nodes

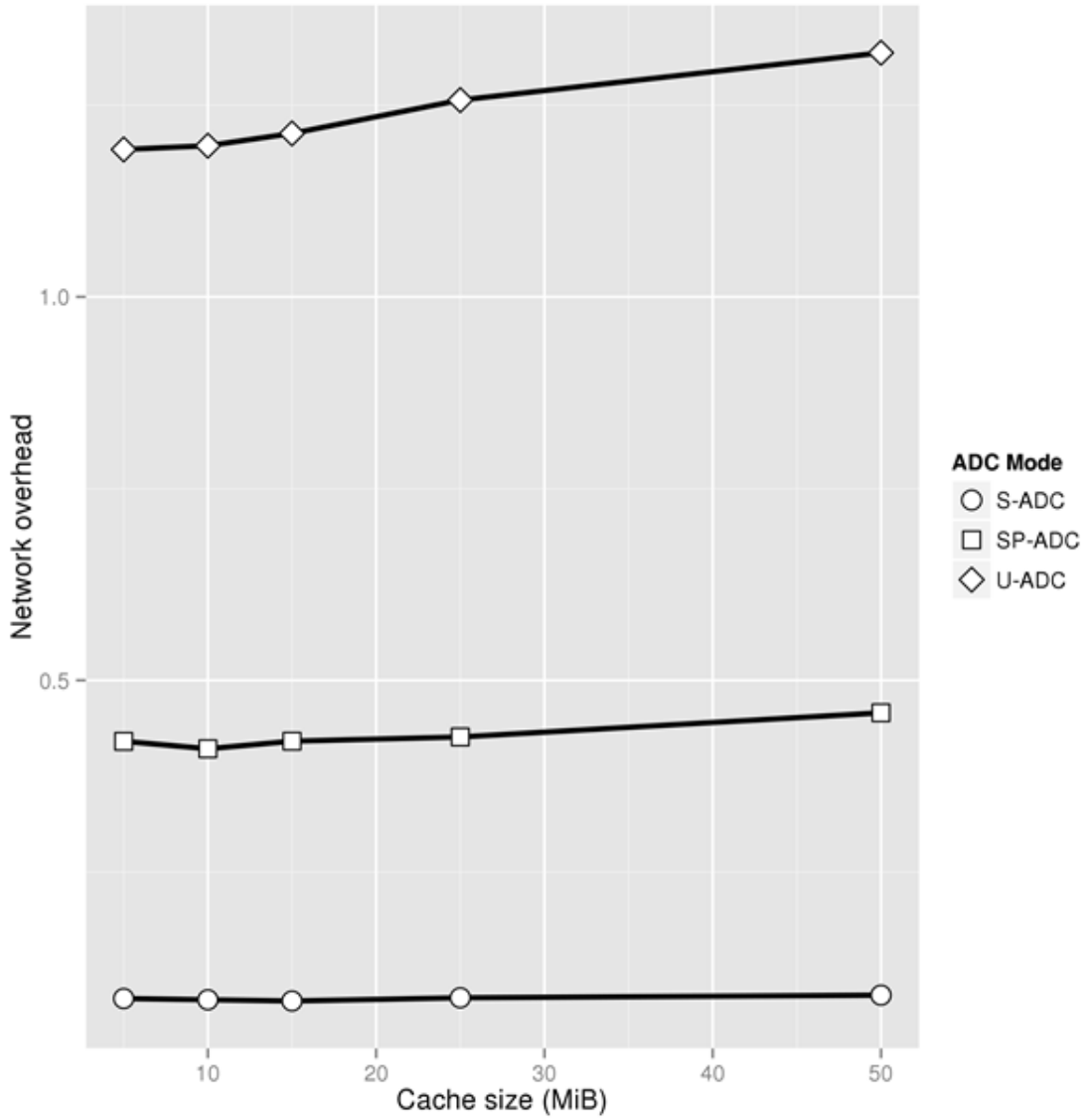

Fig. 28b Network overhead for each Application-level Dissemination Channel mode, varying the cache size on tram nodes

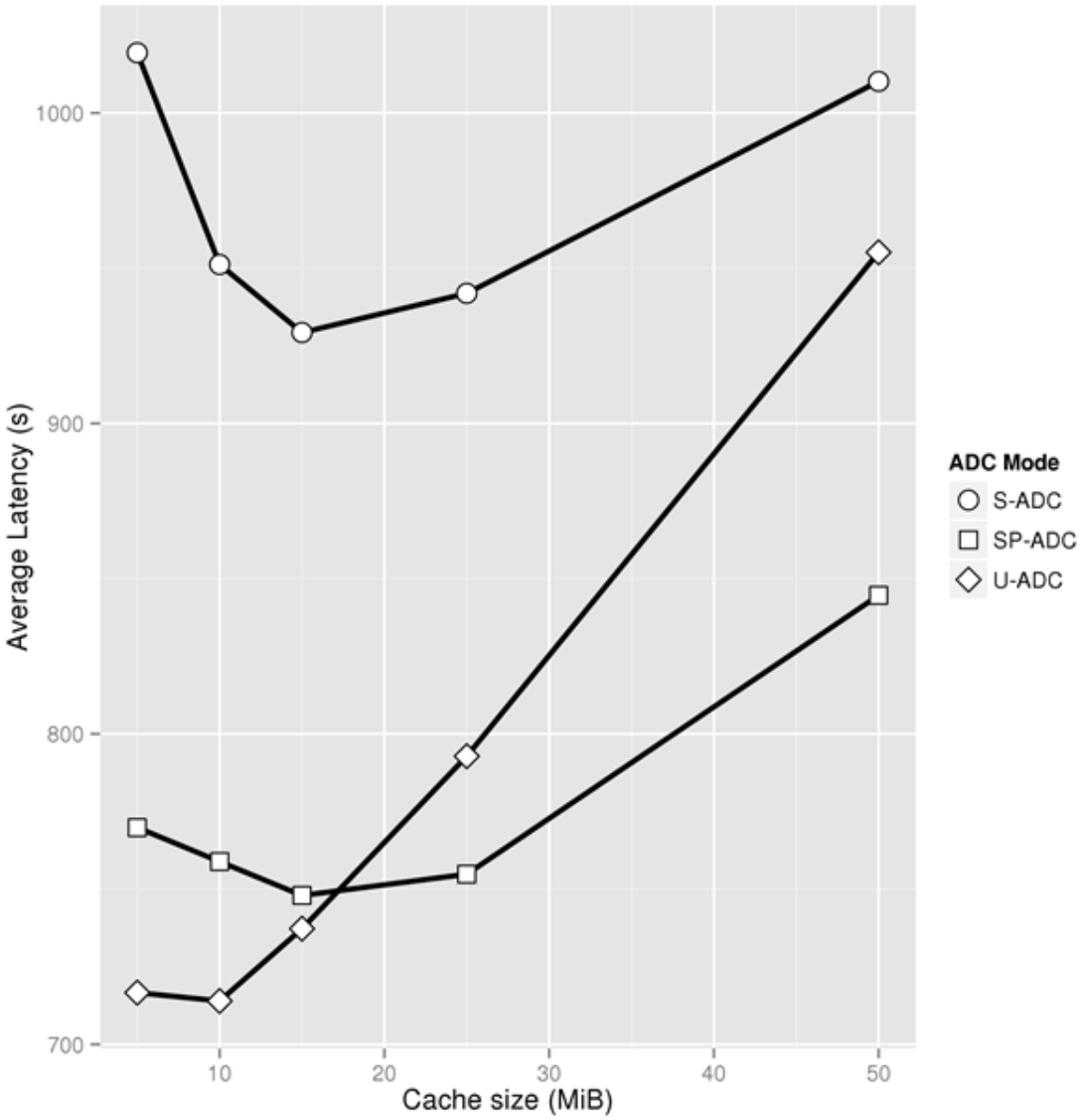

Fig. 28c Average message delivery latency for each Application-level Dissemination Channel mode, varying the cache size on tram nodes

6. LEVERAGING PREDICTIONS TO OPTIMIZE THE USAGE OF SCARCE RESOURCES

Given the peculiarities of the urban environment and the challenging requirements of next-generation applications, it seems natural to try to take the maximum advantage out of the Opportunistic Networking paradigm. By intelligently exploiting the intrinsic periodicity in the mobility pattern of some strategic nodes (e.g., buses, trains, and metros, but also commuters, who might drive every day the same path to work), it is possible to support the forwarding of data generated by sensors and mobile devices in a smart city to the wired network infrastructure [99].

This Chapter presents an extension of the ACM DisService component that adds the capability of predicting future encounters with resource-rich nodes to the middleware. The proposed extension relies on a mathematical approach that can detect a broad spectrum of periodically recurring patterns in nodes' mobility while keeping the computational complexity of the prediction model under control. Experimental results obtained in a simulated environment show that my solution produces accurate predictions about future node contacts that DisService can consider in its decision-making process to promote the offloading of the cellular network.

6.1. A Middleware for Opportunistic Networks

Fig. 29 below shows a smart city scenario. Surveillance applications are installed on sensor nodes equipped with a camera to take high-resolution pictures of the current traffic conditions in some critical areas of the city. To get the most out of this kind of applications, the captured images need to be gathered and stored in a data center, usually located on the cloud or in the smart city data center, where enough computational and memory resources are available to process them and derive useful information. The reader could imagine the cameras connected directly to the data center via 3G/4G communications. However, using the cellular network to move high amounts of data is very expensive, and it would worsen the problem of congestion, as described in Chapter 1.3.1.

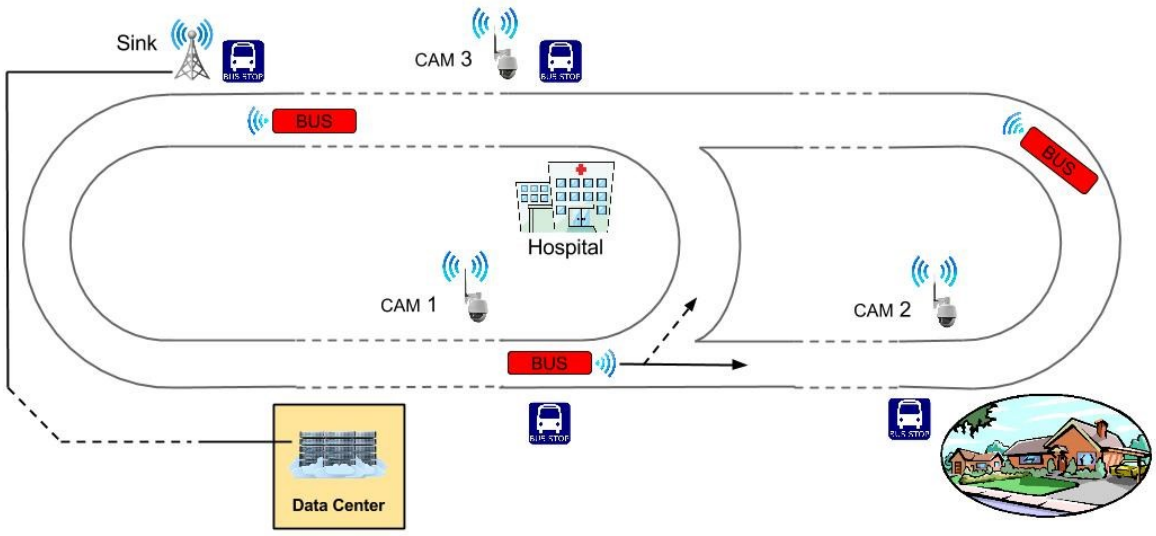

Fig. 29 Smart city scenario with a Surveillance System

An interesting option to improve nodes' connectivity is to opportunistically take advantage of mobile nodes that come into proximity and that could operate as "message ferries" between the cameras and one or more "sink nodes" connected to the data center that manages the smart city information layer. A possible solution could be, for instance, to equip the public transportation vehicles with Wi-Fi or Bluetooth devices, so that the camera nodes can use ad hoc links to send the images to buses, trams, and trains passing by. Those vehicles would then become message ferries, in the sense that they would carry received images to one or more sink nodes connected to the smart city data center, which I assume they would be intelligently placed along the routes of public transportation. Notice that also passengers on buses could exploit the proximity to a sink for uploading/downloading heavy data contents, like social network activities, videos, or high quality images, thereby avoiding connecting to the more expensive cellular network and contributing themselves to its offload.

In order for applications to be able to take advantage of new communication opportunities, they need to rely on an adaptive communications middleware designed for opportunistic networks, capable of analyzing the current network conditions and of exploring all surrounding connection opportunities. Such middleware would tailor the dissemination strategy based on the discovered connection opportunities, under the constraints that characterize each device, and will provide applications with a set of mechanisms and tools to define policies to match their goals.

Adaptive communications middleware require a complete and accurate representation of the network status and its resources in order to be able to satisfy application requirements. However, often the knowledge about the current state of the system is not enough. For

example, it is possible that a node not currently reachable will soon fall under the Wi-Fi range of another device. This would open new connection possibilities in the near future, although currently unknown. Therefore, to provide applications with all the information to design effective resource utilization policies, the communication middleware should implement techniques capable of predicting the presence of future resources, whenever possible. Implementing the prediction algorithms at the application middleware level facilitates reuse, extensibility, and maintainability of the whole system. In addition, this choice decouples the application logic from all strategies and functionalities that support the opportunistic discovery and management of available resources (routing, forwarding, message caching, prediction models, and so on).

Having the knowledge on future contacts with other nodes at their disposal, applications can implement smart disruption tolerant policies. Furthermore, predictions enable the design of policies that foster a fairer usage of the available resources. For example, prioritizing short-medium range communication technologies, like Wi-Fi or Bluetooth, against more expensive solutions, such as 3G/4G communications, entails a higher utilization of cheap networking solutions, effectively offloading the cellular network and improving the global performance of the smart city network [136].

Discovering periodic behaviors in nodes' mobility patterns is not a trivial task. In fact, in order to satisfy the needs of the citizens, the routes of public means of transportation might change periodically, for instance to adapt to congestion, to connect important areas of the city more frequently, or to serve different neighborhoods. Therefore, it is necessary to devise an approach that enables the discovery of a wide spectrum of complex periodic patterns that recur in nodes' mobility. This permits to predict future contact opportunities with potential communication resources and design advanced information dissemination strategies that favor the usage of alternative, cheaper communication solutions, such as Wi-Fi or Bluetooth, available in the majority of the modern mobile devices, over the cellular network, thereby contributing effectively to its offload.

6.2. Predicting Future Node Contacts

In order to make the best decisions when it comes to opportunistic routing, applications require a knowledge of the environment in which they are submerged that has to be as complete as possible. The knowledge required comprises the set of nodes available within communication range, their characteristics, and the NICs equipped, the network status, and any requirement that the applications might have in terms of bandwidth allocation, maximum latency, transmission reliability, set of destinations, etc.

Although this information is necessary for applications to select the best routing strategies, considering only the present network conditions might limit the output of the dissemination algorithm to a local optimum. In fact, in highly dynamic environments and under certain conditions, delaying the delivery of messages might open the door to new communication options that could reveal themselves to be better choices. However, systematically delaying sending any message to look for additional communication possibilities would extremely increase latency; this is unacceptable in some application domains and, anyway, never a desirable result.

To delay the messages delivery only when convenient, the communication middleware needs to provide applications with the knowledge about future contacts with potential communication resources that will be likely to happen. In order to do so, the middleware can exploit the history of past contacts with other nodes to build a forecast model capable of inferring the next contact times. However, due to the complex periodic behaviors that nodes can exhibit in some cases, the process of discovering the patterns underlying them is generally extremely complex, hence computing forecasts of future node contacts is challenging and computationally expensive. In addition, forecast models need to be continuously reevaluated to keep their accuracy within a certain level. Finally, there is the need to provide a measurement of forecast accuracy [137], so that applications can autonomously decide when to rely on the computed predictions.

For example, let us consider the bus route depicted in Fig. 29. The itinerary might have been conceived to prioritize the connections with certain areas of the city against others, a quite common situation in modern urban realities. Consequently, the bus might follow an itinerary that is not always the same, but varies accordingly to a predefined schedule. In the figure, the bus travels two different paths, distinguished by two different arrows: the dashed arrow represents the shortest path, whereas and the normal arrow represents the longest one. To connect the most important areas of the city (to the left in the figure) to the hospital with higher frequency, the route was designed in such a way that the bus would take the shortest path twice in a row, before taking the longest path once, and then it will start over, repeating the same pattern.

While unsophisticated forecast models would require low processing power, they would also fail to recognize many common nodes' mobility patterns or reach lower, possibly inadequate, levels of accuracy. For instance, a model that assumes that the nodes will follow a constant itinerary between consecutive contacts would fail to capture the behavior that characterizes the bus node in the scenario described above. Nonetheless, a completely

different approach, based on accessing the Internet to download the timetables of bus lines that pass by the camera, would present other problems. In fact, smart cities might have smart traffic management systems that exploit the cellular network to provide all interested nodes (traffic cameras, bus stops, traffic lights, etc.) with information on the next arrival time of buses at the requested location. However, all the traffic generated to update this information to all the nodes continuously would place an additional burden on the cellular network and contribute to its congestion.

An interesting possibility to reduce the traffic would be to limit the number of update messages to only one message, which is used to notify when a bus leaves the closest bus stop. With this information, a camera node nearby would simply have to learn the amount of time buses require to reach the camera from that stop to know when the next bus will pass by. That knowledge can then be strengthened by combining it with the times at which update messages were sent, in order to capture fluctuations in travel times due to varied street traffic congestion levels at different moments of the day/week/month. Finally, note that data gathered for distinct buses travelling the same path to reach a camera could be merged to reduce memory usage and to increase the accuracy of the predictions.

For the reasons expressed above, there is the need for advanced prediction models that can recognize complex recurring patterns in the nodes' mobility, leading to solutions that can perform well under many circumstances. However, the limited memory and computational resources available on sensors and mobile nodes require a trade-off between the accuracy, the refinement, and the complexity of the forecast model. The chosen trade-off can vary based on the characteristics of devices. Alternatively, the middleware might provide applications with a set of multiple models, each with different complexities and characteristics. In turn, the applications will be responsible for choosing the model which best satisfy their requirements.

In my work, I propose a general solution that, paying the cost of a more complex process than those necessary for simple solutions such as those described above, implements a model that can detect a broad spectrum of periodically recurring patterns in nodes' mobility. I believe the patterns that my solution can detect are realistic representations of those that characterize the intrinsic periodic behavior of many subjects of the smart city, such as the public means of transportation. The proposed solution exploits a mathematical approach that allows searching for and discovering periodic patterns while keeping the computational complexity of the model under control.

6.2.1. An Efficient Mobility Prediction Model for the Urban Environment

In a modern city, the intrinsic periodic behavior of public transportation allows us to approach the problem of detecting periodically recurring mobility patterns of nodes from a simpler perspective. In fact, public means of transportation equipped with a medium-range network device such as a Wi-Fi card, or with a small-range, low-power Bluetooth interface, can become mobile nodes with a very predictable behavior.

The largest part of public means of transportation either have a fixed schedule throughout the day (that is, the inter-arrival time at the same destinations stays almost constant), or they move according to a certain constant pattern that repeats itself with some periodicity (several times per day, daily, weekly, etc.). These observations reduce the complexity of the problem of finding predictable patterns in the nodes' behavior, as it becomes reasonable to assume the existence of periodically recurring patterns that underlie the intercontact times between two nodes. In addition, I can consider that discovered patterns will not change in the short period, since bus and train schedules and routes tend to remain unvaried for a long time, usually months or years. In this study, I addressed the latter case, where nodes move according to some periodically constant pattern, since the former is just a special, simpler case of the latter.

In a smart city, several categories of nodes could take advantage of predictions about future contacts with other nodes. For example, the surveillance application described at the beginning of chapter 3.1 could leverage predictions to implement a smart information dissemination policy, which aims at increasing the ratio of messages sent using cheap, short-medium range communication links, like Wi-Fi or Bluetooth, instead of more expensive ones such as 3G/4G channels. In fact, the knowledge derived from the prediction model enables informed decisions on whether to send images via one NIC or the other, in accordance with both the estimated likelihood that a bus will approach the camera in the near future and the urgency of the data.

The middleware I propose implements a prediction model that analyzes sequences of intercontact times collected for each node. It features a mathematical approach that can compute the autocorrelation of a time series, from which it enables the discovery of periodic patterns in the data. Moreover, the relatively low complexity of the proposed solution makes it appropriate to be employed on devices with low computational resources, such as sensors or smartphones.

The ability to keep track of the nodes' contact history is a key feature of my communication middleware, which allows it to gather the necessary data to feed its prediction model. This enables the forecast of the next contact times with the nodes and the computation of the predictions' reliability. The model can be configured with parameters that specify the maximum tolerance and the minimum accuracy and reliability allowed, so that applications are able to control the quality of the forecasts and change their dissemination strategy accordingly (readers can refer to [137] for a more detailed discussion on these parameters). This way, developers can implement applications that adopt adaptive and sophisticated dissemination strategies, based on the knowledge about the current state of the network and information about future contacts with strategic nodes, as provided by the middleware.

I further extended the communications middleware with the feature of collecting statistics on link durations. Combining this knowledge with the prediction of the next contact time, the middleware can assess the amount of data that can be transmitted to another node during the next contact window. This feature increases the middleware's adaptability, which provides overlying applications' with an evaluation of the bandwidth available during the next contact with a node, enabling the design of more robust and refined policies. However, investigating the impact that this functionality has on the dissemination process requires further work.

In the next three sections, I am going to introduce two possible approaches to detect periodically recurring patterns in nodes' mobility and the algorithm to predict the next contact time with other nodes implemented in the proposed middleware.

6.2.1.1. A Straightforward Approach for Detecting Recurring Mobility Patterns

Both the approaches I will present consider the sequence of intercontact times with a certain node η as the *finite time-series* x_n, where $0 \leq n < N$, and N is the number of intercontact times observed so far.

A straightforward way to extrapolate the periodicity of x_n is to calculate its autocorrelation function for some set of predefined lags (with the largest lag that cannot be greater than N/2). Nodes that public means of transportation, as discussed in the previous paragraphs, typically exhibit a regular, recurring mobility pattern, which repeats itself with some periodicity. This characteristic motivates the assumption that a *wide-sense stationary stochastic process* can describe the time series composed of the intercontact times

between a static node and a node that identifies a public means of transportation. In case of such a stochastic process, the autocorrelation function is defined as follows:

$$r_{xx}[\tau] = E[x[n] \cdot x^*[n - \tau]] \quad (1),$$

where τ is the lag for which the expected value E[•] is computed and x^* is the complex conjugate of x. Since x_n is finite for each value in the range [0, N - 1], the autocorrelation function $r_{xx}()$ also exists and is finite. After the computation of the autocorrelations for all the predefined values of τ, the value of τ that produces the highest output of the autocorrelation function represents the sought periodicity.

6.2.1.2. Exploiting the Wiener–Khinchin Theorem to Discover Recurring Patterns in Nodes' Mobility

The problem of the solution described above lies in its *complexity*. In fact, if k is the number of lags included in the search, the complexity of computing the autocorrelation is $O(n^2 \cdot k)$. Even if it is reasonable to assume $k \ll n^2$, the complexity is still quadratic in the length of the input.

To improve the efficiency of the search for periodic patterns in the time-series, I propose a different approach, based on the Wiener–Khinchin theorem and characterized by a smaller computational complexity. In the discrete-time case of wide-sense stationary processes for which the autocorrelation function, defined as in (1), exists and is finite, the theorem states that the spectral density S(f) of x_n can be computed from the autocorrelation, as follows:

$$S(f) = \sum_{\tau=-\infty}^{+\infty} r_{xx}[\tau] e^{-i(2\pi f)\tau} \quad (2).$$

From (2), it is possible to obtain the autocorrelation function r_{xx}[•] by computing the inverse Fourier transformation on S(f). Compared to the solution that directly computes the autocorrelation values, the complexity of performing the direct and inverse Fourier Transformations dominates the complexity of this second approach.

6.2.1.3. An Algorithm for the Prediction of the Next Contact Time

Starting from the result of the Wiener–Khinchin theorem, I will present two algorithms implemented in my communication middleware: first, an algorithm for discovering of the periodicity of recurring patterns in a time series, and second, an algorithm for the forecast of the next contact time with a node. It is important to note that the output of the first algorithm is part of the input of the second. Given X the vector containing all the intercontact

times observed so far for node η, I can define an algorithm to discover the pattern recurring in the samples in X with the following steps:

1. Compute the Fast Fourier Transform (FFT) of X:
$$Y = FFT(X)$$
2. Compute the spectral density S(Y):
$$S(Y) = Y \cdot Y^*$$
3. Obtain the autocorrelation vector R_{XX} applying the Inverse Fast Fourier Transform (IFFT):
$$R_{XX} = IFFT(S(Y))$$
4. Find the index p, with p > 0, for which the value of R_{XX} is the greatest.

p is the output of the algorithm and the periodicity with which patterns in X recur. The complexity of this second solution is dominated by the FFT and IFFT functions, which can both be computed with a complexity of $O(n \cdot log(n))$, where n is the size of the input.

The result of step 3 is a vector containing the values of the autocorrelation function computed over the input vector X for the lags in either the range [0, (N-1)/2] or the range [-(N-1)/2, (N-1)/2], depending on the implementation of the algorithms for computing the FFT and its inverse. In either case, given the symmetry of the autocorrelation function, the information contained in the output vector R_{XX} is the same.

If the number of samples in the vector X is large enough, the value of *p* returned by the algorithm in step 4 is the periodicity of the time series. Note that the described algorithm cannot discover periods greater than (N-1)/2. While this means that the algorithm needs the samples from at least two complete cycles to discover the periodicity in a time series, my experience with the problem, acquired from running numerous experiments and simulations with different input and parameters, suggests that the samples from three complete cycles are enough for it to produce sufficiently accurate results.

Once the periodicity in the data has been discovered, two more steps are necessary to predict the next contact time with node η. The first one involves the assessment of the next intercontact time. In order to do this, I used a technique based on the Exponentially Weighted Moving Average (EWMA), as described in [137]. Considering *p* the periodicity of the input vector X (*p* is the output of step 4 of the algorithm previously described), *t* the highest index in the nodes' contact history with respect to node η (with the first entry having index 0), *ewma_s* the value returned by each invocation to the EWMA function, and α the

smoothing parameter, the pseudocode of the algorithm that predicts the value of the next intercontact time is as follows:

```
time find_next_intercontact_interval (period p, contacts_vector X) {
      i = (t + 1) % p;
      ewma_s = X[i + 1].start - X[i].end;
      i += p;
      for (; i < t; i += p) {
            ewma_s = EWMA (α, X[i + 1].start - X[i].end, ewma_s);
      }
      return ewma_s;
}
```

The first three lines serve to initialize the *ewma_s* variable with a valid value before it is used as a parameter for the call to *EWMA()*. Finally, the forecast of the next contact time can be computed by retrieving the end time of the last contact with node η from the nodes' contact history and adding it to *ewma_s*, as returned by the *find_next_intercontact_interval()* function.

6.3. Experimental Study

I tested the proposed solution using a simulated environment to reproduce the scenario depicted in Fig. 29. More specifically, I used the Network Simulator 3 (NS3, available at the address https://www.nsnam.org/), version 3.16, for all the experiments concerning this work.

To enable message dissemination and replication in the simulated environment I used the DisService middleware, a component of the ACM. DisService supports applications by enabling the smart management of multiple links and by providing several message forwarding, caching, and replication strategies. These features characterize DisService as a general, effective solution for enabling Opportunistic Networking in challenging environments.

To support applications' adaptivity, I extended the statistics that DisService gathers in the World State by adding nodes' contact history and link durations. Based on those additional data, I implemented the prediction model described above within the middleware. DisService makes available the output of the prediction model to overlying applications, so

that they can build the strategy that best fits their requirements and the current status of the network, as inferred from the statistics.

For the computation of the Fast Fourier Transform (FFT) and the Inverse FFT I relied on the high-performance FFTW library (http://www.fftw.org), which includes fast routines optimized for several CPU architectures.

6.3.1. The Scenario

In the simulation scenario, there are five different NS3 nodes: three cameras, one bus, and one sink, as shown in Fig. 29. Every node has a standard 802.11b wireless interface installed [116], with a maximum available bandwidth set to 11 Mbps. In addition to Wi-Fi, the camera nodes also have a 3G-enabled interface which allows them to connect directly to the sink node. For the purposes of the simulation, I used an NS3 point-to-point radio link with a bandwidth of 1 Mbps to model the 3G connection between the cameras and the sink. A surveillance application is running on each camera. They generate messages containing highly detailed pictures of the monitored area that need to be delivered to the data center managing the smart city. Each node has installed DisService to handle both reception and forwarding of messages for the application using one of the NIC equipped on the node. DisService can store messages in a local buffer to deliver them to one or more sink nodes when they fall under nodes' range. Alternatively, it can deliver messages directly to the sink, using the 3G network.

The application running on the cameras implements the following policy for managing the cached images. If the cache is full and the camera takes a new picture, the oldest image is replaced (First-In First-Out policy, or FIFO). This behavior is consistent with the purposes of the application, because it is reasonable to assume that the goal of surveillance software is to deliver the most recent information. Nevertheless, to provide more flexibility and more control over the lifetime of generated messages, DisService allows applications to associate different priority levels to messages. This way, new messages can only replace older, lower-priority messages. The simple application I wrote for the experiments generates messages with 3 different priority levels: low (normal images), medium (images took at fixed intervals, to provide periodic updates about the status of the street traffic throughout the smart city), and high (images generated when specific events, like accidents or other hazards, are detected in the monitored area, or requested directly by the data center). Only messages belonging to the two higher priority levels can be delivered via 3G, if no other path to the data center is available. This restriction is necessary to avoid overloading the cellular

network with low priority traffic, which is reserved for urgent data. In all experiments, the average size of a picture is 5 MB.

Each camera is located more than one kilometer from the others and from the sink, and installed in a strategic area of the smart city, like a large crossroad, or a traffic light that regulates the vehicle flows in streets likely to be subject to congestion. Given the importance of those areas, it is reasonable to think that there might be at least one bus stop in their proximity. For this reason, in the simulation scenario I assumed the presence of a bus stop close to each camera and to the sink. The distance between each camera and the sink prevents any ad hoc communications via Wi-Fi. Therefore, the only way to use the cheap connectivity solution to deliver images is to exploit the temporarily available connections with a mobile node, like the buses in the considered scenario, which will function as ferries and carry the messages from the cameras to the sink.

DisService periodically broadcasts packets, called HELLO messages, to signal the presence of the node to its neighbors. Instances of DisService running on camera nodes will use the information derived from received HELLO messages to fill in their World State, including a vector containing all the intercontact times with bus nodes (which I named X in Section 6.2.1.3).

I modeled the bus movements with a fixed waypoint mobility model. In accordance with the behavior of some public means of transportation discussed in Section 6.2, the bus node in the simulation scenario does not follow a constant route, but it changes periodically. As in Fig. 29, the node can take two possible paths whenever it reaches the fork near CAM #1. In the experiments, the bus will take the shortest route twice in a row (identified by the dashed arrow in the figure), and then drive the longest path the third time (identified by the other arrow). These choices will then be repeated until the end of the simulation, thus identifying a pattern that recurs with periodicity of three.

Two bus stops and the sink are located along the short path, while the bus encounters all stops and the sink when traveling the long path. 32 and 40 segments describe the short and long paths in the simulator, respectively. The bus travels over those segments each time with a different speed, randomly chosen from a uniform distribution that ranges from 46.8 km/h to 57.6 km/h. The bus also remains at each stop a random amount of time, uniformly distributed between 30 and 40 seconds, before resuming its ride. The choices above introduce a certain degree of variability and permit to simulate the effects of small changes in the current traffic conditions and the effects of other elements on the bus's behavior, as well as to evaluate the robustness of the proposed solution.

I performed eight different simulations, to cover all the possible configurations of the prediction algorithm (enabled and disabled) with four different values for the buffer size allocated by DisService on the camera nodes (5, 10, 15, and 20 messages). When making this choice, I considered the possibility that other applications might be running on the cameras, and so different values for the buffer size represent situations where other applications are taking up a smaller or larger part of memory available on the nodes. Each simulation ran for 6 hours of simulated time. During the first two hours of simulation, cameras generate no messages: this allowed DisService to collect enough information about the mobility pattern of the bus node to feed its prediction model. In addition, this made possible a fairer comparison between the solution with predictions and the one without them. The chosen amount of time was adequate to generate enough messages for each priority class in order to collect significant statistics.

6.3.2. Results

During the experiments, I collected data representing the status of the simulations to elaborate statistics to describe the evolution of tests. Fig. 30 shows the Wi-Fi delivery ratio, that is the percentage of messages delivered to the sink node via Wi-Fi, i.e., that reached the sink node via the bus, against the buffer size, for the cases with predictions enabled and predictions disabled.

Independently from the specific buffer size, the performances in terms of Wi-Fi delivery ratio are significantly higher when camera nodes can leverage predictions about the forthcoming arrival of a ferry node. With predictions disabled, the Wi-Fi delivery ratio goes from about 67% to 81% when the buffer grows from 5 to 20 messages, whereas, with predictions enabled, those percentages range from about 86% to almost 94%. Labels in the figure show how many messages were delivered using Wi-Fi against the total. The difference between the two series of data ranges from 217 messages, with a buffer size of 5, to 143 messages, with the buffer capable of storing up to 20 messages. Considering the average message size of 5MB, enabling predictions redirected about 700MB-1GB of traffic from the cellular network to the ad hoc opportunistic network, corresponding to about 12-19% of the total traffic in the simulations.

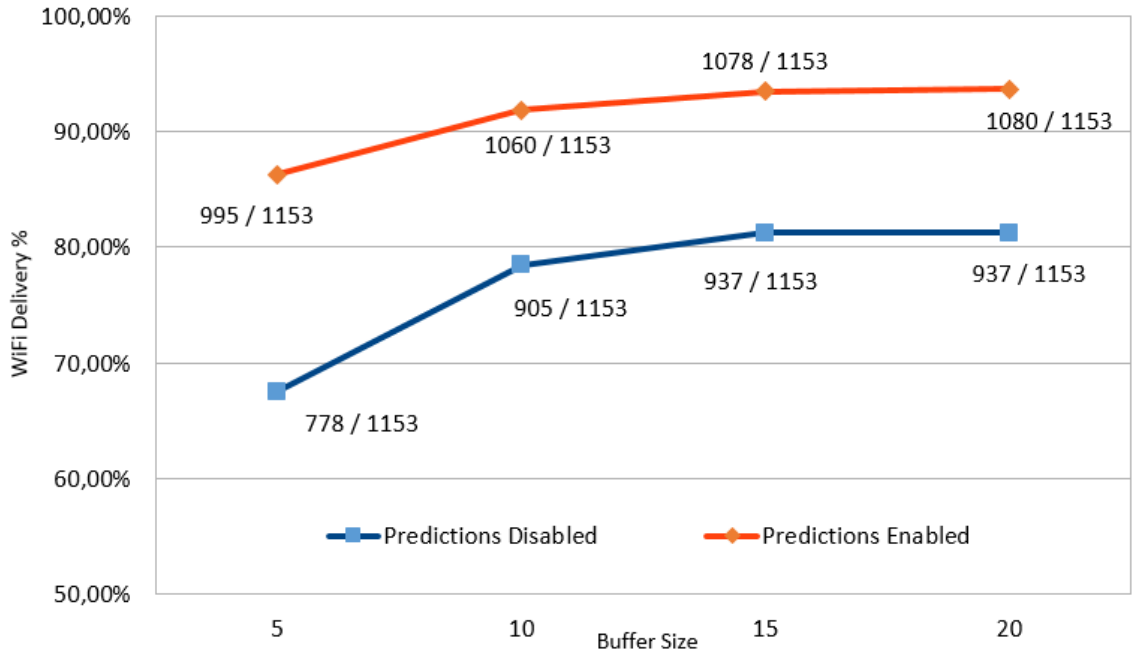

Fig. 30 Ratio of messages delivered via Wi-Fi to the sink node against the buffer size with predictions enabled and disabled

We believe that these results show a very important point. In fact, knowing in advance if a new resource will soon be available, applications can develop smarter policies that enable them to reach their goals with better resource usage. This includes a more uniform usage of the network resources, which will lead to a better QoS throughout the system by reducing the load placed on the congested parts of the network.

The Wi-Fi delivery ratio metric proves the efficiency of the proposed solution as a means to reduce the load on the 3G network by increasing the use of low-medium range connectivity solutions. Instead, Fig. 31 shows the impact of enabling predictions on the *delivery ratio*. It is possible to notice that the results for medium and high priority messages are the same, while only low priority messages suffer from a reduction in the delivery. Although the results may seem counterintuitive at first, the behavior they delineate is a direct consequence of the limited buffer size.

In fact, when an application generates a message, if valid predictions of future contacts with ferry nodes are available, DisService will store the message in its buffer and delay its delivery, waiting for the next ferry to approach. As the buffer fills up, low priority messages will be discarded, thus affecting negatively the delivery rate. However, an increase in cache size would mitigate this effect. As appears in Fig. 31, the difference between the number of messages delivered with predictions enabled and those delivered with predictions disabled drops significantly with a cache capacity of 15 messages or higher. This suggests that, in

order to get the best out of DisService predictive system, cameras should be equipped with enough memory to store all messages while waiting for the next bus node to arrive.

These observations and results demonstrate that the exploitation of future connectivity resources requires a higher memory usage. In section 6.2, I discussed the costs in terms of computational power to set up a prediction model; now, experimental results allow me to point out that a different buffer management arises from the exploitation of predictions of future contacts with potential connectivity resources. Because of it, messages tend to occupy the buffer for longer times when the prediction feature is enabled and, consequently, a higher number of low priority messages becomes eligible for replacement. In the experiments, I have considered a scenario where DisService provided each application relying on it with a static buffer size, and so each application could use up to a fixed amount of memory on the node. Nonetheless, I believe that a dynamic management of the buffer share could help mitigating the effect of predictions on memory management. Moreover, since storage is typically the cheapest resource, I believe that this does not represent a severe issue.

Finally, I would like the reader to note that I designed the experiments so that bus nodes would mirror as much as possible the mobility behavior of real public means of transportation. To this end, I included in the simulations random inter-arrival times, random node velocity that changes with each street segment, and variable routes. In addition, the efficacy of the Wiener–Khinchin theorem, and in turn of my solution, does not depend on the mobility pattern manifested by nodes, as long as it encapsulates some periodic behavior. To verify this, I ran multiple tests changing the lengths of the short and long paths and varying the type of pattern, i.e., changing the number of times the bus would travel the short and long paths before repeating the scheme. The results were all comparable with those reported above.

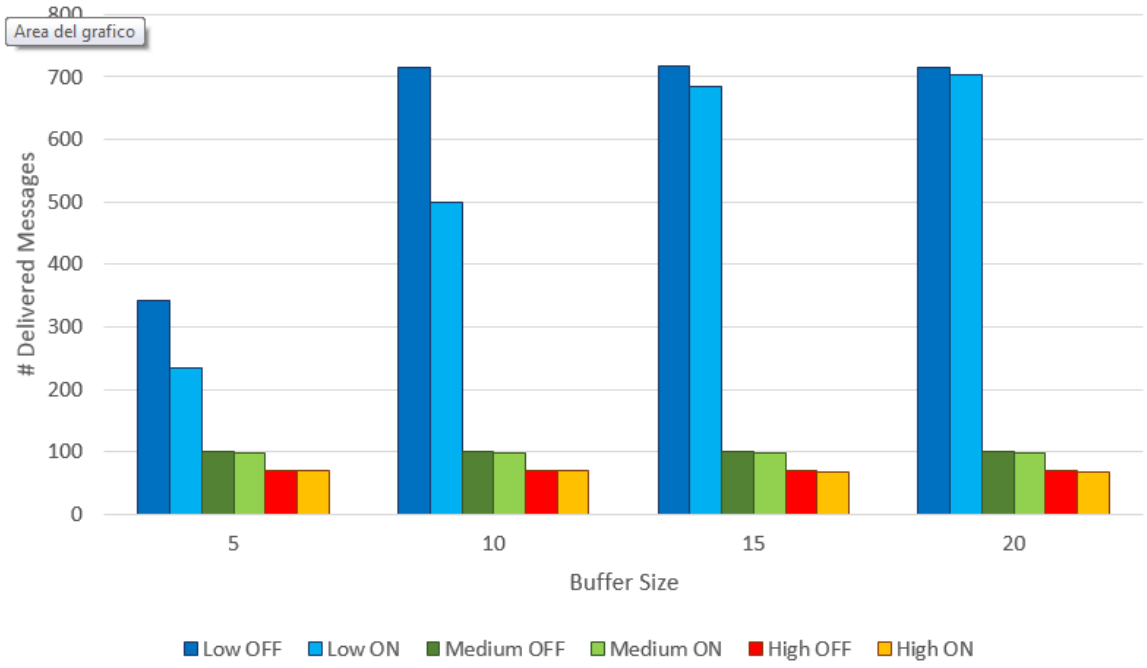

Fig. 31 Number of messages correctly delivered to the sink node against the buffer size with predictions enabled and disabled

7. RELATED WORK

The research literature on communications in TENs recognizes the efficacy of middleware-based approaches. Both [138] and [139] focus their effort on optimizing the allocation of resources between competing applications and nodes in the network. More specifically, the authors of [139] 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. NetProxy, instead, takes a different approach and focuses on providing QoS enhancements to applications transparently and remapping their communications over other components of the ACM in order to increase efficiency and reduce the impact on the network resources.

In [138], the authors propose QAM, a QoS-aware middleware for communications in tactical environments; to the best of my knowledge, this is the most similar work to 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 multiple components of the 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.

The authors of [115] also propose a solution based on a transparent network proxy. Their approach aims at increasing the performance of TCP by implementing advanced buffer and packet management solutions for 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. NetProxy 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 fit applications' needs under a broad range of scenarios.

Other proxy-based solutions, such as I-TCP [112], Mobile-TCP [113], and the Remote Sockets Architecture [114], have emerged in the past with the goal to improve TCP in wireless networks. However, in contrast with NetProxy, these systems are not transparent to applications and do not provide any specific QoS features to meet applications'

requirements, but they are limited to increasing the throughput of TCP in wireless networks and its resilience to mobility.

The ACM NetProxy can be classified as a splitting distributed Performance Enhancing Proxy (PEP) [140]. 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, NetProxy supports other protocols besides TCP and it adapts to a variety of networks. Moreover, NetProxy can be configured to provide a collection 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 [141]. 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. The authors of [142] 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. However, the paper focuses only on the problems of clustering and channel allocation. Finally, Kim et al. present a QoS framework for tactical networks based on commercial technologies like DiffServ and SNMP [143]. The framework assumes a hierarchical network architecture with leader nodes that enable communications between one layer of the hierarchy and the one above it. 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.

The scientific research on ICN has produced many implementations. They include DONA [144], CCN (http://blogs.parc.com/ccnx/) [145], NDN (http://named-data.net/) [146], 4WARD (http://www.4ward-project.eu/) [147], SAIL (www.sail-project.eu), PSIRP (http://www.psirp.org/), and its continuation PURSUIT (www.fp7-pursuit.eu) [148], but the list is by no means exhaustive (a more complete catalog and review of all available implementations of the ICN architecture is given in [17] and [83]). However, these solutions have been developed for the wired Internet and assume the availability of powerful routers where IOs can be cached [94] [117]. Some studies have already investigated the possibility of employing and adapting content-centric networking to address the challenges of

MANETs and wireless networks [73] [75] [80] [149]. The studies conducted in [73] and [149] also present some experimental results focused on analyzing the performance of different forwarding strategies, although they do not propose a complete ICN-based solution for MANETs. More specifically, the results obtained by Amadeo et al. support the use of flooding-based techniques in case of one-to-many and many-to-many communication schemes, or when the number of hops between source and destination in one-to-one schemes grows too large [73]. Similarly, the authors of [149] argue in favor of employing the content-centric view typical of ICN-based approaches to improve the performance of Opportunistic Networking in NGN scenarios. However, the experimental analysis in [149] is limited to showing that content-aware social-based routing solutions perform better than their content-oblivious counterparts do in terms of message delivery ratio, cost, and average latency.

In the literature, there exist just a few examples of ICN-based frameworks designed to support communications in dynamic mobile and heterogeneous networking environments. Amadeo et al. proposes a Content-Centric architecture for IEEE 802.11 MANETs called CHANET [150]. CHANET relies on naming to identify the content and it makes use of broadcast for the transmission of both interest packets and data. Other interesting techniques implemented in CHANET to increase effectiveness in the wireless environment are the overhearing of nearby nodes' transmissions and the local decision-making processes regarding packets forwarding. The results achieved experimentally show that their approach performs better than traditional TCP/IP-based solutions in terms of network overhead and download time. Despite the very interesting outcome, the authors do not investigate performances in large-scale, realistic scenarios, nor do they address critical topics such as the resources consumption on the mobile nodes. Finally, the paper does not describe the nodes' mobility model used in their experiments.

In [123], Detti et al. present TPS-CCN, a topic-based publish/subscribe solution that adopts the routing strategy implemented in CCN and adapts it to work in MANETs. The changes made to CCN are oriented to provide the support for delay-tolerant communications and publisher discovery, which are fundamental features in mobile networking environments. The publish/subscribe system implemented in TPS-CCN is based on the concept of topics, which effectively controls the in-network caching process across the network in a manner similar to S-ADC in ICeDiM. Content delivery in TPS-CCN follows a pull model in which applications can choose between reliable and best-effort delivery semantics. Despite the performance improvements that the support for delay-tolerant communications and in-network caching produce in TPS-CCN, it is worth noting that the authors set up a very favorable and simple experimental scenario. In fact, they ran their tests in a simulated

environment with 15 nodes moving around in a 200x200 meters area according to a random waypoint mobility model and equipped with an 802.11g Wi-Fi interface capable of transmitting within a range of 50 meters. In these conditions, end-to-end paths from source to destination are likely to be present, which greatly simplify the routing process; the authors themselves state that their solution was not designed for sparse networks. Additionally, the authors admit that employing a push model would be more efficient, but it would require additional work to introduce a new message type in TPS-CCN and it might not be able to satisfy reliability requirements. Therefore, a hybrid model would guarantee the highest flexibility to the system. This statement is also supported in [77], where the authors present MANET CCN, a communications middleware for tactical and emergency scenarios based on CCN. However, their solution is not valid in general, as MANET CCN takes advantage of the intrinsic hierarchical organization of tactical MANETs to drive information dissemination in the network.

The authors of [76] propose an opportunistic, content-centric architecture that takes advantage of the increasing number of pervasive systems available today to share contents. The proposed solution, called the Information and Context Oriented Networking (ICON) framework, encompasses techniques that come from both the research fields of data-centric networking and Opportunistic Networking. ICON exploits caching strategies developed in ICN to share and place contents across devices in the network, and it relies on opportunistic strategies based on social, location, and application data usage knowledge to route and forward the content to nodes interested in it. The authors describe the framework architecture and give details on the structure of exchanged messages. Finally, they discuss the applicability and feasibility of ICON and present the results obtained comparing their solution against other social-aware and social-oblivious frameworks based on Opportunistic Networking using the ONE simulator and a scenario based on real mobility traces. However, the decision-making process designed in ICON does not take into account the resources available on the other devices, and so it cannot discover the presence of rich nodes in the network that might be exploited to increase the system performance. Additionally, presented results are obtained in a simulated scenario where a single source node generates messages for one or more recipients that, instead, do not produce any packet; such scenarios fail to recreate the many-to-many communications scheme that characterizes next-generation networking environments.

Several research efforts showed the importance of taking into account nodes' position to improve the performance of the routing and forwarding processes. The authors of [151] and [152] present Position-based Opportunistic Routing (POR) and Location-aware Opportunistic Content forwarding (LOC), respectively, two solutions both based on the

knowledge of the destination's' location information to select the best next-hop among one node's neighbors. POR also features the usage of broadcast transmissions, which allows the routing algorithm to be very robust against some types of attacks like selective message dropping. Instead, LOC is based on a more advanced exploitation of the information on nodes' positions and achieves higher delivery ratios and lower resource consumption than other social-based approaches, such as Bubble Rap [153] and Lobby Influence [154]. Notwithstanding the good performances achieved, both POR and LOC assume the availability of a centralized system that nodes can query to obtain nodes' current location, instead of predicting it using some other kind of knowledge previously available. This raises privacy-related and single point of failure problems that the authors do not discuss.

Karamshuk et al. [15] show the importance of accurate human mobility models to increase the effectiveness of predictions and improve the routing process in Opportunistic Networks. Effective models exploit the network's context, such as knowledge on users' home addresses, work place, closest friends, and most frequently visited locations, to predict future contact opportunities and identify better candidates for relaying the data towards their destination. Song et al. [155] analyzed human trajectories extracted from traces coming from data acquired by cell towers of mobile phone carriers and empirically computed the entropy of users' mobility. The results show that there is a 93% potential predictability in users' mobility, due to the inherent regularity in human behavior, regardless of population heterogeneity. The results achieved by these works imply that there is great potential to improve routing in mobile networks by exploiting predictions concerning node mobility and future locations.

The work of Cheng et al. [156] is a step towards this direction. In their paper, the authors present GeoDTN+Nav, a routing protocol for VANETs that puts together the results achieved by [157] and [158] (works conducted in the areas of wireless networks and VANETs, respectively) with a system that derives from research efforts in the field of DTNs to enable packet delivery in potentially disconnected VANETs. In order to predict the future positions of certain nodes, GeoDTN+Nav takes advantage of the information coming from the navigation systems equipped on private and public vehicles. Thereby, the authors assume that all vehicles are equipped with a navigation system and that nodes are cooperative, hence willing to share with other nodes their travel destination and the path they will take. This solution solves the problem of having a centralized system that stores all nodes locations, but it raises other privacy-related issues. The authors address them by saying that it would be possible to reduce the amount of information shared or introduce noise in it in order to conceal the real destination of vehicles, at the expense of less accurate predictions.

Several solutions have been proposed in the literature that leverage predictions to increase the performance of routing in DTNs and MANETs, but significantly reduce the problems related to privacy. In [67], the authors present PROPHET (as in Probabilistic Routing Protocol using History of Encounters and Transitivity), which introduces the concept of delivery predictability P(a, b), namely the probability of delivering a message to the destination (node b) by forwarding it to node a. Each node builds its own delivery predictability table in an empirical manner by considering past contacts with other nodes. In addition, whenever a node enters another node's connection range, they exchange vectors that contain the delivery predictability information collected so far. Then, in PROPHET, a node forwards messages only to neighbors that have higher delivery predictability than itself. Similarly, Spray and Focus (SnF) [69] builds on top of Spray and Wait (SnW) [66] and adds a utility function based on the time elapsed since the last encounter between the potential relay node and the destination. The utility computation for each neighboring node drives the forwarding phase of SnF.

Along this line of research, the authors of [159] propose PRO, a routing protocols for pocket switched networks that relies on a combination of a probabilistic (also called observation) and a social-based dissemination utility function. PRO exploits periodic patterns in node encounters to build a profile for each node from which it can predict future contacts. The observation function selects as next-hop the nodes that are expected to encounter the destination earlier than others are. If no information about the destination is available, PRO falls back to dissemination mode, which takes advantage of the social properties of nodes to spread the message to different communities. History Meeting Prediction Routing (HMPR) [160] also relies on the history of past encounters, but the implemented utility function additionally takes into account the efficiency of bandwidth usage and the average contact duration of links between pairs of nodes to drive message forwarding.

PROPHET, SnF, PRO, and HMPR are all examples of probabilistic approaches: they predict future contacts based on how often two nodes met in the past, but they do not consider how long it will take before the next encounter. PRO is one step ahead of the other solutions, because it prioritizes nodes that will encounter the destination first, but it does not predict the exact contact time. In addition, all the approaches mentioned above assume that all nodes have a cooperative behavior. Predict and Spread (PreS) [161] tries to improve on the former issue by constructing a time homogeneous discrete Markov chain model to predict nodes' movements between a set of frequently visited venues, called main-venues or hubs [162]. Based on the model built, nodes using PreS can select the neighbors with the highest probability of encountering the destination within the shortest time. However,

PreS still requires a completely cooperative environment to function and previous knowledge on the ID of all main venues across all nodes. Furthermore, in their model the authors assume that each node will stay in any main venue long enough to deliver all messages they carry and do not take into account connection opportunities during transfers from one location to the others. This limits the model usefulness only to some scenarios, like the one of a university campus that the authors provide as an example.

Differently from the previous solutions, the prediction-based extension of DisService relies on the discovery of the periodic patterns that underlies the mobility of strategic nodes in a smart city network, such as buses and trams, to build a prediction model. Predictions are provided to DisService in terms of next expected contact time and prediction accuracy. The middleware leverages the model to choose the best NIC for message delivery, with the final goal of improving the offloading of the cellular network. The low memory footprint and contained computational cost of the model make it suitable for use on nodes with scarce resources. Moreover, I believe it is perfectly reasonable to assume the cooperativeness of nodes like trams and buses. In fact, public means of transportation are managed by the public administration, which would benefit from reduced cellular network bandwidth usage, hence lower costs, by exploiting buses and trams as ferries to move data from sensors to the smart city data center on the cloud.

8. CONCLUSIONS

In this Thesis, I analyzed the issues that arise in challenging networking scenarios like tactical edge and next-generation networks from the communications perspective. Part of these problems, especially when considering tactical edge networks, are caused by the very harsh environment conditions, which present significantly high channel packet loss rate and latency, facilitate links disruption, undermine information availability in the network, and can only take advantage of very scarce communication, computational, and memory/storage resources. At the same time, applications and services, with particular reference to those that operate in next-generation networking scenarios, will implement advanced social- and location-based features and will strongly interact with surrounding devices. Such functionalities pose stringent requirements on the network QoS, necessary for applications to work properly and provide a satisfying experience to the users.

The severe operating conditions of tactical edge networks and the advanced features on which next-generation applications will rely expose the inadequacy of traditional communication solutions, which perform very poorly in challenging networking environments. Additionally, connection-oriented approaches do not fit well with extremely dynamic networks, where nodes' mobility might cause temporary network partitioning and frequent disruption of end-to-end communication paths, thereby hindering considerably the effectiveness of those approaches. Thus, the insufficient support that traditional communication solutions offer to applications operating in challenging networking environments make life difficult for developers, who cannot reuse COTS solutions nor rely on third party libraries built on top of traditional protocols such as TCP or UDP. This calls for communications middleware that implement paradigms specifically designed to support next-generation applications and enable the reuse of third party and COTS software components.

In my work, I proposed solutions to address many aspects of the aforementioned problems. More specifically, my contribution to the state-of-the-art of research on tactical edge networking environments focused on the design and implementation of a network proxy solution, called NetProxy, to enable the reuse of COTS, legacy, and SoA-based applications in tactical edge networks. The main features of NetProxy is protocol remapping, which enables NetProxy to remap traditional protocols to components of the Agile Computing Middleware specifically designed to operate in extremely challenging networking environments, like Mockets and DisService, in a completely transparent fashion; this is a key feature to enable software reuse in tactical edge networks. I also implemented in NetProxy other important features to enhance the QoS offered to tactical applications, which

include stream compression, packet filtering, message consolidation, connection multiplexing, and flow prioritization. Finally, I designed two different operational modes for NetProxy, namely Host Mode and Gateway Mode, which match different requirements and adapt to different network configurations.

On the topic of managing the scarce resources available in next-generation networking environments, I worked on the design and implementation of ICeDiM, an information dissemination middleware based on the Information-centric Networking paradigm for extremely dynamic wireless mobile networks of the next-generation. ICeDiM leverages the concepts of Application-level Dissemination Channels (also referred to as Strict Application-level Dissemination Channels) and their Semipermeable variant. Those concepts address the problem of keeping the resource consumption of information dissemination policies under control while still providing high performance in terms of message delivery ratio and latency. Additionally, Application-level Dissemination Channels define thematic channels that nodes need to join in order to publish messages within the context of a channel. By becoming members of a channel, nodes commit themselves to share resources with other members and, at the same time, they receive support from other members. Thereby, Application-level Dissemination Channels also try to solve the problem of motivating and regulating resource sharing between nodes in a network.

I then continued this line of research by designing an extension of the Agile Computing Middleware DisService component to predict future contacts with nodes that are potentially important networking resources. Providing applications with knowledge on the future availability and estimated duration of cheap, high bandwidth links enables applications to design smart resource management and prioritization policies and exploit opportunities to offload the cellular network. Results showed that predictions are efficacious to promote mobile offloading while not affecting the delivery rate of high priority messages.

I can visualize many directions towards which my research activity could proceed in the future. Probably, one of the most interesting ones involves adding to NetProxy a network sensor component that takes advantage of Gateway Mode to implement a continuous monitoring of the internal and external networks and build useful statistics on network usage, bandwidth availability, link characteristics, and so on. Collected statistics could then be shared with other sensing nodes, making it possible to build a comprehensive view of the network from which to detect fluctuations in link quality and available bandwidth. I believe that adapting the amount of traffic sent over links depending on their present quality and available bandwidth could significantly improve the system performance and ensure

the transmission of most critical data despite of steep, sudden changes in the network resources available.

In order to further improve performances, NetProxy should complement network awareness, as provided by components like the one just described, with an increased application awareness. In this sense, an interesting research topic consists in the design of a plugin system that can automatically detect what applications are generating traffic and/or what type of data is being transferred, e.g., video, images, text, or audio, and apply application- and data-specific compression, reshaping, and resizing techniques. The combination of these two tools has the potential to increase QoS and QoE in a sensible manner.

Another extremely interesting research direction concerns the extension of ICeDiM with a mechanism that automatically tunes the forwarding and caching probability parameters to the conditions of the network in one node's proximity. I expect that such mechanism would be extremely effective to improve the dissemination process further. Moreover, adaptive dissemination strategies could be used to promote mobile offloading, for instance by favoring the routing of messages towards nodes close to Wi-Fi access points or directly connected to the network infrastructure.

Finally, another interesting research direction focuses on improving the contact prediction model I implemented in DisService by integrating systems based on different approaches, such as the Hidden Markov Model or a probabilistic model. New prediction systems could better capture particular periodic behaviors in nodes' mobility patterns or produce more/less accurate models that consume higher/lower processing power and memory. This would create a trade-off that enables applications to choose the prediction model that best adapts to the resources available on the node.

REFERENCES

[1] Global Health Observatory (GHO), “Urban population growth”, http://www.who.int/gho/urban_health/situation_trends/urban_population_growth_text/en/

[2] E. Ferro, B. Caroleo, M. Leo, M. Osella, E. Pautasso, "The Role of ICT in Smart City Governance", International Conference for E-Democracy and Open Government, Krems, Austria, 2013

[3] H. Schaffers, N. Komninos, M. Pallot, B. Trousse, M. Nilsson, A. Oliveira, “Smart Cities and the Future Internet: Towards Cooperation Frameworks for Open Innovation”, The Future Internet, Lecture Notes in Computer Science, Vol. 6656, pp. 431-446, 2011

[4] A. Asadi, Q. Wang, V. Mancuso, "A Survey on Device-to-Device Communication in Cellular Networks", IEEE Communications Surveys & Tutorials, Vol. 16, No. 4, pp.1801-1819, April 2014

[5] Y. Sambo, M. Shakir, F. Héliot, M. Imran, S. Mumtaz, K. Qaraqe, “Device-to-Device Communication in Heterogeneous Networks”, in S. Mumtaz, J. Rodriguez (Eds.) “Smart Device to Smart Device Communication”, Springer, pp. 219-235, 2014

[6] G. Benincasa, A. Morelli, C. Stefanelli, N. Suri, M. Tortonesi, "Agile Communication Middleware for Next-generation Mobile Heterogeneous Networks", IEEE Software, Vol. 31, No. 2 (Special Issue on Next Generation Mobile Computing), pp. 54-61, March-April 2014

[7] D: S. Alberts, J. J. Garstka, F. P. Stein, “Network Centric Warfare: Developing and Leveraging Information Superiority”, Command and Control Research Program (CCRP), US DoD, 2000

[8] 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

[9] N. Suri, “Dynamic Service-oriented Architectures for Tactical Edge Networks”, Workshop on Emerging Web Services Technology, pp. 3-10, 2009

[10] 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

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

[12] J.R. Agre, K.D. Gordon, M.S. Vassiliou, “Commercial Technology at the Tactical Edge”, International Command and Control Research and Technology Symposium (ICCRTS 2013), June 2013

[13] A. Morelli, R. Lenzi, C. Stefanelli, N. Suri, M. Tortonesi, "A Proxy Gateway Solution to Provide QoS in Tactical Networks and Disaster Recovery Scenarios", 11th ACM International Symposium on QoS and Security for Wireless and Mobile Networks (ACM Q2SWinet 2015) – ACM MSWiM 2015, Cancún, Mexico, November 2015

[14] A. Martìn-Campillo, J. Crowcroft, E. Yoneki, R. Martí, “Evaluating opportunistic networks in disaster scenarios”, Journal of Network and Computer Applications, November 2012

[15] D. Karamshuk, C. Boldrini, M. Conti, A. Passarella, “Human mobility models for opportunistic networks”, IEEE Communications Magazine, Vol. 49, No. 12, pp. 157-165, December 2011

[16] A. Mtibaa, K. Harras, “CAF: Community aware framework for large scale mobile opportunistic networks”, Computer Communications, Vol. 36, No. 2, pp. 180-190, 2013

[17] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, B. Ohlman, “A survey of information-centric networking”, IEEE Communications Magazine, Vol. 50, No. 7, pp. 26-36, July 2012

[18] G. Carofiglio, G. Morabito, L. Muscariello, I. Solis, M. Varvello. “From content delivery today to information centric networking”, Computer Networks: The International Journal of Computer and Telecommunications Networking, Vol. 57, No. 16, pp. 3116-3127, November 2013

[19] 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 (Special Issue on Military Communications), pp. 66-73, October 2013

[20] N. Suri, A. Morelli, J. Kovach, L. Sadler, R. Winkler, "Agile Computing Middleware Support for Service-oriented Computing over Tactical Networks", International Workshop on Service-Oriented Computing in Disconnected, Intermittent and Limited (DIL) Networks - VTC 2015 Spring, May 2015

[21] Barry McSweeney, “ICT Supporting the Smart Economy: The Case of Ireland“, The Government Report, “Technology Actions to Support the Smart Economy”, Chapter 2.2, pp. 141-151, 2009

[22] Cisco Systems, “Cisco Connected Roadways Drives Safety, Efficiency, Mobility, and Sustainability”, White Paper, 2015, available online at: http://www.cisco.com/c/en/us/solutions/collateral/industry-solutions/solution-overview-c22-733883.pdf

[23] N. Mitton, s. Papavassiliou, A. Puliafito, K.S. Trivedi, “Combining Cloud and sensors in a smart city environment”, EURASIP Journal on Wireless Communications and Networking, August 2012

[24] N. Leavitt, “Network-Usage Changes Push Internet Traffic to the Edge”, Computer Journal, Vol. 43, No. 10, pp. 13-15, October 2010

[25] S. Singh, H.S. Dhillon, J.G. Andrews, "Offloading in Heterogeneous Networks: Modeling, Analysis, and Design Insights", IEEE Transactions on Wireless Communications, Vol. 12, No. 5, pp. 2484-2497, May 2013

[26] K. Wei, G. Mao, W. Zhang, Y. Yang, Z. Lin, C. S. Chen, “Optimal Microcell Deployment for Effective Mobile Device Energy Saving in Heterogeneous Networks”, IEEE International Conference on Communications (ICC 2014), Sydney, Australia, June 2014

[27] J. T. J. Penttinen, “The Telecommunications Handbook: Engineering Guidelines for Fixed, Mobile and Satellite Systems”, John Wiley & Sons, Ltd, Ch. 25, 2015

[28] Coleago Consulting, “"Will Wi-Fi relieve congestion on cellular networks?", Report prepared for GSMA, 2014, available online at: http://www.gsma.com/spectrum/wp-content/uploads/2014/05/Wi-Fi-Offload-Paper.pdf

[29] P. Ghosekar, G. Katkar, P. Ghorpade, “Mobile Ad Hoc Networking: Imperatives and Challenges”, IJCA Special Issue on MANETs, Vol. 3, No. 9, pp. 153–158, 2010

[30] X. Chen, H. Zhai, J. Wang, Y. Fang, "TCP performance over mobile ad hoc networks”, in Canadian Journal of Electrical and Computer Engineering, Vol. 29, No. 1/2, pp. 129-134, 2004

[31] T.K. Sarkar, J. Zhong, K. Kyungjung, A. Medouri, M. Salazar-Palma, "A Survey of Various Propagation Models for Mobile Communication", in IEEE Antennas and Propagation Magazine, Vol. 45, No. 3, pp. 51-82, June 2003

[32] 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

[33] K. Kumar, J. Liu, Y.-H. Lu, B. Bhargava, “A Survey of Computation Offloading for Mobile Systems”, Journal of Mobile Networks and Applications, Vol. 18, No. 1, pp. 129-140, 2013

[34] J. Hoebeke, I. Moerman, B. Dhoedt, P. Demeester, “An Overview of Mobile Ad Hoc Networks: Applications and Challenges Journal of the Communications Network”, Vol. 3, pp. 60-66, 2004

[35] A. Sheth, S. Nedevschi, R. Patra, S. Surana, E. Brewer, L. Subramanian, "Packet Loss Characterization in WiFi-Based Long Distance Networks", IEEE International Conference on Computer Communications, pp. 312-320, May 2007

[36] 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, pp. 60-69, 2010

[37] S. Krishnasamy, A. Kumar, “Modeling the effect of transmission errors on TCP controlled transfers over infrastructure 802.11 wireless LANs”, ACM international

conference on Modeling, analysis and simulation of wireless and mobile systems (MSWiM '11), pp. 275-284, 2011

[38] N. Passas, S. Paskalis, A. Kaloxylos, F. Bader, R. Narcisi, E. Tsontsis, A. S. Jahan, H. Aghvami, M. O'Droma, I. Ganchev, "Enabling technologies for the ‘always best connected’concept", Wireless Communications and Mobile Computing, Vol. 6, No. 4, pp. 523-540, 2006

[39] K. Chander, D. Juneja, “A Novel Approach for Always Best Connected in Future Wireless Networks”, Global Journal of Computer Science and Technology, No. 11, Vol. 15, pp. 49–53, 2011

[40] Wireless Broadband Alliance, “From 2016 to 5G”, Industry Report, 2015, available at: http://www.wballiance.com/wba/wp-content/uploads/downloads/2015/10/WBA_FullIndustryReport_2015.pdf

[41] CISCO, “The Zettabyte Era—Trends and Analysis”, White Paper, May 2015, available online at: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.pdf

[42] S. Sharafeddine, K. Jahed, N. Abbas, E. Yaacoub, Z. Dawy, "Exploiting multiple wireless interfaces in smartphones for traffic offloading", First International Black Sea Conference on Communications and Networking (BlackSeaCom), pp. 142-146, July 2013

[43] C. Paasch, O. Bonaventure, “Multipath TCP”, Communications of the ACM, vol. 57, No. 4, pp. 51-57, April 2014

[44] P. Bellavista et al., "A Unifying Perspective on Context-Aware Evaluation and Management of Heterogeneous Wireless Connectivity", IEEE Communications Surveys & Tutorials, Vol. 13, No. 3, pp. 337-357, 2011

[45] P. Bellavista, R. Montanari, S.K. Das, “Mobile social networking middleware: a survey”, Pervasive and Mobile Computing, Vol. 9, No. 1, pp. 437-453, 2013

[46] A. Karam, N. Mohamed, “Middleware for Mobile Social Networks: A Survey”, Hawaii International Conference on System Sciences (HICSS ‘2012), pp. 1482-1490, January 2012

[47] A. Corradi, A. Landini, S. Monti, “Workflow Management and Mobile Agents: How to Get the Best of Both Approaches”, in Hershey, PA: IGI Global, “Ubiquitous Multimedia and Mobile Agents: Models and Implementations”, pp. 167-214, 2011

[48] J. Lee, Y. Yi, S. Chong, Y. Jin, "Economics of Wi-Fi Offloading: Trading Delay for Cellular Capacity", IEEE Transactions on Wireless Communications, Vol. 13, No. 3, pp. 1540-1554, 2014

[49] D. Ma, G. Tsudik, "Security and privacy in emerging wireless networks", IEEE Wireless Communications, Vol. 17, No. 5, pp. 12-21, 2010

[50] 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 (MILCOM 2009), October 2009

[51] G. Benincasa, E. Casini, R. Lenzi, A. Morelli, E. Benvegnu, N. Suri, K. Boner, S. Watson, "Extending Service-Oriented Architectures to the Tactical Edge", IEEE Military Communications Conference (MILCOM 2012), pp. 1-7, October-November 2012

[52] R. Lenzi, G. Benincasa, E. Casini, N. Suri, A. Morelli, S. Watson, J. Nevitt, "Interconnecting Tactical Service-Oriented Infrastructures with Federation Services", IEEE Military Communications Conference (MILCOM 2013), pp. 692-697, November 2013

[53] A. De Vendictis, F. Vacirca, A. Baiocchi, "Experimental Analysis of TCP and UDP Traffic Performance over Infra-structured 802.11b WLANs", European Wireless Conference 2005 - Next Generation Wireless and Mobile Communications and Services (European Wireless), pp.1-7, April 2005

[54] J. Gettys, "Bufferbloat: Dark Buffers in the Internet", in IEEE Internet Computing, Vol. 15, No. 3, pp. 96-111, May-June 2011

[55] H. Geng, N. Jamali, “Supporting many-to-many communication”, 2013 Workshop on Programming based on Actors, Agents, and Decentralized Control (AGERE! 2013), pp. 81-86, 2013

[56] A. M. Vegni, C. Campolo, A. Molinaro, T. D.C. Little, “Modeling of Intermittent Connectivity in Opportunistic Networks: The Case of Vehicular Ad hoc Networks”, Routing in Opportunistic Networks, Springer-Verlag, pp. 179-207, 2013

[57] M. Conti, S. Giordano, M. May, A. Passarella, “From opportunistic networks to opportunistic computing”, IEEE Communications Magazine, Vol. 48, No. 9, pp. 126-139, September 2010

[58] C. Boldrini, M. Conti, A. Passarella, “Exploiting users social relations to forward data in opportunistic networks: the HiBOp solution”, Pervasive and Mobile Computing, 2008

[59] C. Boldrini, M. Conti, A. Passarella, “Design and performance evaluation of ContentPlace, a social-aware data dissemination system for opportunistic networks”, Computer Networks, Vol. 54, No.4, pp. 589-604, 2010

[60] B. Han, P. Hui, V.S.A. Kumar, M.V. Marathe, J. Shao, A. Srinivasan, "Mobile Data Offloading through Opportunistic Communications and Social Participation”, IEEE Transactions on Mobile Computing, Vol. 11, No. 5, pp. 821-834, May 2012

[61] K. Lee, J. Lee, Y. Yi, I. Rhee, S. Chong, “Mobile Data Offloading: How Much Can Wi-Fi Deliver?”, IEEE/ACM Transactions on Networking, Vol. 21, No. 2, pp. 536-550, 2013

[62] L. Valerio, F. Ben Abdesslemy, A. Lindgreny, R. Bruno, A. Passarella, M. Luoto, "Offloading cellular traffic with opportunistic networks: a feasibility study", Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET 2015), pp. 1-8, June 2015

[63] C.M. Huang, K.C. Lan, C.Z. Tsai, “A Survey of Opportunistic Networks”, International Conference on Advanced Information Networking and Applications - Workshops (AINAW '08), pp. 1672-1677, 2008

[64] H. A. Nguyen, S. Giordano, “Routing in Opportunistic Networks”, International Journal of Ambient Computing and Intelligence (IJACI), Vol. 1, No. 3, 2009

[65] A. Vahdat, D. Becker, “Epidemic routing for partially connected ad hoc networks”, Technical Report CS-200006, Duke University, 2000

[66] T. Spyropoulos, K. Psounis, C. S. Raghavendra, “Spray and wait: an efficient routing scheme for intermittently connected mobile networks” ACM SIGCOMM workshop on Delay-tolerant networking (WDTN '05), pp. 252-259, 2005

[67] A. Lindgren, A. Doria, O. Schelén, “Probabilistic routing in intermittently connected

networks”, ACM SIGMOBILE Mobile Computing and Communications Review, Vol. 7, No. 3, pp. 19-20, July 2003

[68] J. Burgess, B. Gallagher, D. Jensen, B. N. Levine, “MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networking”, IEEE INFOCOM 2005, Annual Joint Conference of the IEEE Computer and Communications Societies, March 2006

[69] T. Spyropoulos, K. Psounis, C. S. Raghavendra, “Spray and Focus: Efficient Mobility-Assisted Routing for Heterogeneous and Correlated Mobility”, IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW '07), pp. 79-85, March 2007

[70] C. Boldrini, M. Conti, J. Jacopini, A. Passarella, "HiBOp: a History Based Routing Protocol for Opportunistic Networks", IEEE International Symposium on a World of Wireless, Mobile, and Multimedia Networks (WOWMOM 2007), pp.1-12, 2007

[71] H. A. Nguyen, S. Giordano, “Spatiotemporal routing algorithm in opportunistic networks”, IEEE International Symposium on a World of Wireless, Mobile, and Multimedia Networks (WoWMoM 2008), pp. 1-6, 2008

[72] C. Boldrini, M. Conti, A. Passarella, “Impact of social mobility on routing protocols for Opportunistic Networks”, IEEE International Symposium on a World of Wireless, Mobile, and Multimedia Networks (WoWMoM 2007), pp. 1-6, 2007

[73] M. Amadeo, C. Campolo, A. Molinaro, “Forwarding strategies in named data wireless ad hoc networks: Design and evaluation”, Journal of Network and Computer Applications, Vol. 50, pp. 148-158, April 2015

[74] A. Barzan, B. Bonne, P. Quax, W. Lamotte, M. Versichele, N. V. d. Weghe, "A comparative simulation of opportunistic routing protocols using realistic mobility data obtained from mass events", International Symposium on a World of Wireless, Mobile, and Multimedia Networks (WoWMoM 2013), pp. 1-6, 2013

[75] M. Varvello, I. Rimac, U. Lee, L. Greenwald, V. Hilt, “On the Design of Content-Centric MANETs”, international conference on Wireless On-Demand Network Systems and Services (WONS), January 26-28, 2011

[76] P. Mendes, “Combining data naming and context awareness for pervasive networks”,

Journal of Network and Computer Applications, Vol. 50, pp. 114-125, April 2015

[77] S. Y. Oh, D. Lau, M. Gerla, "Content Centric Networking in Tactical and Emergency MANETs", IFIP Wireless Days (WD 2010), pp. 1-5, Venice, Italy, 2010

[78] Yongqiang Huang, Hector Garcia-Molina, "Publish/subscribe in a mobile environment", Wireless Networks, Vol. 10, no. 6, pp. 643-652, November 2004

[79] G. Tyson, N. Sastry, I. Rimac, R. Cuevas, A. Mauthe, "A survey of mobility in information-centric networks: challenges and research directions", ACM workshop on Emerging Name-Oriented Mobile Networking Design - Architecture, Algorithms, and Applications (NoM '12), pp. 1-6, 2012

[80] M. Amadeo, C. Campolo, A. Molinaro, G. Ruggeri, "Content-centric wireless networking: A survey", Computer Networks, Vol. 72, pp. 1-13, October 2014

[81] S. Eum, Y. Shoji, M. Murata, N. Nishinaga, "Design and implementation of ICN-enabled IEEE 802.11 access points as nano data centers", Journal of Network and Computer Applications, Vol. 50, pp. 159-167, April 2015

[82] D. Hughes, G. Coulson, J. Walkerdine, "A Survey of Peer-to-Peer Architectures for Service Oriented Computing", Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications, IGI Global, pp. 1-19, 2010

[83] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, G. C. Polyzos, "A Survey of Information-Centric Networking Research", IEEE Communications Surveys & Tutorials, Vol. 16, No. 2, pp. 1024-1049, July 2013

[84] Y. Kim, I. Yeom, "Performance analysis of in-network caching for content-centric networking", Computer Networks, Vol. 57, No. 13, pp. 2465-2482, September 2013

[85] Y. Li, H. Xie, Y. Wen, Z. Zhang, "Coordinating In-Network Caching in Content-Centric Networks: Model and Analysis", International Conference on Distributed Computing Systems (ICDCS '13), pp. 62-72, 2013

[86] Y. Wang, Z. Li, G. Tyson, S. Uhlig, G. Xie, "Optimal Cache Allocation for Content-Centric Networking", IEEE International Conference on Network Protocols (ICNP), 2013

[87] G. Zhang, Y. Li, T. Lin, “Caching in information centric networking: A survey”, Computer Networks: The International Journal of Computer and Telecommunications Networking, Vol. 57, No. 16, pp. 3128-3141, November 2013

[88] A. Dabirmoghaddam, M. Mirzazad-Barijough, J. J. Garcia-Luna-Aceves, “Understanding Optimal Caching and Opportunistic Caching at “The Edge” of Information-Centric Networks”, International Conference on Information-centric Networking (ICN 2014), 2014

[89] D. Kim, S.-W. Lee, Y.-B. Ko, J.-H. Kim, “Cache capacity-aware content centric networking under flash crowds”, Journal of Network and Computer Applications, Vol. 50, pp. 101-113, April 2015

[90] K. Suksomboon, S. Tarnoi, Y. Ji, M. Koibuchi, K. Fukuda, S. Abe, M. Nakamura, M. Aoki, S. Urushidani, and S. Yamada, "PopCache: Cache more or less based on content popularity for information-centric networking", IEEE Conference on Local Computer Networks (LCN 2013), pp. 236-243, 2013

[91] I. Psaras, W. K. Chai, G. Pavlou, “Probabilistic In-Network Caching for Information-Centric Networks”, ACM SIGCOMM Workshop on Information-Centric Networking (ICN 2012), pp. 55-60, August 2012

[92] M. Draxler, H. Karl, "Efficiency of On-Path and Off-Path Caching Strategies in Information Centric Networks", IEEE International Conference on Green Computing and Communications (GreenCom) 2012, pp. 581-587, November 2012

[93] M. Dehghan, A. Seetharam, T. He, T. Salonidis, J. Kurose, D. Towsley, “Optimal Caching and Routing in Hybrid Networks", IEEE Military Communications Conference (MILCOM 2014), October 2014

[94] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, J. Wilcox., “Information-centric Networking: Seeing the Forest for the Trees”, ACM Workshop on Hot Topics in Networks (HotNets-X), No. 1, 2011

[95] E. Monticelli, M. Arumaithurai, I. Psaras, X. Fu, K. K. Ramakrishnan, “Combining Opportunistic and Information Centric Networks in Real World Applications”, IEEE/KuVS NetSys 2015 PhD Forum, March 2015

[96] E. Benvegnù, N. Suri, J. Hanna, V. Combs, R. Winkler, J. Kovach, “Improving Timeliness and Reliability of Data Delivery in Tactical Wireless Environments with Mockets Communications Library”, IEEE Military Communications Conference (MILCOM 2009), October 2009

[97] M. Breedy, P. Budulas, A. Morelli, N. Suri, "Transport protocols revisited”, IEEE Military Communications Conference (MILCOM 2015), pp. 1354-1360, October 2015

[98] M.M.U. Rathore, A. Ahmad, “A survey of vertical handover techniques based on IEEE 802.21 : media independent handover standard”, Telecommunications review, SK Telecom, Vol. 25, No. 2, pp. 308-324, 2015

[99] A. Morelli, C. Stefanelli, N. Suri, M. Tortonesi, “Mobility Pattern Prediction to Support Opportunistic Networking in Smart Cities”, International ICST Conference on MOBILe Wireless MiddleWARE (Mobilware 2013), November 2013

[100] A. Uszok, J. M. Bradshaw, M. Johnson, R. Jeffers, A. Tate, J. Dalton, S. Aitken, ”KAoS Policy Management for Semantic Web Services”, IEEE Intelligent Systems, Vol. 19, No. 4, pp. 32-41, July 2004

[101] A. Gladisch, R. Daher, D: Tavangarian, “Survey on Mobility and Multihoming in Future Internet”, Wireless Personal Communications, Vol. 74, No. 1, pp. 45-81, October 2012

[102] K.-C. Leung, V.O.K. Li, "Transmission control protocol (TCP) in wireless networks: issues, approaches, and challenges”, IEEE Communications Surveys & Tutorials, Vol. 8, No. 4, pp. 64-79, 2006

[103] B. Chen, I. Marsic, R. Miller, "Issues and Improvements in TCP Performance over Multihop Wireless Networks”, IEEE Sarnoff Symposium, pp.1-5, April 2008

[104] P. Dalal, N. Kothari, K. S. Dasgupta, “Improving TCP Performance over Wireless Network with Frequent Disconnections”, International Journal of Computer Networks & Communications, Vol. 3, No. 6, p. 169-184, November 2011

[105] S. Ha, I. Rhee, L. Xu, “CUBIC: a new TCP-friendly high-speed TCP variant”, ACM SIGOPS Operating Systems Review - Research and developments in the Linux kernel, Vol. 42, No. 5, pp. 64-74, July 2008

[106] R. Stewart, “Stream Control Transmission Protocol”, RFC 4960, September 2007

[107] 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

[108] Y. Gu, X. Hong, R.L. Grossman, “Experiences in Design and Implementation of a High Performance Transport Protocol”, Super Computing Conference (SC 2004), November 2004

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

[110] 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

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

[112] A. Bakre, B. Badrinath, “I-TCP: Indirect TCP for Mobile Hosts”, IEEE International Conference on Distributed Computing Systems (ICDCS '95), 1995

[113] Z. Haas, “Mobile-TCP: An Asymmetric Transport Protocol Design for Mobile Systems”, International Workshop on Mobile Multimedia Communications (IWMM ’95), 1995

[114] 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, No. 1, pp. 23-42, January-February 2001

[115] 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

[116] 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

[117] D. Perino, M. Varvello, “A Reality Check for Content Centric Networking”, Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking (ICN '11), pp. 44-49, 2011

[118] A. Vasilakos, Z. Li, G. Simon, W. You, “Information centric network: Research challenges and opportunities”, Journal of Network and Computer Applications, Vol. 52, pp. 1-10, June 2015

[119] A. Bęben, J. Mongay Batalla, W. K. Chai, J. Śliwiński, "Multi-criteria Decision Algorithms for Efficient Content Delivery in Content Networks", Annals of Telecommunications, Special Issue on Networked Digital Media, Vol. 68, No. 3-4, pp. 153-165, April 2013

[120] K. Katsaros, C. Wei, W. Ning, G. Pavlou, H. Bontius, M. Paolone, "Information-centric networking for machine-to-machine data delivery: a case study in smart grid applications”, IEEE Network, Vol. 28, No. 3, pp. 58-64, May-June 2014

[121] E. Yoneki, P. Hui, S. Chan, J. Crowcroft, “A socio-aware overlay for publish/subscribe communication in delay tolerant networks”, ACM Symposium on Modeling, analysis, and Simulation of Wireless and Mobile systems (MSWiM '07), 2007

[122] Wei Koong Chai, Diliang He, Ioannis Psaras, George Pavlou, “Cache ‘Less for More’ in Information-centric Networks (Extended Version)”, Computer Communications, Vol. 36, No. 7, pp. 758-770, April 2013

[123] A. Detti, D. Tassetto, N. Blefari-Melazzi, F. Fedi, “Exploiting Content Centric Networking to Develop Topic-based, Publish-Subscribe MANET Systems”, Elsevier Ad Hoc Networks, Vol. 24, Part B, pp. 115-133, January 2015

[124] S. Wood, J. Mathewson, J. Joy, M.-O. Stehr, M. Kim, A. Gehani, M. Gerla, H. Sadjadpour, JJ. Garcia-Luna-Aceves, "ICEMAN: A Practical Architecture for Situational Awareness at the Network Edge”, Logic, Rewriting, and Concurrency, Springer International Publishing, pp. 617-631, 2015

[125] S. Baseer, M.I Channa, K. Ahmed, “A Review of Routing Protocols of Heterogeneous Networks”, International Journal of Computer Applications (IJCA), Vol. 2, No. 2, pp. 58-66, 2010

[126] N. Bin, N. Santhapuri, Z. Zifei, S. Nelakuditi, "Routing with Opportunistically Coded Exchanges in Wireless Mesh Networks", IEEE Workshop on Wireless Mesh Networks - WiMesh 2006, pp. 157-159, September 2006

[127] A. Boukerche, A. Darehshoorzadeh, “Opportunistic Routing in Wireless Networks: Models, Algorithms, and Classifications”, ACM Computer Surveys, Vol. 47, No. 2, Article 22, pp. 1-36, November 2014

[128] A. Detti, B. Ricci, N. Blefari-Melazzi, “Mobile peer-to-peer video streaming over information-centric networks”, Computer Networks, Vol. 81, pp. 272-288, April 2015

[129] L. Militano, M. Condoluci, G. Araniti, A. Molinaro, A. Iera, F.H.P. Fitzek, "Wi-Fi cooperation or D2D-based multicast content distribution in LTE-A: A comparative analysis", IEEE International Conference on Communications Workshops (ICC), pp.296-301, June 2014

[130] L. Hogie, P. Bouvry, F. Guinand, “An Overview of MANETs Simulation”, Journal of Electronic Notes in Theoretical Computer Science (ENTCS), Vol. 150, No. 1, pp. 81-101, March 2006

[131] M. Liu, Y. Yang, Z. Qin, “A survey of routing protocols and simulations in delay-tolerant networks”, International Conference on Wireless Algorithms, Systems, and Applications (WASA '11), pp. 243-253, 2011

[132] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, H. Zedan, “A comprehensive survey on vehicular Ad Hoc network”, Journal of Network and Computer Applications, Vol. 37, pp. 380-392, January 2014

[133] A. Keränen, J. Ott, T. Kärkkäinen, “The ONE Simulator for DTN Protocol Evaluation”, International Conference on Simulation Tools and Techniques (SIMUTools’09), Rome, No. 55, pp. 1-10, March 2009

[134] A. Huang, C. Lea, A. K. Wong, “A Joint Solution for the Hidden and Exposed Terminal Problems in CSMA/CA Wireless Networks”, Computer Networks: The International Journal

of Computer and Telecommunications Networking, Vol. 56, No. 14, pp. 3261-3273, September 2012

[135] J.C.-P. Wang, M. Abolhasan, D. R. Franklin, F. Safaei, "Characterising the Behaviour of IEEE 802.11 Broadcast Transmissions in Ad Hoc Wireless LANs", IEEE International Conference on Communications (ICC '09), pp.1-5, June 2009

[136] I. Augé-Blum, K. Boussetta, H. Rivano, R.Stanica, F. Valois, “Capillary Networks: A Novel Networking Paradigm for Urban Environments”, Workshop on Urban networking (UrbaNe '12), pp. 25-30, 2012

[137] M. Marchini, M. Tortonesi, G. Benincasa, N. Suri, C. Stefanelli, "Predicting Peer Interactions for Opportunistic Information Dissemination Protocols", IEEE Symposium on Computers and Communication (ISCC 2012), Cappadocia, Turkey, July 2012

[138] 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 (MILCOM 2011), pp. 1586-1591, November 2011

[139] A.S. Peng, D.M. Moen, Tian He; D.J. Lilja, "Automatic Dynamic Resource Management Architecture in Tactical Network Environments", IEEE Military Communications Conference (MILCOM 2009), pp. 1-7, October 2009

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

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

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

[143] 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 (MILCOM 2008), pp. 1-7, November 2008

[144] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, I. Stoica, “A Data-oriented (and Beyond) Network Architecture”, ACM SIGCOMM Computer Communication Review, Vol. 37, No. 4, pp. 181-192, October 2007

[145] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, R. L. Braynard, “Networking Named Content”, International Conference on Emerging Networking Experiments and Technologies (CoNEXT '09), pp. 1-12, 2009

[146] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. C. Claffy, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang, “Named Data Networking”, ACM SIGCOMM Computer Communication Review, Vol. 44, No. 3, pp. 66-73, July 2014

[147] B. Ahlgren, M. D'Ambrosio, M. Marchisio, I. Marsh, C. Dannewitz, B. Ohlman, K. Pentikousis, O. Strandberg, R. Rembarz, V. Vercellone, “Design Considerations for a Network of Information”, ACM Conference on emerging Networking EXperiments and Technologies (CoNEXT '08), No. 66, pp. 1-6, 2008

[148] N. Fotiou, D. Trossen, G. C. Polyzos, “Illustrating a Publish-Subscribe Internet Architecture”, Journal of Telecommunications Systems, Vol. 51, No. 4, pp. 233-245, December 2012

[149] W. Moreira, P. Mendes, "Social-aware Forwarding in Opportunistic Wireless Networks: Content Awareness or Obliviousness?", International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2014), Sydney, pp. 1-6, 2014

[150] M. Amadeo, A. Molinaro, "CHANET: A content-centric architecture for IEEE 802.11 MANETs", International Conference on the Network of the Future (NOF 2011), pp.122-127, Paris, November 2011

[151] S. Yang, F. Zhong, C. K. Yeo, B. S. Lee, J. Boleng, "Position Based Opportunistic Routing for Robust Data Delivery in MANETs", IEEE Global Telecommunications Conference (GLOBECOM 2009), pp. 1-6, 2009

[152] S. K. A. Khan, J. Loo, M. A. Azam, H. Sardar, M. Adeel, "LOC: Location-aware Opportunistic Content Forwarding Using Direction Vectors", IEEE Symposium on Computers & Informatics (ISCI), pp. 184-189, 2013

[153] P. Hui, J. Crowcroft, E. Yoneki, "BUBBLE Rap: Social-Based Forwarding in Delay-Tolerant Networks", IEEE Transactions on Mobile Computing, Vol. 10, No. 11, pp. 1576-1589, November 2011

[154] S. K. A. Khan, R. J. Mondragon, L. N. Tokarchuk, "Lobby Influence: Opportunistic Forwarding Algorithm Based on Human Social Relationship Patterns", IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 211-216, 2012

[155] C. Song, Z. Qu, N. Blumm, A.-L. Barabási, “Limits of Predictability in Human Mobility”, Science, Vol. 327, No. 5968, pp. 1018-1021, February 2010

[156] P.-C. Cheng, K. C. Lee, M. Gerla, J. Härri, “GeoDTN+Nav: Geographic DTN Routing with Navigator Prediction for Urban Vehicular Environments”, Mobile Networks and Applications, Vol. 15, No. 1, pp. 61-82, February 2010

[157] B. Karp, H. T. Kung, “GPSR: Greedy Perimeter Stateless Routing for Wireless Networks”, International Conference on Mobile Computing and Networking (MobiCom '00), pp. 243-254, 2000

[158] Kevin C. Lee, Pei-chun Cheng , Jui-ting Weng , Lung-chih Tung , Mario Gerla, “VCLCR: A practical geographic routing protocol in urban scenarios”, Technical Report 080009, UCLA, Los Angeles, CA, USA, March 2008

[159] M. A. Bayir, M. Demirbas, "PRO: A Profile-Based Routing Protocol for Pocket Switched Networks", IEEE Global Telecommunications Conference (GLOBECOM 2010), Miami, FL, pp. 1-5, 2010

[160] Y. Li, M. Xu, Q. Liu, J. Yu, “HMPR: Forwarding Based on History Meeting Prediction Routing in Opportunistic Networks”, Lecture Notes in Computer Science, Wireless Algorithms, Systems, and Applications (WASA), Vol. 7405, pp. 584-594, 2012

[161] J. Niu, J. Guo, Q. Cai, N. Sadeh, S. Guo, "Predict and spread: an Efficient Routing Algorithm for Opportunistic Networking”, IEEE Wireless Communications and Networking Conference (WCNC), pp. 498-503, Cancún, Mexico, 2011

[162] J. Ghosh, S. J. Philip, C. Qiao, "Sociological Orbit Aware Location Approximation and Routing (SOLAR) in DTN", Technical report, State University of New York at Buffalo, April 2005