Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 8186310, 19 pages
http://dx.doi.org/10.1155/2016/8186310

Research Article

Quality of Service Specifications in Small-Scale Proximity-Aware Mobile Sensor Sharing Frameworks

Pramita Mitra,[1] Mehdi Golestanian,[2] and Christian Poellabauer[2]

[1]*Ford Research and Innovation Center, Dearborn, MI 48121, USA*
[2]*Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA*

Correspondence should be addressed to Pramita Mitra; pmitra3@ford.com

Received 9 September 2015; Revised 11 February 2016; Accepted 15 February 2016

Academic Editor: Daniele Riboni

Proximity-awareness, that is, a mobile device being aware of the presence and capabilities of other devices in its proximity, can be beneficial in many frameworks that support spontaneous sharing of sensors available in advanced personal mobile devices (e.g., smartphones and tablets). Providing Quality of Service (QoS) guarantees in such frameworks is highly challenging because of the dynamic and loosely coupled nature of Mobile Ad Hoc Networks (MANETs). A framework called the SPontaneous Information and Resource sharing InfrasTructure (SPIRIT) has been presented before to address the challenges of proximity-aware mobile sensor sharing. SPIRIT allows mobile applications to share sensors as services in an automatic fashion by enabling the service providers and clients to express a variety of QoS specifications. This paper presents a set of mobility-aware QoS mechanisms for enabling the implementation of QoS specifications along the multihop paths between service providers and clients in highly mobile environments. Simulations in small-scale Mobile Ad Hoc Networks show SPIRIT's ability to effectively control and manage traffic flows while maintaining desired QoS. The implications of the proposed QoS mechanisms extend beyond the scope of SPIRIT, as QoS provisioning is an important issue in many MANET frameworks and applications including Vehicular Ad Hoc Networks (VANETs).

1. Motivation

The huge growth in the number of advanced personal mobile devices such as smartphones and tablets equipped with sensors (e.g., location, speed, acceleration, image, light, pressure, and sound) presents the potential of building powerful distributed sensing and resource sharing systems. However, most mobile devices operate in isolation from one another; that is, they are not aware of the presence and capabilities of other devices in their proximity. There are numerous situations when proximity-awareness could benefit a mobile device, for example, allowing them to "borrow" sensor data or other resources (e.g., processing, network, and storage) from nearby peers. This is particularly useful in situations where mobile devices form a peer-to-peer Mobile Ad Hoc Network (MANET) to pursue a common goal (e.g., providing emergency response, collecting and analyzing large datasets during a scientific field trip) by sharing their resources in a spontaneous fashion. Specifically, emergency response has received increasing attention; for example, there is a growing recognition by governments and private institutions that MANET-based emergency response (ER) systems could prove to be highly beneficial to minimizing the fatalities of human lives after a natural disaster occurs [1–3]. One of the major in-field ER tasks is vital signs monitoring of multiple patients. Figure 1 shows an example of a proximity-aware mobile healthcare app, called BREathing rate MONitoring (BREMON) [4], which enables smartphone-based monitoring of the breathing activities of multiple patients at once at a disaster site. In BREMON, paramedics place a smartphone on the upper abdominal region (below the rib cage and on top of the thoracic diaphragm) of a patient who is lying on the ground. The acceleration data collected on the smartphone during the patient's inhaling and exhaling activities is processed to calculate the number of Breaths Per Minute (BPM). The latest BPM data is periodically relayed to the smartphones used by all the nearby paramedics over a multihop MANET. If there is any sudden change in the

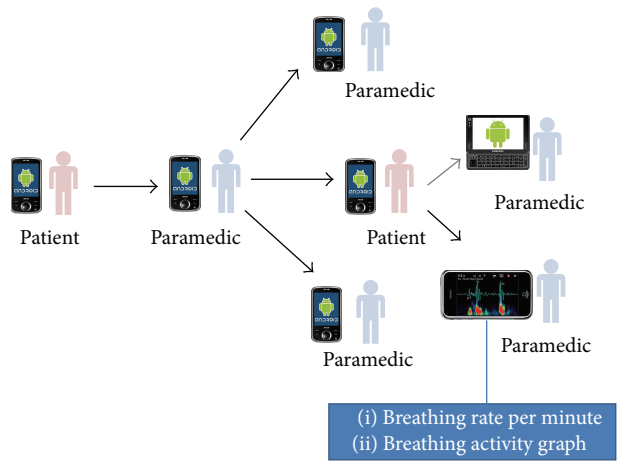

FIGURE 1: Smartphone-based monitoring of multiple patients.

BPM of a patient, the system alerts the paramedics on their smartphones. Thus paramedics can monitor and care for several patients at once, yet they still become alert to sudden changes in any particular patient's physiological status.

Due to the dynamic nature of MANETs, an infrastructure is required that facilitates the processes of finding other devices, sharing sensor data among such devices, and maintaining network connectivity in mobile environments. The sensor resources shared by such proximity-aware applications pertain to services offered by the infrastructure. The service providers and clients interact by exchanging messages over the multihop wireless medium. A spontaneous service sharing environment, called SPontaneous Information and Resource sharing InfrasTructure (SPIRIT) has been previously introduced in [5, 6]. SPIRIT enables mobile devices to share sensor resources as services in an automatic and efficient fashion. It allows service providers to specify constraints (e.g., geographic area, duration, CPU load, and residual power) for sharing services: similarly, the clients are allowed to specify Quality of Service (QoS) requirements (e.g., reliable and persistent data delivery) for services they are seeking. SPIRIT adapts the (m, k) firm guarantee model [7] for mobile scenarios and utilizes it to enable the clients to specify their QoS requirements (i.e., receiving any m out k consecutive messages is sufficient for meeting desired QoS level).

However, given the highly dynamic network topology of MANETs, supporting QoS in any form is extremely difficult in such environments. Some of the top challenges are as follows. (1) *Transient path lifetime*: as node mobility causes new routes to form and old routes to break frequently, it is highly unlikely that one particular multihop, end-to-end data delivery path would exist for the entirety of desired service duration. Hence the notion of providing end-to-end QoS guarantee for the entire service lifetime is not applicable. (2) *Resource constraints:* in SPIRIT-like frameworks, MANET nodes share resources with peers over multiple hops while working together towards a common goal. During multihop forwarding of service data messages, the resource-constrained MANET nodes (e.g., constrained by limited local resources such as battery, storage, and networking bandwidth) are incumbent to additional resource overhead.

This paper presents a set of four mobility-aware mechanisms for implementing the QoS specifications within the SPIRIT infrastructure. Specifically, these proposed QoS mechanisms account for the highly dynamic network topology, inherent unpredictability, and resource constraints of MANETs in a number of ways.

(i) *QoS Signaling*. The service lifetime is divided into multiple discrete, subsequent time intervals called subscription interval (SI): at the beginning of each SI a new, stable, multihop data delivery path that will last for the SI duration is discovered (note that the new SI path can include subpath(s) from the previous SI path). The client QoS specifications are signaled during the path discovery so that the same QoS requirements are maintained along the new SI path. Such discretization of QoS signaling allows for QoS requirements to be continuously met over the desired service duration in highly mobile environments.

(ii) *Resource Policing*. Resource policing in SPIRIT allows MANET nodes to impose and exercise local upper limits on the usage of their resources (i.e., bandwidth and storage) for services requested by other nodes.

(iii) *QoS Classification*. SPIRIT services are classified into three types in terms of service execution time, that is, INSTANTANEOUS, RESERVE-AHEAD, and CONDITIONAL. Such classification enables clients to express a variety of QoS requirements and other nodes (e.g., service providers and intermediate nodes) to reduce their processing complexity, respectively. It also enables SPIRIT to achieve some semblance of priority-based flow control in the network.

(iv) *Admission Control*. When a new multihop data delivery path is being explored during an SI, the admission control process at the potential future intermediate nodes determines whether current resource conditions are sufficient for offering the QoS requested by the service-seeking client, without any QoS violations. During admission control, parameters like available bandwidth, node availability based on predicted mobility, and storage availability are considered for making admission decisions. The process also uses a look-ahead strategy in order to prevent an INSTANTANEOUS/CONDITIONAL service being preempted when a previously admitted RESERVE-AHEAD service stream becomes active.

These four mobility-aware QoS mechanisms make it possible for QoS specifications to be effectively implemented within the framework while minimizing resource consumption in an efficient way. Simulation that utilizes a modified (m, k)-based Weighted Fair Queue (WFQ) [8] demonstrates a real-life implementation opportunity for SPIRIT; the results show SPIRIT's ability to effectively control and manage traffic flows while maintaining desired QoS. The implications of the proposed QoS mechanisms extend beyond the scope of

SPIRIT, as QoS provisioning is an important issue in many MANET frameworks and applications including VANETs.

The rest of this paper is organized as follows: Section 2 compares SPIRIT with related work, and Section 3 provides a brief overview of the SPIRIT software architecture, followed by a detailed discussion of the SPIRIT service sharing model in Section 4. Next, Section 5 describes the QoS mechanisms in detail, while Section 6 discusses simulation results and a proof-of-concept implementation of SPIRIT, and finally, Section 7 concludes the paper.

2. Related Work

Implementing QoS in mobile sensing systems is highly challenging because of the highly dynamic and loosely coupled nature of the system. There are two aspects of QoS in MANETs, that is, (1) the QoS specifications enabled by the system and (2) the QoS provisioning mechanisms.

2.1. QoS Specifications. The most popular and desired QoS specifications made in such systems with high node mobility are (1) reliable delivery of events, (2) timeliness, and (3) security. *Reliable delivery* of an event is crucial because of the nondeterministic behavior of the system. Related efforts [9, 10] propose reducing the nondeterministic behavior of the system by implementing policies such as persistence of events, durability of subscription, and event storage and retransmission. The SPIRIT infrastructure provides similar approaches to ensure reliable delivery of sensor data to mobile clients. The proximity-aware applications enabled by SPIRIT are similar to many real-time applications (e.g., multimedia and automotive control applications) in that it is acceptable to miss a few task deadlines occasionally. One of the popular solutions to the scheduling problem in real-time systems is the (m, k) firm guarantee model [7]. In this paper, the (m, k) model is incorporated into specifying the QoS requirements of the proximity-aware applications which are enabled by SPIRIT. SPIRIT further extends the (m, k) model into (m, k, l) model where the additional parameter l signifies the level of data persistence required by the client: this extended (m, k, l) model helps to increase reliability within the infrastructure as the clients can request the service messages that were lost in event of a data delivery failure.

Timeliness is another crucial issue for many real-time applications. The loosely coupled relationship between the service providers and clients makes it hard to make such a guarantee. Recent research has proposed a number of ways to provide timeliness such as using supervised machine learning [11] or packet scheduling [12] based approaches or autonomous adaptation and coordination of networking layers [13]. Timeliness is not currently supported within the SPIRIT infrastructure; however, it is possible to provide timeliness guarantees in SPIRIT by associating priorities to service messages so that most crucial messages get higher processing and transmission priority in the system.

Implementing *security* in mobile sensing systems is critical because it involves not only granting system access to authorized participants, but also enforcing trust between service providers and clients. A client may want to trust authenticity of the data/events received from the system. Since an event is in general delivered to multiple clients in these systems, the fact that the event traverses several infrastructure nodes during routing makes it tough to ensure trust relationship between the service providers and the clients. Dionysiou et al. [14] propose an idea to enforce trust between publishers and subscribers through a chain of trust relationships involving all the infrastructure nodes along the data dissemination path, while preserving the anonymity between the publishers and subscribers. However, in the most general case where one cannot assume the whole infrastructure to be trustworthy, the possibility of an event traveling through potentially malicious networks or nodes should be considered. Fiege et al. [15] propose a solution to this scenario by organizing groups of trust in scopes, that is, logical domains within the infrastructure. The organization in scopes limits the visibility of publishers, subscribers, events, and subscriptions within a single scope. The SPIRIT infrastructure is limited in that it assumes that all the participants are familiar nodes (like a team of rescue workers, colleagues, etc.) so that the intermediate nodes can be trusted not to corrupt the events, subscriptions, or some of the participants' identities.

2.2. QoS Provisioning. Provisioning QoS in MANETs is much more difficult than in most other types of networks, because the wireless bandwidth is shared by adjacent nodes and the network topology changes very frequently as the nodes move. Most routing protocols for MANETs (e.g., AODV, DSR, and TORA [16]) are designed without explicitly considering the QoS capabilities of the routes they generate. QoS routing requires finding not only a route from a source to destination(s), but also a route that satisfies the end-to-end QoS requirements requested by the client. For this reason, QoS provisioning mechanisms are tightly integrated with the standard routing protocols in MANETs [17, 18]. QoS signaling is a widely deployed approach in the Integrated Services (IntServ) architecture [19]: in this approach, end-to-end messages are signaled using the Resource reSerVation Protocol (RSVP) [20] to enable QoS configurations and reserve necessary resources on the routers along the multihop data path. Similar to the RSVP protocol, SPIRIT allows the service-seeking clients to signal the service provider and intermediate nodes about the specific QoS required in the requested service stream. On the other hand, the Differentiated Services (DiffServ) architecture [21] reduces processing complexity and signaling overhead by aggregating service flows in different QoS classes. Similar to DiffServ, the QoS reservations in SPIRIT are coarsely classified in three categories (mentioned in Section 1) and are statically configured at the service provider and intermediate nodes. As the QoS requirements in SPIRIT are signaled once for each SI, the static QoS configurations only last for a short time period, thus reducing the negative impact of node mobility in comparison to DiffServ in which the static QoS configurations last for the entire lifetime of a service. Applications similar to the class of proximity-aware applications enabled by SPIRIT also employ QoS classification for addressing the challenge of providing QoS in a dynamic,

heterogeneous environment. Such applications range from mobile healthcare applications [22, 23] to cooperative unmanned aerial vehicles (UAVs) applications [24, 25] and exhibit heterogeneous traffic characteristics ranging from low-rate, predictable traffic to bandwidth-hungry real-time applications. Typically, applications are classified into different service classes, based on their particular QoS requirements (e.g., latency constraints) or data criticality (e.g., emergency alarms versus normal data traffic). Then advanced scheduling and routing schemes are employed to provide different levels of priority to each service class.

In addition to QoS signaling and classification, admission control at the intermediate routers is a crucial aspect in QoS provisioning. Two of the highly critical parameters used by policy and admission control procedures are bandwidth and delay [26, 27]. However, additional parameters like mobility, data loss, and energy efficiency should also be considered [28]. SPIRIT uses parameters like available bandwidth, node availability based on predicted mobility, and storage availability (for data persistence) for making admission control decisions.

3. SPIRIT Architecture Overview

The architecture of SPIRIT follows a layered approach as shown in Figure 2. SPIRIT is a middleware that sits between the applications layer and the mobile operating system. As shown in Figure 2, SPIRIT consists of four layers which are discussed below.

SPIRIT API Layer. This layer provides a well-defined interface to the applications that want to share services spontaneously using the SPIRIT infrastructure. The key task of the SPIRIT API layer is to provide abstractions for the functionalities provided by the infrastructure, that is, to enable mobile devices to create, discover, subscribe, invoke, unsubscribe, and control services within the system.

SPIRIT Services Layer. This layer (illustrated by dashed lines in Figure 2) consists of three modules for implementing the service sharing functionalities provided by SPIRIT, namely, Service Description and MaTching (SDMT), Service Discovery and MaNagement (SDMN), and Quality of Service Management (QoSM). The SDMT module maps the user specifications (passed down to the infrastructure by the applications via the SPIRIT API) to abstract service specifications, using an ontology-based subscription language. This module also provides an ontology-enhanced search-and-match engine for finding the service providers that adhere to conditions requested by service-seeking clients. The SDMN module realizes the abstract ontology-based specifications into concrete messages of specific transmittable format. This module also provides an interaction protocol for implementing the communications between service providers and service-seeking clients. The QoSM module provides approaches for implementing end-to-end soft QoS within the infrastructure while handling node mobility in an efficient way. The mechanism implemented by the QoSM module is the focus of the current paper.

Courier Group Communications Layer. Courier uses the underlying unicast routing module to create an overlay routing network of service providers and subscribers. Courier addresses the challenges of group management posed by the highly dynamic and unpredictable character of mobile networks. It provides approaches for bandwidth efficient group management and adaptive data dissemination while focusing on handling mobility in an efficient way.

Peer-to-Peer Asymmetric Geographic Forwarding Layer. This bottommost layer is a unicast routing protocol called Asymmetric Geographic Forwarding (A-GF) [29], which is based on the well-known Geographic Forwarding (GF) principle. A-GF discovers asymmetric links in the network, evaluates them for stability (e.g., based on mobility), and uses them to obtain more efficient and shorter routes. A-GF also successfully identifies transient asymmetric links and ignores them to further improve the routing efficiency.

4. SPIRIT Service Sharing Model

In SPIRIT, the shared sensor resources pertain to services offered by the mobile devices. Each service is identified by a 7-tuple {sensor_name, inp_specs, out_specs, exec_type, periodic_interval, geographic_range, temporal_range}, specifying the name of the sensor resource to be shared as a service, the list of input parameters required to execute the service, the list of output objects and their specifications, the execution type of the service, the periodic time interval, the geographic area within which the service will be provided, and lastly the service lifetime in terms of the start and finish times of the service. A SPIRIT service could be of three types in terms of the time of execution: (1) Instantaneous: an Instantaneous service should begin executing as soon as the resources are available. The start time of this kind of service is defined to be zero in order to imply immediate execution; (2) Reserve-ahead: a Reserve-ahead service should be executed in future but the necessary resources should be reserved in advance; and (3) Conditional: a Conditional service should be executed whenever a particular condition is satisfied during the lifetime of service duration. The service start and finish time in this case define the absolute temporal boundary of the service execution.

4.1. QoS Specifications. QoS management on an end-to-end basis means that the applications can specify what quality they want and what cost they are willing to pay for it. QoS specifications deal with the definition of the requested QoS level and is typically offered through a specifications language or a comprehensive API. QoS could be specified along different dimensions of QoS categories such as performance, cost, or fault tolerance. In addition, a dimension can be defined as a qualitative or quantitative attribute of a category [30, 31].

In addition to the 7-tuple identification parameters described above, a SPIRIT service also includes two specifications parameters, namely, service_constraints and qos_requirements: the first one allows the service providers

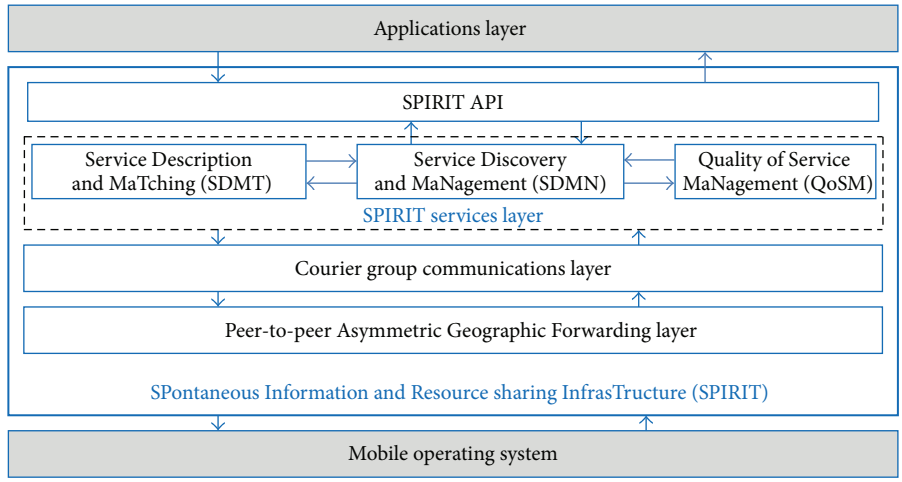

Figure 2: SPIRIT architecture.

to specify service constraints in terms of a set of logical conditions which should be met prior to the service being executed. These constraints are associated with explicit effects which happen as a result of the service execution on the physical resources of the service provider device (e.g., cpu/memory usage, residual battery level). On the other hand, the service-seeking clients are allowed to specify their QoS requirements that are suitable for such highly mobile peer-to-peer environments. The loosely coupled and asynchronous nature of a mobile sensor sharing infrastructure like SPIRIT introduces a nondeterministic behavior in the system. In addition, node mobility dictates the lifetime of a multihop path between the service provider and the client. To that end, there is no single multihop end-to-end path that exists throughout the entire service lifetime. The service update messages that are released at different time instances during the service lifetime are most likely to travel on different paths. That is why it is not feasible to provide firm end-to-end QoS guarantees within such an infrastructure: such systems are only capable of providing "soft" QoS guarantees, that is, when the firm QoS requirement is relaxed by tolerating certain degrees of fluctuation in the service quality. SPIRIT introduces a flexible soft QoS based model which is suitable for the proximity-aware applications enabled by the infrastructure, as described below.

The proximity-aware applications are similar to many real-time applications (e.g., multimedia and automotive control applications) in that it is acceptable to miss a few task deadlines occasionally. One of the popular solutions to the scheduling problem in real-time systems is the (m, k) firm guarantee model [7]. In this model, a periodic task τ is said to have an (m, k) firm guarantee requirement if it is adequate to meet the deadlines of any m out of k consecutive instances of the task where m and k are positive integers and $m \leq k$. The advantage of this guarantee model is that the applications can represent a wide range of tolerance to deadline misses by properly choosing the values of m and k. The (m, k) model is adapted by SPIRIT to specify the QoS requirements of the proximity-aware applications. To that end, it is adequate for these applications if the client-side receives any m out of k consecutive update messages from a remote service provider. In addition, the service lifetime is implicitly divided into multiple discrete, subsequent time intervals: each such interval is called a *subscription interval* (SI). These intervals are of equal length which in turn is computed from the requested (m, k) requirement. Dividing the service lifetime into discrete time intervals helps SPIRIT to distribute the single end-to-end QoS requirement into multiple subsequent end-to-end QoS requirements and thereby achieve a fine-grained control over QoS provisioning in a highly dynamic network (this will be described in detail in Section 4). This relaxation is implicit to the infrastructure: the (m, k) requirements are signaled once for each SI in order to find and reserve available resources necessary for supporting QoS in a highly dynamic environment.

On the other hand, data delivery failure (e.g., failures due to node failure, disconnection from the network) in highly mobile environments is a very common phenomenon, and hence SPIRIT allows the service-seeking clients to request the service data to be persistent. Data persistence allows clients to receive the service update messages that were lost in event of a data delivery failure. The service providers are required to log service update messages in a persistent storage. When a client that failed/was disconnected rejoins the network and the durability of its subscription is still valid, the service provider resends the stored messages to the client. The SPIRIT service model allows a client to specify the level of data persistence by extending the (m, k) requirement to (m, k, l) requirement: the additional parameter l indicates the number of SIs in the recent history for which the service provider is supposed to log all the service update messages for data persistence. Thus the SPIRIT service model provides a comprehensive description of the various aspects of service sharing in highly mobile environments.

4.2. Example Scenario. This section describes the working of the SPIRIT service sharing model, using a mobile healthcare app called BREMON (introduced in Section 1) as an example. It uses the SPIRIT infrastructure as an external library

```
<serviceProfile>
  <sensorResource>Acceleration</sensorResource>
  <serviceExecutionType>INSTANTANEOUS</serviceExecutionType>
  <serviceAttributes>
    <updateInterval> <Period>30</Period> </updateInterval>
    <geographicRange>
      <Center>
        <LAT>+40.898333</LAT> <LON>-86.453889</LON>
      </Center>
      <Radius>100</Radius>
    </geographicRange>
    <temporalRange>
      <beginsAt>12:15:00</beginsAt> <endsAt>14:15:00</endsAt>
    </temporalRange>
    <qualityOfService>
      <m>3</m> <k>5</k> <l>4</l>
    </qualityOfService>
  </serviceAttributes>
</serviceProfile>
```

Algorithm 1: BREMON client requirements.

that provides support for discovery and sharing of acceleration data with simple QoS requirements in a multihop MANET. The shared data pertain to services provided by the infrastructure. SPIRIT runs as a background daemon and manages all the communications between the patient and paramedic devices, which serve as the service providers and service-seeking clients, respectively. These service providers and clients interact with the SPIRIT daemon via the set of function calls defined in the SPIRIT API. The constraints and conditions specified by the service providers and clients, respectively, are embedded in the API data objects which are then passed to the daemon as arguments of the function call. Each function call prompts the daemon to invoke routines to convert the SPIRIT API data objects into ontology-based XML format that will be universally interpretable within the infrastructure. A service is described using an ontology-based data type called Service Profile (SP). An SP instance describes a service with its input and output types, capabilities, and constraints and thus provides a detailed description of the service, needed by a matchmaker software agent to perform in-depth analysis whether the service meets certain requirements. In addition to representing the capabilities of a service, an SP instance is also used to express the QoS requirements of a service-seeking client, so that a matchmaker software agent has a convenient dual-purpose representation upon which to base its operations.

Algorithm 1 shows an example of BREMON client requirements. In this example, a paramedic named Bob is looking for shared acceleration data services within hundred meters of his current location. Bob's interest in finding such shared services will last for the next two hours. The service execution should start instantaneously and a new service update message (containing the latest breathing activity data) is expected every 30 seconds. The QoS requirement of the desired service is $(m, k, l) = (3, 5, 4)$, which means any 3 out of 5 consecutive service update messages must be received at the client node and the service provider should log all service update messages for the latest 4 subscription intervals.

Algorithm 2 shows an example of a BREMON service description. In this example, the service provider shares its acceleration data within a radius of 300 meters around the current location. The service lifetime spans a period that starts at current time (say 12:05 pm) and ends after 30 minutes (e.g., until the time the patient would be transported to a hospital), and it executes periodically every 10 seconds. The output of the service is the current BPM of the patient. The service executes only if the condition ($cpu\_usage \leq 40\%$ && $memory\_usage \leq 50\%$ && $power\_level \geq 60\%$) is satisfied at the patient device during execution time. These abstract specifications and requirements in XML format are serialized into concrete messages of transmittable format, which are then exchanged over the wireless network between the service providers and clients.

5. SPIRIT Quality of Service Mechanisms

SPIRIT incorporates QoS concepts from the widely used Integrated Services (IntServ) [19] and Differentiated Services (DiffServ) [21] architecture models, while adapting itself to suit the dynamic and asynchronous nature of highly mobile environments. SPIRIT delivers end-to-end QoS required by the proximity-aware applications by explicitly managing the network and device resources. SPIRIT allows the service-seeking clients to signal the service provider and intermediate router nodes about the specific QoS required in the requested service stream. Once the service provider and all intermediate nodes along a multihop path accept a new service stream and reserve resources necessary for meeting its QoS requirements, the service provider can initiate the service and begin transmitting the service update messages along the multihop data delivery path. Over the service lifetime, the QoS guarantee requires being maintained in face of

```
<serviceProfile>
  <sensorResource>Acceleration</sensorResource>
  <serviceInpSpec>
    <ioSpec>
      <argvName>Location</argvName>
      <argvSpec>LAT</argvSpec>
      <argvSpec>LON</argvSpec>
      <argvSpec>TIME</argvSpec>
    </ioSpec>
  </serviceInpSpec>
  <serviceOutSpec>
    <ioSpec>
      <argvName>BreathRatePerMinute</argvName>
      <argvSpec>Hz∗60</argvSpec>
    </ioSpec>
  </serviceOutSpec>
  <serviceAttributes>
    <serviceInterval> <Period>10</Period> </serviceInterval>
    <geographicRange>
      <Center> <LAT>+41.698333</LAT> <LON>-86.233889</LON> </Center>
      <Radius>300</Radius>
    </geographicRange>
    <temporalRange>
      <beginsAt>12:05:00</beginsAt> <endsAt>12:35:00</endsAt>
    </temporalRange>
  </serviceAttributes>
  <serviceConstraints>
    <Condition> cpuUsage ≤ 40% && powerLevel ≥ 60% </Condition>
  </serviceConstraints>
</serviceProfile>
```

Algorithm 2: BREMON service description.

network topology changes. SPIRIT provides a set of mobility-aware approaches to establish and maintain QoS in highly dynamic environments. This section describes the key QoS mechanisms in detail.

5.1. QoS Signaling. SPIRIT delivers end-to-end QoS required by the proximity-aware applications by explicitly managing the network and device resources as the topology changes with time. To that end, the QoS requirements in SPIRIT are periodically signaled during the service lifetime (once for each SI) in order to find and reserve available resources in a highly dynamic network. This is called *temporal distribution of subscription*. Each of the SIs for one particular service stream is of equal length which in turn is calculated from the (m, k, l) requirement requested by the client. We denote the length of an SI as T_{SI}. Given a request for a new periodic service stream τ with periodic interval T_τ and (m, k, l) requirement, we define the following temporal parameters:

(i) $t_{\tau(s)}$: the time instance when the requested service would start and the first service message would be released by the service provider.

(ii) $t_{\tau(m)}$: the time instance when the mth service update message would be released by the service provider; $t_{\tau(m)} = t_{\tau(s)} + T_\tau \times (m - 1)$.

(iii) $t_{\tau((m+k)/2)}$: the time instance when the $(m + (k - m)/2)$th, that is, $((m+k)/2)$th, service update message would be released by the service provider; $t_{\tau((m+k)/2)} = t_{\tau(s)} + T_\tau \times ((m + k)/2 - 1)$.

(iv) $t_{\tau(k)}$: the time instance when the kth service update message would be released by the service provider; $t_{\tau(k)} = t_{\tau(s)} + T_\tau \times (k - 1)$.

In SPIRIT, T_{SI} is computed as follows:

$$T_{\mathrm{SI}} = T_\tau \times (k - 1)\,. \tag{1}$$

Note that the length of SI is a critical design decision which provides a tradeoff between performance and resource overhead: an SI which is too short will lead to higher QoS in highly mobile environments but it will also result in increased communications overhead because of frequent service discovery and QoS signaling. On the other hand, a longer SI will have reduced communications overhead but it may result in poor QoS as service update messages could get dropped/lost because of unexpected topology changes in MANETs. In SPIRIT, the length of an SI is just sufficient to meet the (m, k, l) QoS requirements requested by the client. To that end, the client's current subscription to the service is valid for a period of length $T_\tau \times (k - 1)$: in other words, the service provider and intermediate nodes will reserve

resources until time instance $t_{\tau(k)}$. On the other hand, the relationship between the value of l and T_{SI} is that the service provider is supposed to log all the service update messages from the latest l number of SIs (including the current SI). In other words, during the nth SI, denoted as SI^n, the service provider is supposed to log all the messages released during the subscription intervals SI^j, $j \in \{n-l+1, n-l+2, \ldots, n\}$.

The client-side SPIRIT daemon renews its subscription at the end of each SI in order to reserve resources for the next SI. The subscription renewal in SPIRIT takes place in the form of a new *service-path discovery*: the QoS requirement of a service-path discovery is copied from the original service discovery request which was issued by the service-seeking client application. Like the original service discovery, the service-path discovery is broadcasting in nature. However, unlike the service discovery, the service-path discovery request has a specific destination, that is, the service provider application that is sharing the service with the client application. The broadcasting nature of the service-path discovery procedure results in multiple copies of the service discovery request (and response) messages arriving at the service provider (and client) node via multiple paths: this multipath attribute increases the likelihood of successful subscription renewal in MANETs where messages are often lost/dropped. Nevertheless, if no response messages are received at the end of the service-path discovery then the client-side SPIRIT daemon reattempts the subscription renewal up to three times. If all three reattempts fail then the client application is notified that the service stream has been disrupted. Given that the service-path discovery takes some nontrivial amount of time, it is highly critical that the subscription is renewed on time for the next SI so that there is no disruption in the ongoing service stream. For this reason, the client-side SPIRIT daemon initiates the subscription renewal procedure while the current SI has not yet expired: to that end, the subscription is renewed at the time instance $t_{\tau((m+k)/2-1)}$ of the current SI. The *renewal interval* (RI) is denoted as T_{RI} and its length is defined as follows:

$$T_{RI} = T_\tau \times \left(\frac{m+k}{2} - 1 \right). \tag{2}$$

From (1) and (2), we can see that $T_{RI} \leq T_{SI}$.

5.2. Resource Policing. Multihop forwarding of periodic service streams in a spontaneous resource sharing environment like SPIRIT is entirely altruistic in that intermediate nodes incur processing and networking overhead in forwarding messages for "remote" service streams (by "remote" services we mean those services that the local node is not subscribed to). As a result, it is only natural that these intermediate nodes would want to limit the amount of resources being consumed by the remote service streams passing through that node. Resource policing is the procedure of regulating the amount of resources being consumed by the remote service streams: if a maximum limit is reached or exceeded, the policy manager denies a new service request.

To that end, SPIRIT allows each intermediate node to impose upper limits on the *Maximum Average Bandwidth for All Streams* (AB_{AS}^{max}) and *Maximum Average Bandwidth per Single Stream* (AB_{SS}^{max}). The SPIRIT policy manager drops traffic of those remote service streams for which the average bandwidth consumption is higher than the defined limits. On the other hand, a client can request multiple remote service streams each with an estimated average bandwidth that is fairly low: these requests have a higher chance of being accepted than requests with higher values of estimated bandwidth. Policing is required for ensuring not only that no remote service stream exceeds the maximum limit but also that no single client can take over the entire pool of shared bandwidth. SPIRIT allows each intermediate node to specify *Maximum Average Bandwidth per Single Client* (AB_{SC}^{max}) and drops traffic of a remote service stream if the overall average bandwidth consumption by the client associated with that remote service stream exceeds the AB_{SC}^{max} limit.

In addition to managing and policing the bandwidth consumption by remote service streams, the SPIRIT QoS Module (QoSM) also controls the amount of persistent storage consumed for offering data persistence to the periodic service streams. To that end, persistent data delivery will require the service provider to log the service update messages in their local storage. Given a service request with (m, k, l) QoS requirement, the service provider is required to store all the $k \times l$ messages during the latest l number of subscription validity intervals. However, in order to reduce the storage overhead, the service provider only logs the latest $m \times l$ messages, which is just sufficient to meet the QoS requirement. On the other hand, each intermediate node logs only the latest m number of service update messages (this is justified because the same intermediate node may not participate in forwarding the periodic service stream in its next subscription validity interval). These messages are logged in a FIFO-style virtual queue on the persistent storage and each stored message is timestamped: when a new update message is released and the queue is full, the oldest message is discarded to make room for the new one. SPIRIT allows each service provider and intermediate node to specify parameters for limiting persistent storage consumption by remote service streams in a similar fashion like the bandwidth limits. To that end, each node can specify the following parameters, namely, *Maximum Data Storage for All Streams* (DS_{AS}^{max}), *Maximum Data Storage per Single Stream* (DS_{SS}^{max}), and *Maximum Data Storage per Single Client* (DS_{SC}^{max}). The SPIRIT policy manager drops the traffic of a service stream if the overall storage consumption by the client associated with that remote service stream exceeds any of these limits. All these bandwidth and storage limit parameters are defined during setup time. In addition, SPIRIT allows the service providers/intermediate nodes to specify rules so that the values of these parameters change during runtime according to the current resource availability situation at these nodes.

5.3. QoS Classification. As previously described in Section 3, SPIRIT services could be classified into three types in terms of execution, that is, Instantaneous, Reserve-ahead, and Conditional. This classification is similar to DiffServ networks in which QoS is often classified into two categories: book-ahead (BA) requests and Instantaneous Requests

(IR). When an admitted BA request becomes active, some of the ongoing IR are dropped when the bandwidth is insufficient for supporting both IR and BA requests. Several look-ahead strategies [32–34] have been proposed in recent literature to prevent the IR from being dropped. The admission control procedure in SPIRIT (described next in Section 5.4) incorporates the look-ahead strategy proposed by [32] in order to prevent an Instantaneous/Conditional service being preempted when a previously admitted Reserve-ahead service stream becomes active. To that end, the admission control procedure looks up its Reserve-ahead services list to determine whether the available resources will be sufficient for supporting a requested service stream for the duration of an SI.

5.4. Admission Control. Admission control is a validation procedure which performs a check at an intermediate or service provider node before a new service request is accepted: the purpose of the admission control test is to determine whether current resource conditions at the local node are sufficient for offering the QoS requested by the service-seeking client, without any QoS violations. Once the request is accepted, the SPIRIT infrastructure will assure that the client is served with its requested QoS parameters. Thus admission control is critical for (1) providing the required QoS to the currently supported service streams and (2) ensuring that the locally imposed resource policies are not violated. Additionally, in SPIRIT the service messages and updates from the service provider are forwarded to the clients over a multihop path by a number of mobile intermediate nodes. However, frequent topology changes impact the availability and utilization of resources: an intermediate node may not be available for an entire SI despite having all the physical resources (e.g., bandwidth) available to meet the QoS requirements of a service stream. The admission control process should be able to take this into account so that an ongoing service stream is not dropped when the intermediate node moves out of the service region. To address this challenge, SPIRIT proposes a mobility prediction model to guide the admission control process: the purpose of this is to predict the temporal availability of a node and reject those requests which are unlikely to be satisfied. Incorporating mobility-awareness in the admission control procedure allows SPIRIT to reduce the number of interruptions experienced by the service streams and thereby also increase the effective bandwidth utilization. Furthermore, the admission control process also performs a check whether the local node has available storage for meeting the data persistence requirement of the requested service stream.

The decision process of admission control is represented by a set of test conditions that the system evaluates in order to determine whether or not to accept the request from a service-seeking client. The values of the m, k, and l parameters requested by the client impact the decision of the admission controller on each service provider and intermediate node. When a client requests a new service, the admission control process takes the request as input and calculates the amount of resources required by (1) the service streams already being supported and (2) the pending service request. If the sum of the two is not larger than the resource limits defined in Section 5.2, then the service request will be admitted to the system; otherwise, the request will be rejected. For each incoming new service request with (m, k, l) requirement, the SPIRIT admission control process evaluates the following test conditions.

Average Bandwidth-Based Admission Control. Given the current bandwidth usage condition at the intermediate node, would it be able to accept the new service request with (m, k, l) requirement without violating the QoS of ongoing and reserved-ahead streams?

SPIRIT requires each intermediate node to maintain the remote services passing through that node into two lists: (1) a list of currently active services and (2) a list of Reserve-ahead services. Before making a commitment to meet the (m, k, l) requirement requested by a client, an intermediate node is required to make sure that it will be able to provide the minimum average bandwidth required to forward any m out of k consecutive service messages from that remote service stream. We denote the required *minimum average bandwidth* of a periodic service stream τ, which is requested by client C with (m^C, k^C, l^C) requirement, as $\mathrm{AB}^{\min}_{\tau(m^C,k^C,l^C)}$. The value of $\mathrm{AB}^{\min}_{\tau(m^C,k^C,l^C)}$ is defined in (3): in this equation, T^C and I^C indicate the periodic interval and the size of the shared sensor information of the remote service stream, respectively:

$$\mathrm{AB}^{\min}_{\tau(m^C,k^C,l^C)} = \left(\frac{m^C \times T^C}{k^C \times T^C}\right) \times I^C = \frac{m^C \times I^C}{k^C}. \quad (3)$$

Let us assume an intermediate node A has u and v number of remote service streams in its currently active and Reserve-ahead services lists, respectively. Next, node A receives a request from a client C with (m^C, k^C, l^C) requirement. The admission control condition differs depending on the execution type of the requested service, as described below.

(1) Average bandwidth-based admission control for Instantaneous service: If the request is for an Instantaneous service, the average bandwidth-based admission control test performs look-up on both the currently active and Reserve-ahead services lists and checks whether the available average bandwidth is sufficient for the service request. If node A were to accept the service request, then the resource usage-based admission control test at node A would require that all the three conditions in (4), (5), and (6) are satisfied. Equation (4) states that the sum of (1) the overall minimum average bandwidth of the currently active remote services $\{\tau_1, \tau_2, \ldots, \tau_u\}$, (2) the overall minimum average bandwidth of the Reserve-ahead services that will become active during the intended lifetime of the requested service $\{\tau_j,\ 1 \le j \le v,\ t^C_{\tau(s)} \le t_{\tau_j(s)} \le t_{\tau_j(e)} \le t^C_{\tau(e)}\}$, and (3) the required minimum average bandwidth of the requested service τ^C should be less than the allocated maximum average bandwidth for all remote service streams

$$\sum_i \mathrm{AB}^{\min}_{\tau_i(m_i,k_i,l_i)} + \sum_j \mathrm{AB}^{\min}_{\tau_j(m,k,l)} + \mathrm{AB}^{\min}_{\tau(m^C,k^C,l^C)} \le \mathrm{AB}^{\max}_{AS}, \quad \text{where } i = 1, 2, \ldots, u,\ 1 \le j \le v,\ t^C_{\tau(s)} \le t_{\tau_j(s)} \le t_{\tau_j(e)} \le t^C_{\tau(e)}. \quad (4)$$

Equation (5) states that the required minimum average bandwidth of the requested service should be less than the allocated maximum average bandwidth allocated per remote service stream:

$$\begin{aligned} &\mathrm{AB}^{\min}_{\tau(m^C,k^C,l^C)} \le \mathrm{AB}^{\max}_{\mathrm{SS}}, \\ &\text{or } \frac{m^C \times I^C}{k^C} \le \mathrm{AB}^{\max}_{\mathrm{SS}}. \end{aligned} \tag{5}$$

Equation (6) states that the sum of (1) the overall minimum average bandwidth of all the currently active services $\{\tau_1^C, \tau_2^C, \ldots, \tau_w^C\}$ $(0 \le w \le u)$ subscribed by C, (2) the overall minimum average bandwidth of all the Reserve-ahead services $\{\tau_1^C, \tau_2^C, \ldots, \tau_x^C\}$ $(0 \le x \le v)$ subscribed by C, and (3) the required minimum average bandwidth of the requested service should be less than the allocated maximum average bandwidth for all the remote service streams per single client. These three conditions together ensure that the allocated bandwidth for remote service streams is efficiently utilized without any QoS policy violations:

$$\begin{aligned} &\sum_{i=1}^{w} \mathrm{AB}^{\min}_{\tau_i(m_i^C,k_i^C,l_i^C)} + \sum_{j=1}^{x} \mathrm{AB}^{\min}_{\tau_j(m_j^C,k_j^C,l_j^C)} + \mathrm{AB}^{\min}_{\tau(m^C,k^C,l^C)} \\ &\quad \le \mathrm{AB}^{\max}_{\mathrm{CS}}. \end{aligned} \tag{6}$$

(2) Average bandwidth-based admission control for Reserve-ahead service: If the request is for a Reserve-ahead service, the average bandwidth-based admission control test performs look-up on the Reserve-ahead services list and checks whether the available average bandwidth is sufficient for the service request. If node A were to accept the service request, then the average bandwidth-based admission control test at node A would require that all the three conditions in (7), (8), and (9) are satisfied. Equation (7) states that the sum of (1) the overall minimum average bandwidth of all the Reserve-ahead services $\{\tau_j,\ 1 \le j \le v\}$ that will become active during the intended lifetime of the requested service τ^C $\{t^C_{\tau(s)} \le t_{\tau_j(s)} \le t_{\tau_j(e)} \le t^C_{\tau(e)}\}$ and (2) the required minimum average bandwidth of the requested service τ^C should be less than the allocated maximum average bandwidth for all remote service streams. Equation (8) states that the required minimum average bandwidth of the requested service should be less than the allocated maximum average bandwidth allocated per remote service stream. Equation (9) states that the sum of (1) the overall minimum average bandwidth of all the Reserve-ahead services $\{\tau_1^C, \tau_2^C, \ldots, \tau_x^C\}$ $(0 \le x \le v)$ subscribed by C and (2) the required minimum average bandwidth of the requested service should be less than the allocated maximum average bandwidth for all the remote service streams per single client. These three conditions together ensure that the allocated bandwidth for remote service streams is efficiently utilized without any QoS policy violations:

$$\sum_{j} \mathrm{AB}^{\min}_{\tau_j(m,k,l)} + \mathrm{AB}^{\min}_{\tau(m^C,k^C,l^C)} \le \mathrm{AB}^{\max}_{\mathrm{AS}}, \tag{7}$$

$$\text{where } 1 \le j \le v,\ t^C_{\tau(s)} \le t_{\tau_j(s)} \le t_{\tau_j(e)} \le t^C_{\tau(e)},$$

$$\mathrm{AB}^{\min}_{\tau(m^C,k^C,l^C)} \le \mathrm{AB}^{\max}_{\mathrm{SS}}, \tag{8}$$

$$\sum_{j=1}^{x} \mathrm{AB}^{\min}_{\tau_j(m_j^C,k_j^C,l_j^C)} + \mathrm{AB}^{\min}_{\tau(m^C,k^C,l^C)} \le \mathrm{AB}^{\max}_{\mathrm{CS}}. \tag{9}$$

Node Mobility-Based Admission Control. Given the mobility of the intermediate node, would it be able to forward any m out of k consecutive service updates during service lifetime so that the new service stream meets its (m, k, l) requirement?

Before making a commitment to meet the (m, k, l) guarantee requested by a client, an intermediate node needs to make sure that it will be within the geographic boundary of the requested service until it could forward any m packets of k consecutive packets to a downstream node that takes the packets closer to the location of the client. The SPIRIT QoSM further relaxes the admission test criterion *from any m packets* to *the first* $m + p$ *packets* $(0 \le p \le k - m)$. Without this relaxed criterion, an intermediate node that would not be around to forward all k packets will reject the service request. In fact, in highly mobile environments it is extremely difficult to find a node that will be around long enough to forward all k packets. The relaxed criterion says that it is sufficient if an intermediate node could forward only the first $m + p$ packets (the parameter p adds tolerance to the packet loss phenomenon which is inherent in such environments). Thus the relaxed criterion is expected to result in the increase in the likelihood of the service request with (m, k, l) requirement being accepted at an intermediate node. Note that the prediction model used in the node mobility-based admission test is extended from the mobility prediction model used in Courier group communications algorithm proposed in [6]. To that end, this prediction model uses the location and velocity of a node to define a *Node Forwarding Zone* (NFZ) such that it covers a region capturing all possible locations of that node at a future time. For instance, the current location of node A at time t_o is denoted by loc_A. At some future time t_s $(t_s > t_o)$ it is possible to define a circular region centered at loc_A. From Figure 3, we can see that, during the time interval $\{t_o, t_s\}$, A cannot be anywhere else but inside the circular region centered at loc_A with radius $v_A \times (t_s - t_o)$ (v_A is the recorded velocity of A at time t_o). This circle is the Node Forwarding Zone of node A during the time interval $\{t_o, t_s\}$ and is denoted as $\mathrm{NFZ}_{A(t_o,t_s)}$.

Let us assume that, at time t_o, an intermediate node A receives a request for a periodic service τ with (m, k, l) requirement. The prediction model defines the following parameters for the node mobility-based admission control test (in addition to the parameters $t_{\tau(s)}$, $t_{\tau(m)}$, and $t_{\tau(k)}$ defined previously in this section):

(i) $t_{\tau(m')}$: the time instance when the m'th service update message would be released by the service provider; $t_{\tau(m')} = t_{\tau(o)} + T_\tau \times (m' - 1)$ and $m' = m + p$.

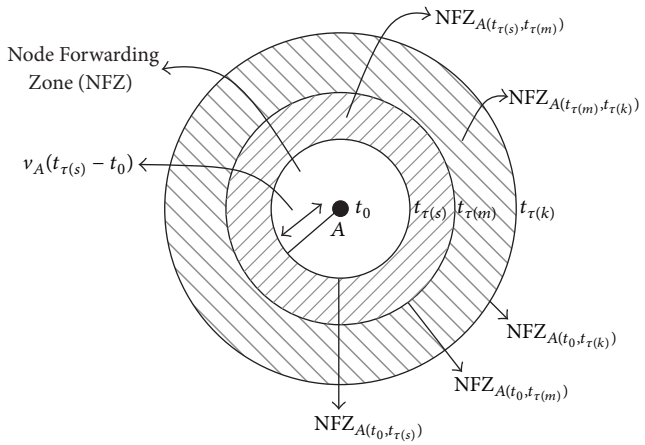

Figure 3: Node mobility prediction-based admission control in SPIRIT.

(ii) $\text{NFZ}_{A(t_o,t_{\tau(s)})}$: the circular disk that captures all possible locations of node A during the time interval $\{t_o, t_{\tau(s)}\}$.

(iii) $\text{NFZ}_{A(t_o,t_{\tau(m')})}$: the circular disk that captures all possible locations of node A during the time interval $\{t_o, t_{\tau(m')}\}$.

(iv) $\text{NFZ}_{A(t_{\tau(s)},t_{\tau(m')})}$: the circular ring that captures all possible locations of node A during the time interval $\{t_{\tau(s)}, t_{\tau(m')}\}$.

(v) $\text{NFZ}_{A(t_{\tau(m')},t_{\tau(k)})}$: the circular ring that captures all possible locations of node A during the time interval $\{t_{\tau(m')}, t_{\tau(k)}\}$.

If the service request is to be accepted with the requested (m, k, l) requirement, then the node mobility-based admission control test requires $\text{NFZ}_{A(t_{\tau(s)},t_{\tau(m')})}$ to be completely inside the geographic boundary of the requested service so that node A could forward at least the first m' packets of k consecutive packets to a downstream node that takes the packets closer to the location of the client, before node A moves out of the service region.

Data Persistence-Based Admission Control. Does the current node have sufficient storage to satisfy the data persistence requirement of the new service stream, without violating the QoS of ongoing and reserved-ahead streams?

Before making a commitment to meet the (m^C, k^C, l^C) requirement requested by a service-seeking client C, a service provider is required to make sure that its current storage situation would allow it to log the m^C number of service update messages released during each of the latest l^C number of subscription intervals (including the current interval). In other words, at the q^Cth subscription interval $T_{\text{SI}}^{q^C}$, the service provider is required to log all the latest $m^C \times l^C$ messages released during the subscription intervals T_{SI}^j, $j \in \{q^C - l^C + 1, q^C - l^C + 2, \ldots, q^C\}$. On the other hand, each intermediate node is required to log only the latest m^C number of service update messages from the current subscription interval $T_{\text{SI}}^{q^C}$. We denote the required *minimum data storage* of a periodic service stream S, which is requested by client C with (m^C, k^C, l^C) requirement, as $\text{DS}(m^C, k^C, l^C)_S^{\min}$. The value of $\text{DS}(m^C, k^C, l^C)_S^{\min}$ is defined in (10) and (11) for the service provider and an intermediate node, respectively:

$$\text{DS}\left(m^C, k^C, l^C\right)_S^{\min} = m^C \times l^C \times I^C, \tag{10}$$

$$\text{DS}\left(m^C, k^C, l^C\right)_S^{\min} = m^C \times I^C. \tag{11}$$

The test conditions for data persistence-based admission control are similar to the conditions for average bandwidth-based admission control and hence will be skipped in this paper for sake of brevity.

Admission Control Test Procedure. If a service request passes all three admission control tests (i.e., resource usage-based, node mobility-based, and data persistence-based), then the service is accepted at node A with the requested (m, k, l) requirement and the service discovery request (SDREQ) message is rebroadcasted in node A's neighborhood. On the other hand, if the service request fails one or more of the admission control tests then the SPIRIT QoSM rejects the service request. Figure 4 shows the flow of the key QoS mechanisms at the client, intermediate, and service provider nodes over a multihop path of N intermediate nodes. When the subscription renewal interval is up, the SPIRIT client signals its QoS requirements for the next SI via a new service-path discovery. The intermediate node IN_0 receives the SDREQ message and forwards it to the local admission control process. If the QoS requested in the SDREQ message passes all the admission control tests then IN_0 forwards the message to all those neighbors which are located closer to the service provider than itself. The same procedure is repeated by all the next intermediate nodes $\text{IN}_1, \text{IN}_2, \ldots, \text{IN}_{N-1}$ before the SDREQ message is ultimately received by the service provider node. When the service provider receives a new SDREQ message for the next SI of an ongoing service stream, it reserves resources as requested in the SDREQ message and forwards a QoS Guarantee Acknowledgment (QoSGACK) message over the newly discovered multihop service path in the reverse direction. As each intermediate node in the service path receives this QoSGACK message, it reserves resources in a similar way as the service provider and then forwards the message to the next intermediate node. The QoSGACK message is ultimately received by the client node and serves as a notification to the client that the infrastructure has reserved necessary resources for supporting its QoS requirements for the next SI. Finally, at the onset of the next SI, the service provider begins transmitting the service update messages over the multihop service path while meeting the requested QoS requirements.

6. Evaluation

In order to evaluate the performance of SPIRIT, we performed simulations to study the impact of complex network

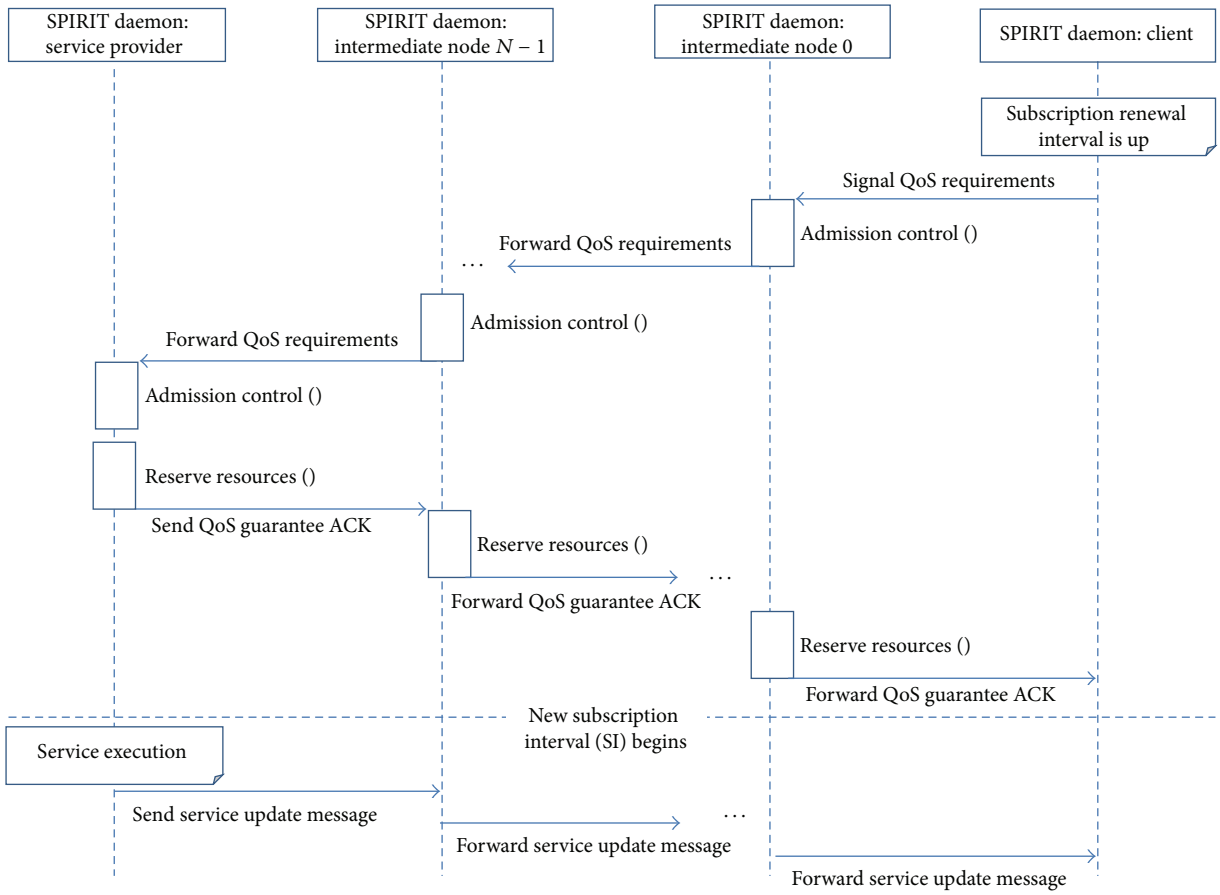

Figure 4: Flow of key QoS mechanisms in SPIRIT.

parameters. In addition to simulation, we also implemented SPIRIT as a proof-of-concept Android app and performed a small-scale experiment to see how accurately the simulation results match the results from a real system. As described later, the real experimental results exhibit similar trends as the simulation results, albeit on a smaller scale. The simulation and proof-of-concept app experimental setups and results are described below in detail.

6.1. Simulation. We investigated the performance of SPIRIT using the OPNET simulator [35].

6.1.1. Simulation Strategy and Mechanisms. In order to implement the QoS mechanisms for SPIRIT in OPNET, we have used a similar approach as proposed in [36–38]. In this approach, resources (e.g., processor and buffer space) are preallocated for satisfying the QoS requirements of sensitive services. Additionally, queuing techniques are utilized for QoS provisioning as they result in predictable network performance by providing dedicated bandwidth, controlled jitter and latency, and improved packet loss characteristics.

As described above, SPIRIT adapts the (m, k) firm guarantee model to specify the QoS requirements of the proximity-aware applications and further extends it to a (m, k, l) relaxed requirement to enable the client to specify the level of data persistence with the additional parameter l. On the other hand, there are two levels of QoS management schemes available in OPNET, that is, for (i) local and (ii) global QoS requirements. Previous work [39] has primarily focused on the global QoS requirement for developing (m, k) firm requirement and utilized the Weight Fair Queuing (WFQ) [8] scheme. In order to implement the (m, k, l) relaxed requirement in SPIRIT, we utilized a similar approach that considers both local and global QoS requirements. To that end, the parameters (m, k) together represent the global requirement and the parameter l represents the local requirement. The global requirement is used to guarantee packet arrival of any m service update message from k consecutive messages. In the simulation this is achieved by appropriate assignment of the WFQ weight value. The local requirement allows each client to request a persistent QoS by using the parameter l which in turn is utilized in specifying the Maximum Average Bandwidth per Single Client at each intermediate node. In the simulation, we adjust the relative bandwidth of each queue while forwarding the data packets to their destinations.

While the WFQ part is not integral to the proposed QoS mechanisms in SPIRIT, the purpose of using it in the simulation is to show an implementation path for the QoS mechanisms in real networks. To that end, WFQ is widely used by network schedulers for data packet scheduling. It is

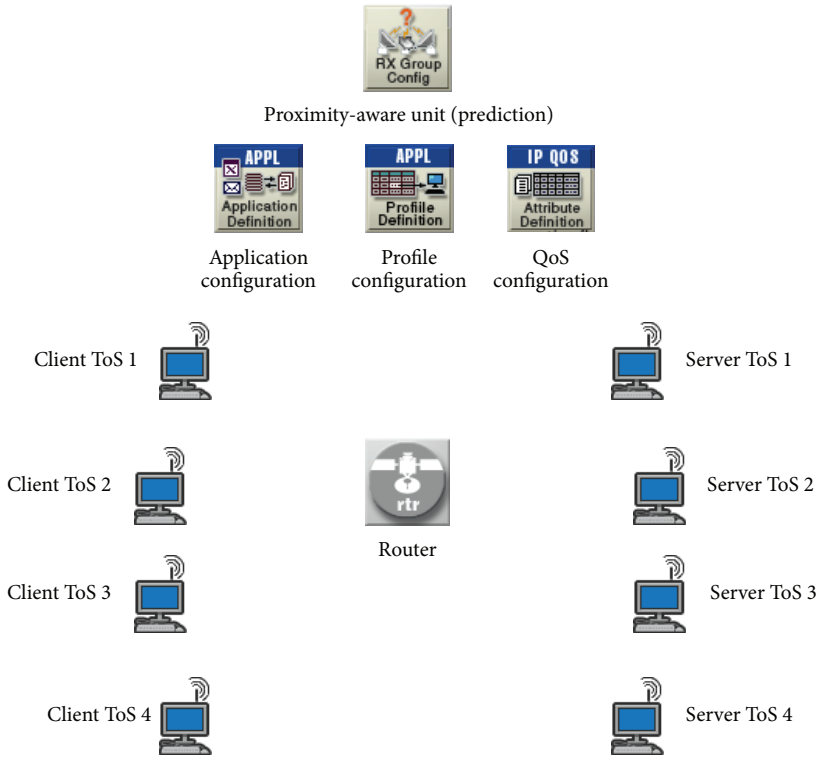

Figure 5: Network used to evaluate global QoS requirements ((m, k) ratio).

a packet-by-packet based implementation of generalization of Fair Queuing (FQ); that is, WFQ allows specifying, for each flow, which fraction of the capacity will be given. This makes WFQ an attractive choice for an implementation path for SPIRIT QoS. Additionally, previous work [40] has already proved the value of (m, k) firm guarantee based WFQ for real-time applications (e.g., multimedia); that is, it provides a well-defined delay guarantee for network flows which are tolerant of packet drops as defined by the (m, k) requirement. Based on this previous work and the proposed SPIRIT service model based on a relaxed (m, k, l) requirement, a modified (m, k, l)-WFQ is a natural choice of demonstrating the real network implementation opportunities for SPIRIT.

In addition to the (m, k, l) relaxed requirement, SPIRIT also performs admission control that considers node mobility and proximity for selecting intermediate nodes. In OPNET the proximity-awareness can be indicated by parameters like the received signal strength (path loss threshold dB), distance threshold, and channel match criteria. We use the distance threshold as the criteria for selecting next intermediate nodes for data packet forwarding.

6.1.2. Simulation Setup. We created a network with dimension of $100 * 100\,\mathrm{m}^2$. We assume a random mobility pattern for all nodes; every node uses the GRP routing protocol. Every forwarding/router node in the network uses the WFQ (class-based) scheme with a buffer size of 100 Bytes to implement the (m, k, l) method. Nodes can identify their neighbors within a range of 20 meters. Table 1 summarizes the node and the network configurations parameters.

Table 1: Network parameters.

Network dimension	100×100
Mobility pattern	Random
Simulation time	600 sec
Ad hoc routing	GRP protocol
ip QoS parameters (router configuration)	(i) WFQ (class-based) (ii) Buffer size: 100 Bytes (iii) Reserved bandwidth type: absolute
Proximity threshold	20 m
Supported services	(i) Background (ii) Standard traffic (iii) Excellent effort (iv) Streaming

Figure 5 shows the simulation scenario in which we employed a group of four servers and clients and one router to evaluate the performance of SPIRIT utilizing the WFQ queuing scheme. In this scenario, we evaluated the global QoS requirement by assigning the weight value (as a priority parameter) equal to the m/k ratio. In other words, each service has a priority (i.e., the relative weight in the WFQ scheme) which is determined by the m/k ratio. The router forwards data packets to the clients based on this weight value.

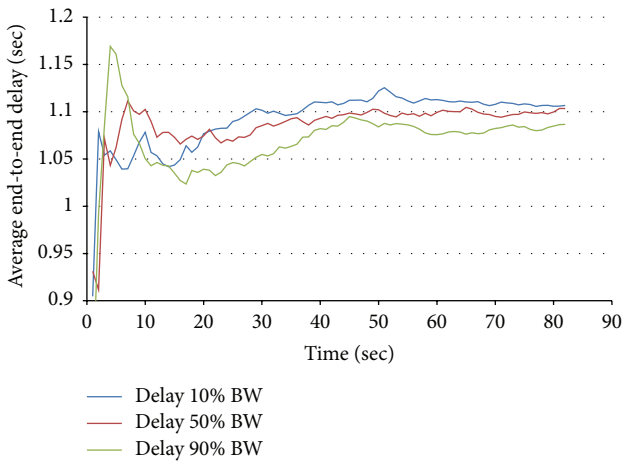

Figure 6: Average delay considering the (m, k) ratio as the weight value for WFQ.

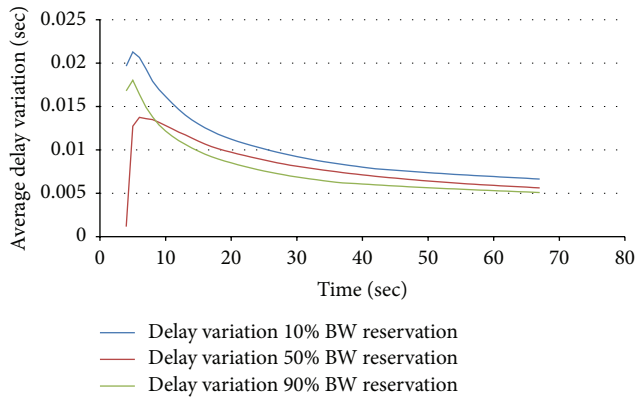

Figure 7: Average delay variation considering the (m, k) ratio as the weight value for WFQ.

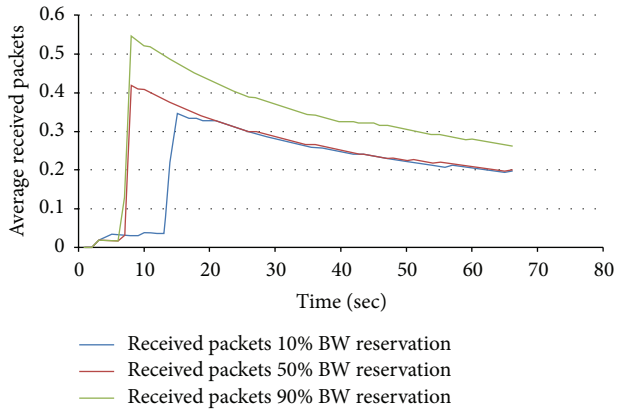

Figure 8: Average number of received packets for different weight value.

6.1.3. Simulation Results

Global Requirements. We studied the performance of the network in terms of three metrics, that is, delay, delay variation, and number of received packets for clients. In the first simulation scenario, we only consider the global QoS requirement, that is, (m, k) requirement.

Figure 6 shows the average delay for three services with varying weight values ($W = 10, 50$, and 100). This figure shows that increasing the m/k ratio results in a reduction of the average delay. Figure 7 shows the average delay variation for the same three services. This figure shows that increasing the relative weight value results in an increase of the average delay variation. We will show later in this section how SPIRIT addresses this issue by utilizing the (m, k, l) relaxed requirement. Figure 8 presents the average packet reception by clients. This figure shows that increasing the relative weight results in an increase of the packet reception. It is noteworthy that no significant improvement in packet reception is achieved by increasing the weight value by more than $W = 50$. This phenomenon is also corroborated by Figure 7; that is, increasing the relative weight value results in an unfavorable impact on the average delay variation. Figure 9 shows that the network throughput increases with increasing value of relative weight (as corroborated by previous results for packet reception presented in Figure 8). The reason why increasing the relative weight value results in improved packet reception rate and throughput is that an increase in the relative weight translates to assignment of larger buffer spaces for the network services. However, an important parameter for satisfying the QoS requirement is the rate of packet forwarding from the buffer. This parameter is not considered in the (m, k) global QoS requirement and we will show next how adding the local parameter l in SPIRIT further improved the QoS.

Global and Local Requirements. In the next simulation scenario, we demonstrate how adding a third parameter l helps improve the network performance in SPIRIT. We implemented the third parameter l (i.e., local data persistence level/buffer size requirement) by adjusting the ratio of reserved bandwidth to the buffer size for each queue which in turn translates to configuring the router so that it can forward data packets with respect to the local QoS of nodes in the network. Note that although the simulation explicitly used a router for simplicity of implementation, in a typical MANET it is expected that all nodes are capable of assuming the role of a router and adjusting the packet forwarding parameter accordingly. Therefore, a node with (m, k) global requirement can request higher QoS by assigning a higher reserved bandwidth to local buffer size requirement (i.e., l) ratio to the router.

Figures 10–13 show the network performance utilizing the (m, k, l) relaxed requirement. Figure 10 shows that increasing the reserved bandwidth for a service results in decrease of the average delay. Figures 11, 12, and 13 show similar positive impact of increasing reserved bandwidth on (reduced) average delay variation, (increased) packet reception, and (increased) packet throughput, respectively. Note that these figures show the effect of reserved bandwidth assignment when the global requirement is determined by $W = 50$; any other value of W yields similar results.

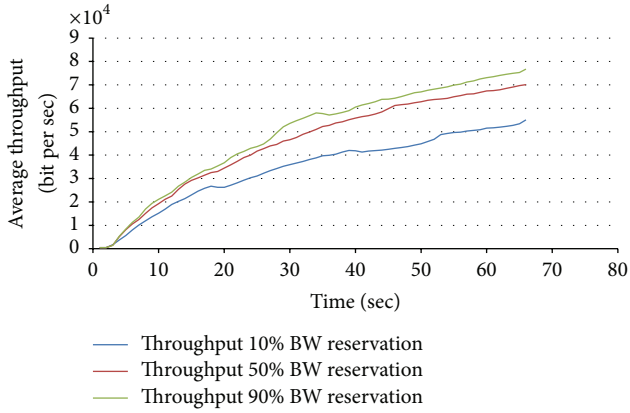

Figure 9: Throughput for different weight values.

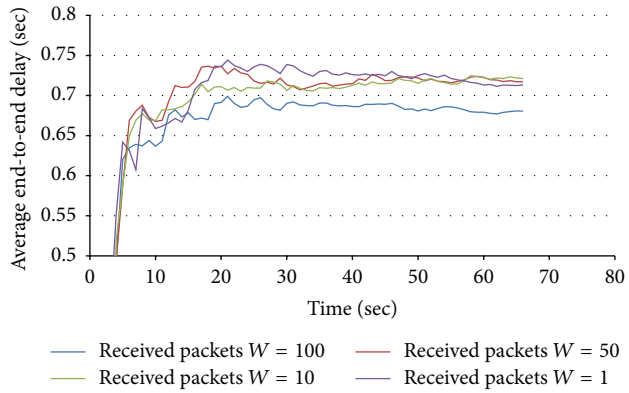

Figure 10: Average delay for different reserved bandwidth assignment (buffer size = 100 Bytes).

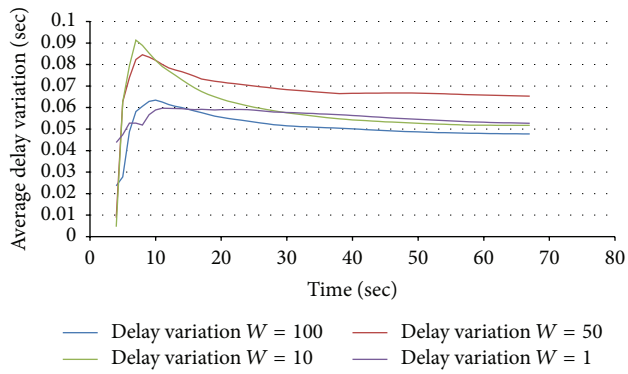

Figure 11: Average delay variation for different bandwidth assignment (buffer size = 100 Bytes).

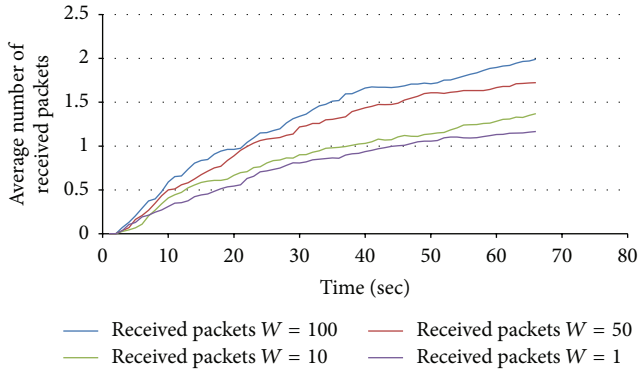

Figure 12: Received packets for different bandwidth assignments.

Scalability. As discussed above, SPIRIT effectively improves the QoS performance in comparison to simple (m, k) firm guarantee approach in a small-scale network. On the other hand, scalability is a serious concern in MANETs. In order to study how SPIRIT performs in a large-scale network, we implemented a network of one server and a group of 32 clients that also act as routers. Figure 14 shows that SPIRIT offers no significant improvement in the QoS performance in large-scale networks.

Reliability. In order to evaluate the reliability performance of SPIRIT, we simulated a simple network including a server and a client and set of intermediate nodes to forward data packets. We utilized a constant ratio for (m, k, l) requirement (i.e., $W = 50$ (m/k ratio) and $l = 75\%$ of bandwidth) of the network and changed the number of intermediate nodes (from 1 to 3) to evaluate the routing signaling and reliability performance in SPIRIT. Figure 15 shows that the average number of packets dropped by intermediate nodes decreases with increasing number of intermediate nodes. However, it is noteworthy that while higher path density (i.e., higher number of intermediate nodes) improves routing reliability, it also increases the average signaling overhead between nodes. We demonstrated this effect in Figure 16 that shows the average received routing traffic in the network. Figure 17 presents the (m, k) requirement success ratio, that is, the ratio of number of received packets which met their (m, k, l) requirement to the number of sent packets. As seen in Figure 17, the success ratio is 100% when m/k is low (i.e., 60%), and it decreases as m/k increases (i.e., $<$ 40% when $m/k = 1$).

Mobility Prediction. In order to evaluate the performance of the mobility prediction model utilized by SPIRIT admission control process, we selected a random mobility pattern for all intermediate nodes and changed the degree of prediction for temporal availability of intermediate nodes by varying the coverage area for admission control. Figures 18 and 19 show the average received traffic and average routing traffic, respectively. These figures show that utilizing a larger coverage area (i.e., increasing predicted temporal availability of intermediate nodes) results in increased received and routing traffic.

6.2. Proof-of-Concept Implementation. This section describes a proof-of-concept implementation of simple QoS provisioning within SPIRIT infrastructure. The BREMON app presented in Section 1 was implemented and evaluated on off-the-shelf Android smartphones. The underlying SPIRIT infrastructure uses the Android Bluetooth API for connecting the phones in a peer-to-peer ad hoc fashion: this ad hoc

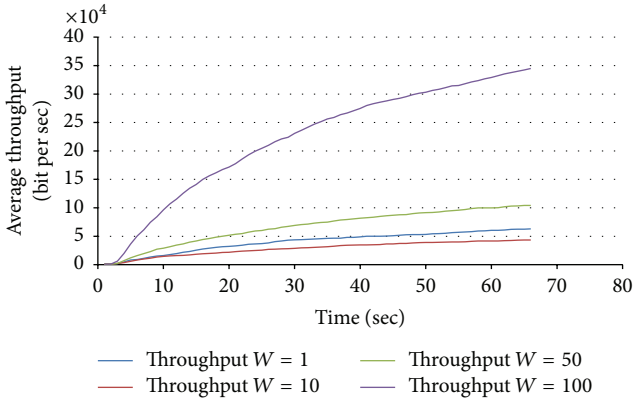

Figure 13: Throughput for different bandwidth assignments.

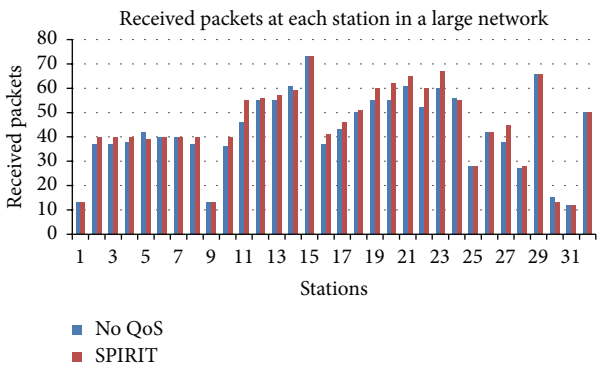

Figure 14: Throughput of clients in a large-scale network.

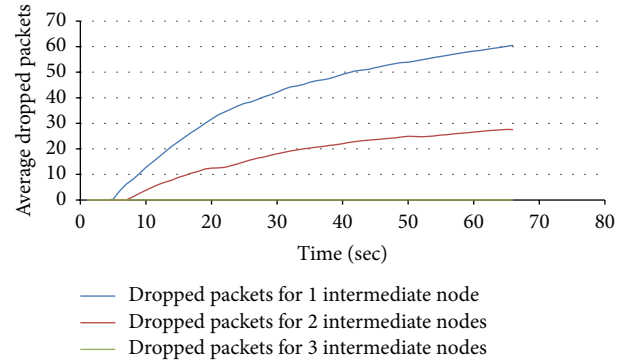

Figure 15: Average number of dropped packets for varying intermediate node density.

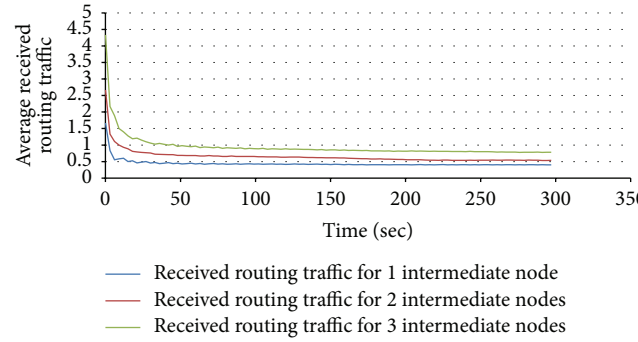

Figure 16: Average routing traffic for varying intermediate node density.

networking capability enables paramedics using BREMON to monitor patients in a postdisaster site where Wi-Fi/3G network availability may be inconsistent or disrupted. Furthermore, using Bluetooth enables BREMON to consume less energy than a Wi-Fi/3G network.

In order to evaluate the performance of BREMON, we performed experiments in a testbed of 5 Google Nexus One smartphones running Android 2.3 (Gingerbread) Operating System. Ten runs of experiments were conducted for each metric and the final results were taken as the average of the ten experiments. The duration of each experiment was 600 seconds. A two-hop network topology was created as shown in Figure 20. Phones A and B served as two service provider devices, phones D and E served as two client devices, and phone C served as the intermediate node connecting the service provider and the client devices in a peer-to-peer multihop fashion. There are two periodic service streams τ_1 and τ_2 which are flowing between the service provider and client pairs {A, D} and {B, E}, respectively. The service stream τ_1 releases a new service update message every 1 second. The service stream τ_2 releases a new service update message every 100 milliseconds. We varied the value of the m/k ratio (i.e., the (m, k, l) guarantee requirement) of both services in the range of 50–100% in steps of 10%. Using this setup we evaluated the performance of BREMON as described below.

Figure 21 presents the (m, k, l) *guarantee success rate*, that is, the number of service update messages in each service stream which met their (m, k, l) guarantee requirement. As shown in Figure 21, both service streams have 100% (m, k, l) guarantee success rates when the m/k ratio is low: the success rate decreases as the value of the m/k ratio increases. Note that the success rate of the service stream τ_2 is slightly less than that of service stream τ_1; this is attributed to the fact that in τ_2 the periodic service update messages are released more frequently than in τ_1, so it is likely that more number of service update messages from service stream τ_2 would be lost/dropped in the wireless medium under the same (m, k) constraint as in service stream τ_1.

Figure 22 presents the *communications overhead*, that is, the total number of control packets sent/received at all nodes in each service stream. Figure 22 shows that the communications overhead increases with increasing value of m/k ratio; that is, as $m/k \cong 1$ the SPIRIT daemon running on the mobile devices adds more control packets to the network as a result of the increasing number of data loss and subsequent retransmission requests made by the clients using the data persistent SPIRIT API. On that note, the service stream τ_2 incurs more communications overhead than service stream τ_1 because of its higher service execution frequency and consequently shorter subscription intervals.

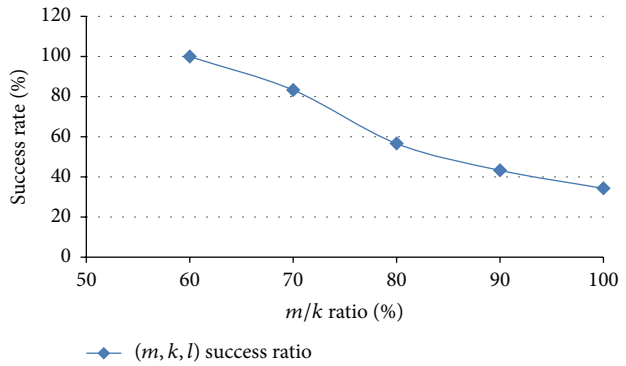

Figure 17: Success ratio for varying (m, k, l) requirement.

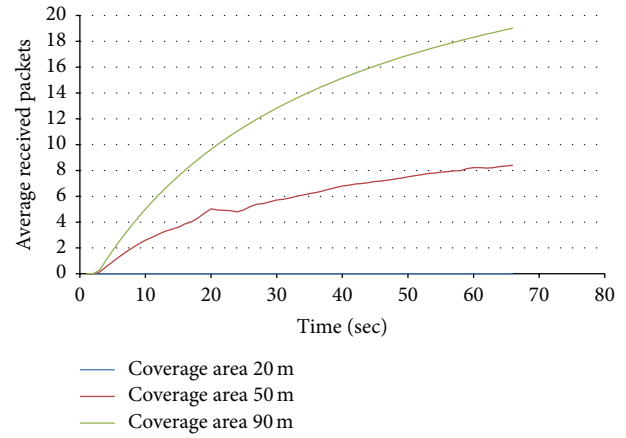

Figure 18: Average received traffic for varying coverage area.

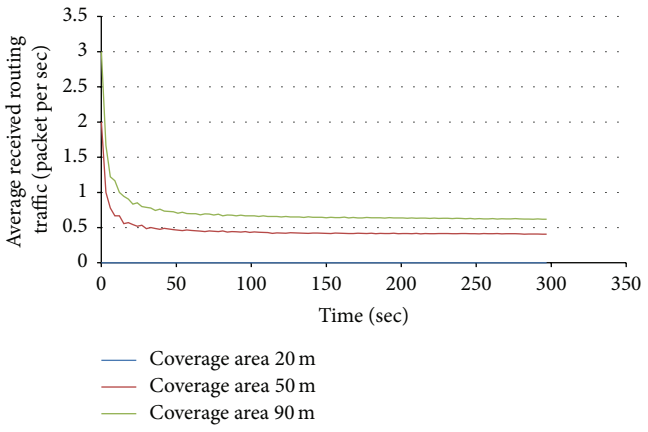

Figure 19: Average routing traffic for varying coverage area.

7. Conclusion and Future Work

Providing QoS in MANETs is a highly challenging problem. This paper presents a set of mechanisms for supporting soft QoS within a proximity-aware mobile service sharing infrastructure called SPIRIT. The importance for the proximity-aware applications to be able to specify and receive simple QoS such as reliable and persistent data delivery

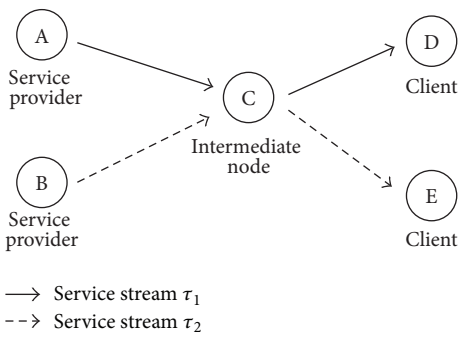

Figure 20: Experimental topology with multiple service streams.

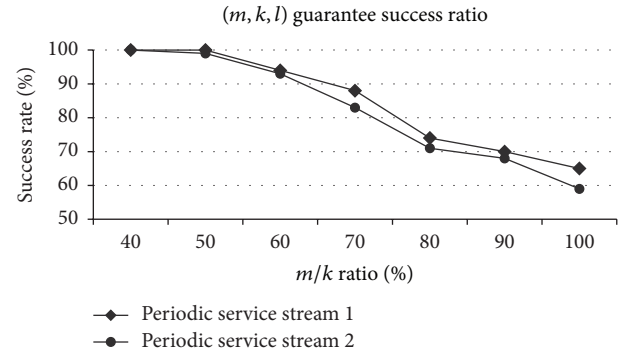

Figure 21: Percentage of service update messages in each service stream which met their (m, k, l) requirement.

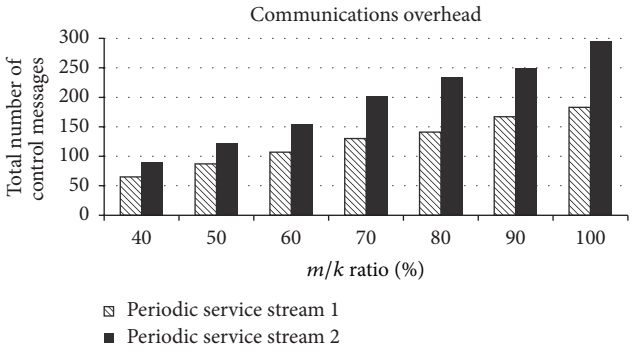

Figure 22: Total number of control messages in each service stream.

has been recognized and implemented in this paper. A real application scenario where off-the-shelf smartphones are used by paramedics to monitor the breathing activities of multiple patients at once at a disaster site is also discussed. The breathing activity data pertains to service provided by the infrastructure. SPIRIT runs as a background daemon on the patient and paramedic devices (which serve as the service providers and clients, resp.) and facilitates the processes of finding other devices, sharing sensor data, and specifying QoS requirements. The set of mobility-aware mechanisms presented in this paper (e.g., QoS signaling, QoS classification, admission control, and resource policing) enable the implementation of the QoS specifications provided by the

applications; to that end, they help to continuously reestablish and maintain the required QoS along multiple short-lived, multihop paths between service providers and clients, while handling mobility and node resources in an efficient way. Simulation results indicate that considering the local QoS requirement can improve the network performance in terms of average delay, delay variation, throughput, and packet reception.

The proposed SPIRIT infrastructure is limited in that it only provides simple QoS guarantees such as reliable and persistent data delivery whereas there are additional critical QoS requirements such as timeliness and security that remain to be included. Future work will include provisioning of latency requirements and priorities for most crucial service requests. In addition, approaches for specifying QoS requirements as range of values [41–44] will also be investigated: treating resource reservations as ranges will enable the flexibility required for operation in a dynamic environment. Future work will also look at augmenting the admission control procedure with adaptive approaches via support for dynamic feedback.

Competing Interests

The authors declare that they have no competing interests.

References

[1] F. Souza and I. Kushchu, "Mobile disaster management system applications—current overview and future potential," Tech. Rep., International University of Japan, Minamiuonuma, Japan, 2011.

[2] V. W. Consulting, "mhealth for development: the opportunity of mobile technology for healthcare in the developing world," Tech. Rep., United Nations Foundation, Washington, DC, USA, 2009.

[3] E. G. Nilsson and K. Stølen, "Ad hoc networks and mobile devices in emergency response—a perfect match?" in *Ad Hoc Networks: Second International Conference, ADHOCNETS 2010, Victoria, BC, Canada, August 18–20, 2010, Revised Selected Papers*, vol. 49 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 17–33, Springer, Berlin, Germany, 2010.

[4] P. Mitra and C. Poellabauer, "Emergency response in smartphone-based mobile ad-hoc networks," in *Proceedings of the International Workshop on Mobile Consumer Health Care Networks, Systems and Services (ICC '12)*, pp. 6091–6095, IEEE, Ottawa, Canada, June 2012.

[5] P. Mitra and C. Poellabauer, "Service sharing in mobile sensing systems," in *Proceedings of the Joint Workshop on Complex Networks and Pervasive Group Communication in Conjunction with IEEE GLOBECOM (GC Wkshps '11)*, pp. 110–114, IEEE, Houston, Tex, USA, December 2011.

[6] P. Mitra and C. Poellabauer, "Efficient group communications in location aware mobile ad-hoc networks," *Pervasive and Mobile Computing*, vol. 8, no. 2, pp. 229–248, 2012.

[7] M. Hamdaoui and P. Ramanathan, "A dynamic priority assignment technique for streams with (m, k)-firm deadlines," *IEEE Transactions on Computers*, vol. 44, no. 12, pp. 1443–1451, 1995.

[8] C.-C. Li, S.-L. Tsao, M. C. Chen, Y. Sun, and Y.-M. Huang, "Proportional delay differentiation service based on weighted fair queuing," in *Proceedings of the 9th International Conference on Computer Communications and Networks*, pp. 418–423, IEEE, Las Vegas, Nev, USA, October 2000.

[9] R. Baldoni, R. Beraldi, S. T. Piergiovanni, and A. Virgillito, "Measuring notification loss in publish/subscribe communication systems," in *Proceedings of the 10th International Symposium Pacific Rim Dependable Computing*, pp. 84–93, March 2004.

[10] A. Corsaro, L. Querzoni, S. Scipiono, S. T. Piergiovanni, and A. Virgillito, "Quality of service in publish/subscribe middleware," in *Global Data Management*, R. Baldoni and G. Cortese, Eds., Emerging Communication: Studies in New Technologies and Practices in Communication, pp. 79–97, IOS Press, Amsterdam, The Netherlands, 2006.

[11] J. Hoffert and D. C. Schmidt, "Maintaining qos for publish/subscribe middleware in dynamic environments," in *Proceedings of the International Conference on Distributed Event-Based Systems(DEBS '09)*, ACM, Nashville, Tenn, USA, 2009.

[12] H. Luo, S. Lu, V. Bharghavan, J. Cheng, and G. Zhong, "A packet scheduling approach to QoS support in multihop wireless networks," *Mobile Networks and Applications*, vol. 9, no. 3, pp. 193–206, 2004.

[13] V. Bharghavan, K.-W. Lee, S. Lu, S. Ha, J.-R. Li, and D. Dwyer, "The timely adaptive resource management architecture," *IEEE Personal Communications*, vol. 5, no. 4, pp. 20–31, 1998.

[14] I. Dionysiou, D. Frincke, D. E. Bakken, and C. Hauser, "Actor-oriented trust," Tech. Rep. EECS-GS-006, School of Electrical Engineering and Computer Science, Washington State University, 2005.

[15] L. Fiege, A. Zeidler, A. Buchmann, R. Kilian-Kehr, and G. Muhl, "Security aspects in publish/subscribe systems," in *Proceedings of the IEEE 3rd International Workshop on Distributed Event-Based Systems (DEBS '04)*, Edinburgh, Scotland, 2004.

[16] R. Misra and C. R. Mandal, "Performance comparison of aodv/dsr ondemand routing protocols for ad hoc networks in constrained situation," in *Proceedings of the IEEE International Conference on Personal Wireless Communications (ICPWC '05)*, January 2005.

[17] D. Vali, S. Paskalis, L. Merakos, and A. Kaloxylos, "A survey of internet QoS signaling," *IEEE Communications Surveys & Tutorials*, vol. 6, no. 4, pp. 32–43, 2004.

[18] I. Jawhar and J. Wu, "Quality of service routing in mobile ad hoc networks," in *Resource Management in Wireless Networking*, M. Cardei, I. Cardei and, and D. Z. Du, Eds., pp. 365–400, Kluwer Academic, Boston, Mass, USA, 2004.

[19] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture: an overview," Internet Draft RFC 1633, 1994.

[20] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation protocol (RSVP)," Internet Draft RFC 2205, 1997.

[21] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," Internet Draft RFC 2475, 1998.

[22] E. Kartsakli, A. S. Lalos, A. Antonopoulos et al., "A survey on M2M systems for mhealth: a wireless communications perspective," *Sensors*, vol. 14, no. 10, pp. 18009–18052, 2014.

[23] S. Tennina, M. Di Renzo, E. Kartsakli et al., "WSN4QoL: a WSN-oriented healthcare system architecture," *International*

Journal of Distributed Sensor Networks, vol. 2014, Article ID 503417, 16 pages, 2014.

[24] O. Bouachir, F. Garcia, N. Larrieu, and T. Gayraud, "Ad hoc network QoS architecture for cooperative unmanned aerial vehicles (UAVs)," in *Proceedings of the 6th IFIP/IEEE Wireless Days Conference (WD '13)*, pp. 1–4, IEEE, Valencia, Spain, November 2013.

[25] G. Ahn, A. Campbell, A. Veres, and L. H. Sun, "Swan: service differentiation in stateless wireless ad-hoc networks," in *Proceedings of IEEE 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, vol. 2, pp. 457–466, New York, NY, USA, 2002.

[26] E. W. Knightly and N. B. Shroff, "Admission control for statistical QoS: theory and practice," *IEEE Network*, vol. 13, no. 2, pp. 20–29, 1999.

[27] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, 1996.

[28] E. Ibarra, A. Antonopoulos, E. Kartsakli, J. Rodrigues, L. Alonso, and C. Verikoukis, "Qos-aware energy management in body sensor nodes powered by human energy harvesting," *Sensors*, vol. 16, no. 2, pp. 542–549, 2016.

[29] P. Mitra and C. Poellabauer, "Asymmetric geographic forwarding: exploiting link asymmetry in location aware routing," *International Journal of Embedded and Real-Time Communication Systems*, vol. 2, no. 4, pp. 46–70, 2011.

[30] N. Mabrouk, N. Georgantas, and V. Issarny, "A semantic end-to-end qos model for dynamic service oriented environments," in *Proceedings of the ICSE Workshop on Principles of Engineering Service Oriented Systems*, Vancouver, Canada, May 2009.

[31] S. Frolund and J. Koistinen, "Quality of service specification in distributed object systems design," in *Proceedings of the 4th Conference on USENIX Conference on Object-Oriented Technologies and Systems (COOTS '98)*, Santa Fe, NM, USA, April 1998.

[32] Y.-D. Lin, C.-H. Chang, and Y.-C. Hsu, "Bandwidth brokers of instantaneous and book-ahead requests for differentiated services networks," in *Proceedings of the IEEE Global Telecommunicatins Conference (GLOBECOM '01)*, pp. 2285–2289, San Antonio, Tex, USA, November 2001.

[33] A. G. Greenberg, R. Srikant, and W. Whitt, "Resource sharing for book-ahead and instantaneous-request calls," *IEEE/ACM Transactions on Networking*, vol. 7, no. 1, pp. 10–22, 1999.

[34] O. Schelén and S. Pink, "Sharing resources through advance reservation agents," in *Building QoS into Distributed Systems: IFIP TC6 WG6.1 Fifth International Workshop on Quality of Service (IWQOS '97), 21–23 May 1997, New York, USA*, IFIP—The International Federation for Information Processing, pp. 265–276, Springer, New York, NY, USA, 1997.

[35] O. M. Documentation, Opnet Technologies, Inc, 2003, http://www.opnet.com.

[36] M. Aamir, M. Zaidi, and H. Mansoor, "Performance analysis of DiffServ based quality of service in a multimedia wired network and VPN effect using OPNET," *International Journal of Computer Science Issues*, vol. 9, no. 3, article 368, 2012.

[37] H. A. Mohammed, A. H. Ali, and H. J. Mohammed, "The affects of different queuing algorithms within the router on QoS VoIP application using OPNET," *International Journal of Computer Networks & Communications*, vol. 5, no. 1, pp. 117–124, 2013.

[38] I. A. Lawal, A. M. Said, and A. A. Muazu, "Simulation model to improve qos performance over fixed wimax using opnet," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 6, no. 21, pp. 3933–3945, 2013.

[39] A. Koubâa and Y.-Q. Song, "Graceful degradation of loss-tolerant QoS using (m,k)-firm constraints in guaranteed rate networks," *Computer Communications*, vol. 28, no. 12, pp. 1393–1409, 2005.

[40] A. Koubaa and Y.-Q. Song, "(m,k)-WFQ: integrating (m,k)-firm real-time constraints into guaranteed-rate networks," in *Proceedings of the Conference on Real-Time Systems—RTS Embedded Systems*, INRIA, 2004.

[41] S. Lu and V. Bharghavan, "Adaptive resource management algorithms for indoor mobile computing environments," in *Proceedings of the Annual Conference of the Special Interest Group on Data Communication (SIGCOMM '96)*, pp. 231–242, 1996.

[42] O. Angin, A. T. Campbell, M. E. Kounavis, and R. R.-F. Liao, "The mobiware toolkit: programmable support for adaptive mobile networking," *IEEE Personal Communications*, vol. 5, no. 4, pp. 32–43, 1998.

[43] S. Lee and A. Campbell, "Insignia: in-band signaling support for QoS in mobile ad hoc networks," in *Proceedings of the 5th International Workshop on Mobile Multimedia Communications*, Berlin, Germany, October 1998.

[44] M. Mirhakkak, N. Schult, and D. Thomson, "Dynamic bandwidth management and adaptive applications for a variable bandwidth wireless environment," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 10, pp. 1984–1997, 2001.