NOOP: An IoMT System for Notifying Public Security Issues and Increasing Police Patrol Coverage

Camila A. Wanous, Flávia Pisani, Markus Endler
Department of Informatics
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rio de Janeiro/RJ, Brazil
cantonaccio13@gmail.com, {fpisani,endler}@inf.puc-rio.br

***Abstract*—Although public security is still one of the major problems in most large cities, uninterrupted overt patrolling and rapid police intervention have been identified as effective mechanisms to combat street theft and crime. However, several police departments do not have access to technologies for reliable data communication and workforce management to be notified about occurrences and effectively respond to them. This paper presents the design and evaluation of NeighborhOOd Patrolling (NOOP), an open-source Internet of Mobile Things system for notifying public security issues and increasing police patrol coverage. This tool is being used as an initial version for the "Segurança Presente 5.0" overt patrolling initiative in the city of Rio de Janeiro, Brazil.**

***Index Terms*—Smart Cities, Internet of Things, Public Security, Police Patrol Coverage**

I. Introduction

As concern for public security in cities grows, the demand for smart city initiatives to tackle this issue also rises. For this reason, there has been a surge in the number of proposals for applications that use the Internet of Things (IoT) to prevent crime and increase public security [1].

In areas where safety is a predominant problem, the population may seek to be informed in real-time about possible risk zones; thus, it is essential to provide mechanisms for citizens to be notified about occurrences that happen close to them or to where they live. There is already an initiative with this goal from the Municipal Secretariat for Urban Safety in the city of Sao Paulo, Brazil. In 2017, they launched an application to facilitate access to information in real-time, allowing users to warn each other about occurrences (e.g., lack of light, falling trees, graffiti, or any other crime) [2].

One of the postulates of routine safety activity theory states that crimes occur when a criminal finds a suitable target at a time and a place where there is no adequate protection [3]. As such, patrols, consisting of police officers on foot or in vehicles, can effectively prevent crime. Therefore, increasing the distribution of patrols to expand the patrolled area can lead to crime prevention.

In this paper, we discuss patrolling as a way for several police units to monitor an area by traveling around it and responding to occurrences together. It is the responsibility of these patrols to respond immediately to occurrences near their location. The faster they arrive at the event as a group, the more effective we can consider them. As a result, redirecting units closer to the event shortens the response time and prevents patrols in unfavorable positions from moving unnecessarily, leading to increased patrol effectiveness. In this case, controlling the number of patrols that respond to a particular occurrence would also prevent an excessive number of units from being relocated, which could hinder the distribution of patrols in the region.

For instance, consider a scenario where N patrols are allocated to secure a region of a city, and that these units are concentrated in the southern part of this region. In these circumstances, the northern areas would be unprotected and, therefore, more susceptible to crime. In this hypothetical situation, the response time for occurrences in the north would be longer as patrols would take more time to arrive. From that, it follows that if some of the N patrols were allocated to the northern areas, this would reduce that part of the region's susceptibility to crime and average response time to occurrences.

In this situation, we can consider that the closest patrols' response time is usually shorter than all others. Furthermore, if all patrols were to go to each new occurrence, there would be a large amount of unnecessary movement. Also, many areas would be empty, given that there is a limit on the number of patrols needed to resolve an issue. Therefore, ideally, only the patrols closest to the occurrence should respond to it.

Considering this context, the need for a service that increases patrol coverage by efficiently allocating patrols in real-time becomes apparent, given that it could shorten response times to occurrences and possibly lead to safer cities. With that in mind, we propose an open-source IoT service called NeighborhOOd Patrolling (NOOP)[1] for incident reporting and patrol allocation using the ContextNet middleware [5] and the InterSCity platform [6]. NOOP enables users to register occurrences, notifies citizens and patrols about nearby occurrences, and redistributes patrols based on real-time data about their positioning.

The main contributions of this paper are: the design and implementation of a flexible police force dispatch service that

The authors thank CAPES and FAPESP (grant #2019/19312-9) for the financial support, and the INCT of the Future Internet for Smart Cities for the technological support.

[1]NOOP is available at https://bitbucket.org/cwanous/noop-project/ and is part of the "Segurança Presente 5.0" initiative [4].

aims to increase patrol coverage by relocating units based on real-time information; the evaluation of this system with simulations that help us assess the impact that relocation strategies can have on patrol coverage and unit response time; and the evaluation of the system with scalability tests that verify the performance of NOOP in situations with a large influx of occurrences.

This text is organized as follows: Section II discusses related work, Section III introduces the models we used for the system and the patrolled regions, Section IV presents the overall architecture of NOOP, Section V describes implementation details, Section VI goes over the experimental setup and test results, and Section VII gives our conclusions and possibilities for future work.

II. Related Work

In recent years, there has been much discussion related to the topics of IoT and public security. In this section, we consider some of the investigations that are more closely related to our proposal.

Du and Zhu [7] list the main research venues and technologies behind alert and emergency management systems related to urban public security. Their study calls attention to several fundamental factors for good management, such as active decision making, having technologies that ensure the operation of nodes and the network for a long time in hostile environments, and data protection. Moreover, they conclude that systems that operate in real-time increase the city's ability to withstand emergencies and reduce the damage caused by them, thus indicating the effectiveness of platforms that are similar to NOOP.

NOOP has some of the essential technologies pointed out by Du and Zhu, given that it makes active decisions for the allocation of patrols and notification of occurrences, as well as guarantees the proper operation of the network in hostile environments by using the ContextNet middleware. However, due to the simplicity of the current implementation, it offers no data protection. Still, this feature could be supported by modifying NOOP to use a version of the ContextNet protocol MR-UDP with cryptography [8], and this is a significant improvement to include in future work.

Hochstetler, Hochstetler, and Fu [3] propose a patrol strategy for smart cities to maximize the response capacity of patrols in a scenario with limited resources (i.e., the number of police officers and vehicles). In the same way as NOOP, they start from the premise that the distribution of patrols can suppress possible crimes. However, while their investigation aims to design an optimal patrol distribution strategy, NOOP focuses on presenting an effective IoT architecture for implementing an allocation scheme based on the real-time location of patrols. Given that their optimal distribution approach considers the historical crime data in a particular region, we note that NOOP could be used to collect this information.

An article by Dunnet, Leigh, and Jackson [9] with support of the Leicestershire Police, UK, states that it is crucial that police forces operate in a cost-efficient manner and that the most efficient resources be allocated to respond to incidents. Their proposed framework combines mapping and routing algorithms to create a decision process to facilitate optimal patrol selection for incident response. They tested and validated their tool with simulations and noted that it reduced response times and increased response unit availability.

Their process of dispatching patrols, much like NOOP, uses information such as quickest response time, response unit availability, and demanded coverage. On the other hand, they also use data that NOOP does not currently consider, such as predicted traffic conditions and driver qualifications. Considering that their work focuses on the dispatching method and that they intend to use real-time data to improve their algorithm in the future, we point out that NOOP could be used in conjunction with their approach.

III. Model

This section presents our approach to modeling the regions of the city and their occurrences, as well as the allocation algorithms used in the current NOOP implementation. The analysis and proposition of an optimal patrol allocation strategy are outside the scope of this paper, so there may be more efficient approaches for the region model and the allocation algorithms described.

A. Region Model

NOOP considers that a region of the city is divided into areas and that each area requires only one patrol unit. While establishing this partition is essential for adopting this system, the criteria used to define the areas are flexible and different strategies can be easily implemented by simply modifying a NOOP interface called *CityAreaInterface*.

NOOP's default region division is a square split into smaller squares of equal size. The location of each area is defined by orthogonal Cartesian axes whose origin coincides with the region's lower-left limit. The sides of the areas measure one unit each, and the location of an area is defined as the center of the square that represents it. The areas are numbered for identification from left to right and from bottom to top, and the number of squares into which the region is divided is flexible.

B. Occurrence Model

The occurrences have location and type. The notifier informs both attributes, with the location being their geographical coordinates (latitude and longitude) and the type being a measure of the severity of the occurrence, which will indicate the maximum number of patrols required to resolve it (either 1, 2, or 3). The UK police forces have a similar classification [9]. Each notifier may only register occurrences in the same area where they are at the time of notification.

C. Patrol Allocation Algorithms

The patrol allocation algorithms have two main functions: *CalculateNewPatrolArea* and *PatrolAllocation*, which both receive a parameter with the list of patrols in operation.

CalculateNewPatrolArea returns the next area to be patrolled by a unit. Therefore, every time a unit is about to start

a new patrol, the allocation algorithm uses this function to assign it to an area. The algorithm can also check if there are patrols that need to migrate from one area to another to achieve a better distribution. It uses *PatrolAllocation* to obtain the identification of a unit and the new area to which it should be assigned.

Like the region model, the patrol allocation algorithm is flexible and can be re-implemented through *CityAreaInterface*, which has the *CalculateNewPatrolArea* and *PatrolAllocation* methods. In this subsection, we present three allocation approaches that we used to test NOOP, called Furthest Distance (NOOP FD), Nearest Distance (NOOP ND), and Increase Cover (NOOP IC). The following notation is used: $d(x, y)$ is the Euclidean distance between areas x and y, $\sum_{i=1}^{N} d(x, i)$ is the sum of the Euclidean distance from an area x to all other areas in a region of N areas, and $C(x)$ is the set of areas covered by a patrol in area x.

1) NOOP FD: The *CalculateNewPatrolArea* function of NOOP FD returns an area A for which the sum of the distances to the patrolled areas is maximum. This area is furthest from the patrols, so it is the area where the response time would be the longest. Placing a patrol in this area greatly decreases the response time to occurrences that might happen in or around it. In the case of NOOP FD, the sum of these distances is calculated for all areas that do not have a patrol, and the area with the largest sum is chosen. Figure 1 illustrates a case where patrols are placed in areas 6, 10, and 13. We see that $A = 4$ was chosen because $d(4, 6) + d(4, 10) + d(4, 13)$ is maximum.

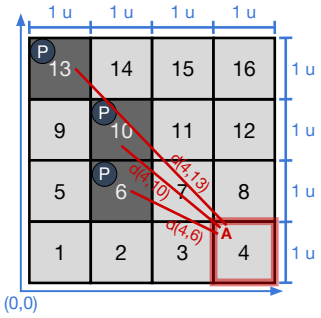

Fig. 1. Selecting the area that is the farthest from the patrols.

The *PatrolAllocation* function of NOOP FD uses *CalculateNewPatrolArea* to find out which area is the furthest from the patrols, selecting it as a possible new area to be patrolled. This function then pre-selects a candidate patrol for the assignment if it is in an area x such that $\sum_{i=1}^{N} d(x, i)$ is maximum (i.e., the patrol that would take the longest to move to other areas). It then only moves the patrol from area x to A if $\sum_{i=1}^{N} d(x, i) > \sum_{i=1}^{N} d(A, i)$. Figure 2 shows an example where there are patrols in areas 1 and 5 and we need to verify if either of them can be assigned to area 15. The patrol in area 1 was pre-selected for assignment because $\sum_{i=1}^{N} d(1, i) > \sum_{i=1}^{N} d(5, i)$. It then was moved to area 15, as $\sum_{i=1}^{N} d(1, i) > \sum_{i=1}^{N} d(15, i)$.

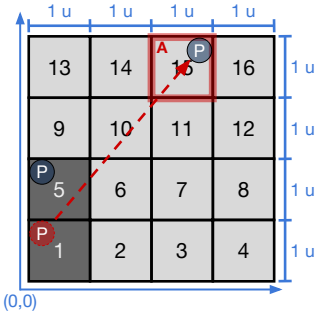

Fig. 2. The patrol in area 1 is pre-selected and moves to area 15.

2) NOOP ND: NOOP ND uses the same *CalculateNewPatrolArea* function as NOOP FD to both select the next area to be patrolled and to pre-select, in the *PatrolAllocation* function, the target area of the assignment (A). Nevertheless, the *PatrolAllocation* of NOOP ND pre-selects the patrol closest to A as a candidate for migration in order to reduce the time units spend moving between areas. If this patrol is in an area x, the function chooses to perform the migration from x to A if $\sum_{i=1}^{N} d(x, i) > \sum_{i=1}^{N} d(A, i)$.

3) NOOP IC: NOOP IC seeks to maximize the cardinality of the set of areas covered by patrols. Areas covered by a given patrol are the area where the patrol is located and the ones that are adjacent to it. For example, in Figure 3, the patrol positioned in area 11 is neighboring areas 6, 7, 8, 10, 12, 14, 15, and 16. Thus, these areas are part of $C(11)$, and the cardinality of the set is $|C(11)| = 9$. In NOOP IC, the *CalculateNewPatrolArea* function returns the area A with the lowest index that maximizes the cardinality of the set of covered areas (area 6 in the example of the figure).

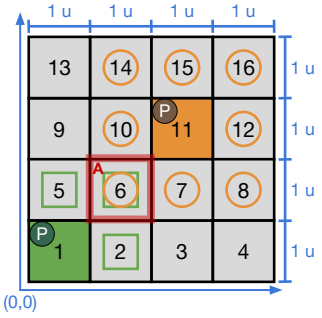

Fig. 3. Selecting the area A that maximizes $|C(A)|$.

The NOOP IC *PatrolAllocation* function uses *CalculateNewPatrolArea* to determine which area would maximize the cardinality of the set of covered areas if it had a patrol in it and then selects this area as a possible next assignment. The patrol whose migration maximizes the cardinality of the set of areas covered by all patrols is pre-selected. Figure 4 shows an example where there are patrols in areas 5 and 16, and we need to verify if one of them should be assigned to area 6. In Scenario 1 (patrol in area 5 migration), we have that $C(6) + C(16) = 12$ and in Scenario 2 (patrol in area 16

migration), $C(5)+C(6) = 9$. Therefore, the patrol in area 5 is pre-selected for moving. The cardinality of the set of covered areas after the migration is then calculated, and if it is greater than the set without the migration, the migration is performed. In the example, $|C(6) + C(16)| > |C(5) + C(16)|$, so the patrol moves from area 5 to 6.

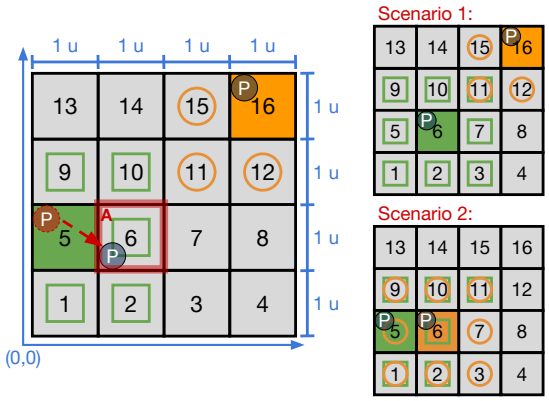

Fig. 4. The patrol in area 5 is pre-selected and moves to area 6.

IV. NOOP Architecture

This section presents the architecture of the NOOP system and the main technologies we used to develop it. Figure 5 illustrates a simplified version of this architecture, which is composed of five main entities: NOOP Resident, NOOP Patrol, ContextNet, NOOP Monitor, and InterSCity. NOOP Resident and NOOP Patrol are distinct types of mobile users that can dynamically connect and disconnect from the system. NOOP Monitor is the service responsible for the notification of incidents and patrol allocation. The ContextNet middleware performs the communication between the mobile elements and the service. NOOP Monitor records all activities on the InterSCity platform.

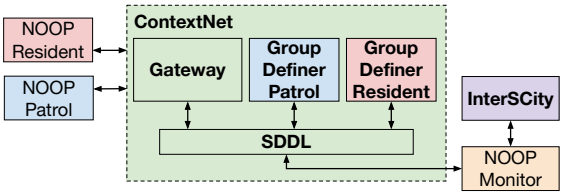

Fig. 5. NOOP architecture.

A. ContextNet

The ContextNet project [5] aims to provide context services for large-scale pervasive applications like online monitoring and mobile entity coordination. Figure 5 shows the communication layer of ContextNet, which is implemented using SDDL [10], and the other services and extensions that are built as software modules on top of it. The figure also displays the four services used in the implementation of NOOP: Gateway, Group Definer Patrol, Group Definer Resident, and NOOP Monitor.

The Gateway is responsible for defining connection points with the mobile nodes, managing various mobile node connections, and sending messages from the nodes to the services and from the services to the nodes. The communication between the Gateway and the mobile nodes is done via a two-way MR-UDP connection.

ContextNet's architecture allows unicast, groupcast, and broadcast communication between services and mobile nodes. The Group Definer Patrol and Group Definer Resident services are responsible for using context information to designate in real-time the groups to which each Patrol and Resident node belong, respectively, and then transmitting the composition of these groups to the Gateway. With this definition in place, the Gateway can address groupcast messages to the mobile nodes.

B. InterSCity

The InterSCity platform [6] is an open-source project designed to give technical support to the development of smart cities. Its main objective is to provide high-level services and APIs to facilitate the development of new city services, bringing together key technologies such as IoT, Big Data, and cloud computing. InterSCity adopts a microservice-based architecture created to handle the integration of large amounts of devices and data, thus being able to provide scalable services for entire cities.

The platform defines a city resource as a logical concept that encapsulates a physical entity that is part of the city, such as cars, buses, traffic lights, and light poles. Such resources have functional capacities and attributes (e.g., location and description), called capabilities. Table I has a description of the capabilities of the three NOOP InterSCity resources: Patrol, Resident, and Occurrence.

The state diagram of Figure 6 displays the possible values for the Position capability of the Patrol resource and the events that lead to each of them. Before starting, the Patrol is in the "Request Area" state. Upon receiving an area from the Monitor, it transitions to the "To Area" state and remains there while moving to the area to which it was assigned. If it does not receive any notification of an occurrence while moving to the designated area, the Patrol transitions to the "On Area" state and starts patrolling there. Otherwise, it transitions to the "To Occurrence" state.

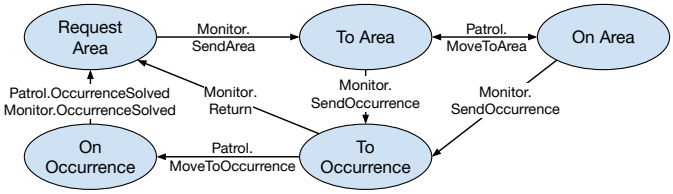

Fig. 6. State diagram for the Position capability of the Patrol resource.

While in the "On Area" state, the Patrol may receive a new area from the Monitor and transition back to the "To Area" state. It may otherwise receive a notification about an occurrence and transition to the "To Occurrence" state.

TABLE I
NOOP'S INTERSCITY CAPABILITIES AND RESOURCES.

Capability	Resource	Description
AreaID	Patrol, Resident, Occurrence	Area designated for a patrol or a resident's domicile
OccurrenceID	Patrol	Identification of the event to which a patrol is assigned
Latitude/Longitude	Patrol, Resident	Element's geographical coordinates
Position	Patrol	State of a patrol, see Figure 6
NotifierID	Occurrence	Identification of the element that reported the event
Situation	Occurrence	Situation, resolved or not, of the occurrence
Type	Occurrence	Type of occurrence

A Patrol remains in the "To Occurrence" state while moving to the occurrence. If it does not receive any return orders from the Monitor while moving, the Patrol transitions to the "On Occurrence" state. If it receives a return order, it transitions back to the "Request Area" state and starts a new patrol. A Patrol transitions from "On Occurrence" to "Request Area" when it resolves the occurrence to which it was assigned. As all patrols notify the Monitor at each state transition, the Monitor has enough information to allocate Patrols considering the position of all Patrols in the system.

Finally, the InterSCity platform provides the history of the resources registered in it over time and is therefore used to register and store all NOOP resource activities. The records of each InterSCity resource are updated whenever any capabilities, except "Latitude" and "Longitude", change. It also allows real-time queries to this history, enabling activity analysis at any time at which NOOP is operating. As the platform records the values of each capability over time, it can be used to obtain information about events such as registration of new residents and patrols, assignment of patrols to areas, changes in the state of patrols, which patrols responded to an occurrence, and the notification and resolution of occurrences.

C. NOOP Components

Figure 7 shows the main interactions between the entities that compose NOOP. The five types of components in the system are city residents that operate entities called Resident, patrols that operate entities called Patrol, a service called Monitor, and two Group Definer services from ContextNet (one for Patrols and another for Residents).

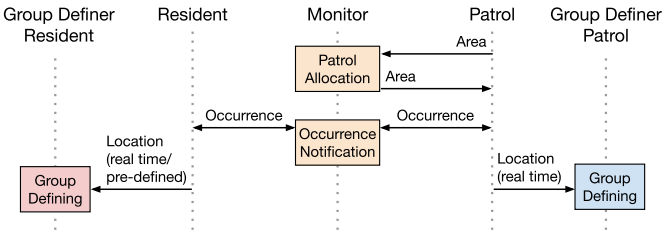

Fig. 7. Diagram for the interaction of NOOP entities.

Each component has a different function. Residents move freely around the region and periodically send their location to Group Definer Resident, in addition to sending the location of their homes and new occurrences to the Monitor. Patrols periodically send their location to Group Definer Patrol and send new occurrences, the state of old occurrences, and area requests to the Monitor. Moreover, they move to new areas or patrol inside areas indicated by the Monitor, as well as moving to occurrences and resolving them. The Monitor recalculates the areas patrolled by each Patrol from time to time to increase the distribution of units across the region. It sends occurrence state notifications to groups of Residents and Patrols, responds to area requests made by the Patrols, ensures that no more Patrols than necessary are sent to a particular occurrence, and records events in InterSCity.

The figure also illustrates the exchange of "Area" messages between a Patrol and the Monitor that occurs during the allocation of the Patrol. Before Patrols start patrolling, they request an area to the Monitor. The Monitor receives the request, calculates which area of the region is most unprotected from the positioning of the other patrols, returns the area that should be patrolled next, and records the Patrol in InterSCity. Once the Patrol receives the assignment, it begins to move towards the area and notifies the Monitor that it is on the move. When the migration is complete, the Patrol notifies the Monitor that it is in the designated area and begins patrolling. The Monitor receives the notification and updates Patrol's InterSCity record.

As occurrences happen, Patrols are displaced to respond to them, which leaves some areas uncovered. To minimize this, the Monitor uses the current patrol positioning to recalculate periodically the areas that should be patrolled, notifies the Patrols of their new assigned areas, and updates the Patrols' records in InterSCity.

In NOOP, each area of the region represents a group, and Patrols and Residents belong to the group of the area where they are currently located. Furthermore, Residents also belong to the group of the area in which they reside. As shown in Figure 7, the Group Definer Patrol and the Group Definer Resident receive as context information the location of the Patrols and Residents, respectively. From this location, the Group Definer uses the Region Model to define which areas and groups each of the Patrols and Residents belong to, thus enabling the Monitor to send groupcast messages.

Also in Figure 7, it is possible to see the exchange of "Occurrence" messages between the Monitor and a Patrol or

Resident. At any time, Patrols and Residents can notify the Monitor about an event by sending a "Occurrence" message containing the location and type of the occurrence. If the notifier is a Patrol, they begin to resolve the occurrence soon after the notification. When the Monitor receives an occurrence, it records it in InterSCity, uses its location to determine which groups of Patrols and Residents should receive a notification about it, and then notifies them. If the notifier is a Patrol, the Monitor records that there is already a patrol resolving the occurrence.

If a Patrol receives an occurrence report and is not already resolving another occurrence, it begins to move towards it and notifies the Monitor that they are on the move. The Monitor receives the notification and checks whether there are enough Patrols either on the way or already on the occurrence. If there are, the Monitor gives an order for the Patrol to return. The Patrol receives the return order and requests a new area to the Monitor. When a Patrol arrives at an event, it notifies the Monitor of its arrival. The Monitor receives the Patrol's position and updates the Patrol's InterSCity record.

After a Patrol resolves an occurrence, it notifies the Monitor that the occurrence has been resolved. The Monitor receives the notification, updates the InterSCity incident record, and then determines which Resident groups should receive a notification about the solution and notifies them. The notifications are sent to all Residents who reside or who are momentarily located in the same group as the occurrence and to Patrols located in the same group or in groups neighboring the occurrence (i.e., Patrols in the areas neighboring the area in which the occurrence is located).

If the Monitor knows about an occurrence but has not received a notification saying that Patrols are responding to it, it will forward the occurrence notification to all the Patrols. If all patrols are busy at the time this second notification is sent, the occurrence will go unanswered.

V. Implementation

This section provides details about the implementation of NOOP's components using the Java language. We considered that: the communication channels are secure, there are no failures in the nodes, there is no loss or duplication of messages, and there is no guarantee of finite delays or ordered message delivery. Exploring these issues is left as future work.

All NOOP components use a library called *noop-data* that contains the following classes: *Area*, which defines the attributes and methods of an area; *DataType*, which enumerates the types of system messages; *Location*, which defines the attributes and methods of a location; *NoopJASONData*, which standardizes the message content format; *CityAreaInterface*, which represents the city region according to the model described in Section III; and *PatrolPosition*, which enumerates the possible Patrol states.

Resident and Patrol are mobile nodes. All communication between them and the Gateway is carried out with the help of *ClientLib*. To do this, they must implement the *NodeConnectionListener* interface, which contains methods for notification of connection and disconnection to the Gateway, as well as notification of the arrival of new messages sent by the Gateway. Mobile nodes create a *MrUdpNodeConnection* and place themselves as listeners to the Gateway to establish a connection to the Gateway and send messages.

A Resident sends *SendLocationTask* to be executed by a thread pool every five seconds, which sends their location information to the Gateway. To simulate a citizen's behavior, two other threads, *ResidentMoveTask* and *SendOccurrenceTask*, are also scheduled to run every five and ten seconds, respectively. *ResidentMoveTask* simulates the movement of the citizen through the region following the model described in Section III and *SendOccurrenceTask* simulates the observations of occurrences by the Resident with a certain probability each time it is executed.

Patrol, like Resident, sends its location with a *SendLocationTask* thread every five seconds. To simulate a patrol's behavior, *SendOccurrenceTask* is also scheduled to run every ten seconds. It simulates Patrol observations of occurrences with a certain probability each time it is executed. Also, for simulation purposes, we implemented the *PatrolMove* class, which is used by Patrols to simulate the migration to areas and occurrences following the model described in Section III.

The Monitor is the most complex component of the system. It implements the *UDIDataReaderListener* interface that contains methods for notifying the arrival of new messages sent by the Gateway. The *UniversalDDSLayerFactory* class is instantiated to create the connection to the Gateway and send messages to it. To connect to InterSCity, the *ConnectionInterSCity* class is instantiated and becomes responsible for sending requests and returning their results. The *MonitorData* class is responsible for the formulation of the request syntax, which is specific to NOOP. To periodically recalculate the areas patrolled by each Patrol, the Monitor programs a *PatrolAllocationTask* thread to be executed every thirty seconds by a thread pool. *PatrolAllocationTask* is responsible for running the reallocation algorithm described in Section III.

Group Definer Resident and Group Definer Patrol instantiate a GroupDefiner that connects to the Gateway and receives the context messages. It also implements the *GroupSelector* interface that is responsible for the algorithm that determines the group composition. Both Group Definers have similar implementations, diverging only in the determination algorithm and the *getGroupType*, a *GroupSelector* interface method that returns the group type.

VI. Experiments

This section describes the experimental setup and results of the tests we made to assess the effectiveness and scalability of NOOP. Hardware test setup comprised one real computer running one ContextNet Gateway in one process; two Group Definers, each in a different process; and mobile nodes (Patrols and Residents). The mobile nodes were simulated in a Thread Pool, where each one was allocated in a thread and scheduled to send simulated coordinates through MR-UDP to the Gateway periodically.

A. *Effectiveness Tests*

This subsection describes the tests we performed to verify the effectiveness of adding group reporting and a patrol allocation algorithm to a patrolling system. We used the following metrics: the average time between recording an occurrence and all patrols arriving at its location (**TimeAR**); the average time between recording an occurrence and the first patrol arriving at its location (**TimeFR**); and the percentage of the time of operation in which a patrol is effectively patrolling (**EffePatrol**), that is, the time when a patrol is not on the move or resolving an occurrence. It is important to evaluate both **TimeAR** and **EffePatrol** as there are occurrences that need more than one patrol to be solved, as explained in Section III-B.

1) Experimental Setup: We used the data from a study [11] made by the Institute of Public Safety of Rio de Janeiro, Brazil, to estimate the frequency of occurrences in a 16 km^2 neighborhood in the South Zone of the city of Rio de Janeiro. Following the numbers of this study, we simulated a frequency of eight occurrences per hour, with four patrols running on all tests.

To compare the effectiveness of NOOP in different patrol configurations, we tested two regions of different sizes, one simulating 16 km^2 divided into 16 areas and another simulating 25 km^2 divided into 25 areas. This last simulation represents a scenario with a greater scarcity of resources than the first, given that the same number of units patrols a larger area. Each patrol and resident randomly move through the territory, changing their direction every five seconds with a probability of 50%. In all cases, we considered that the patrols and the residents are moving to the occurrences on foot at 9 km/h.

We designed five scenarios: the first without using any NOOP functionalities (**WN**), the second using group reporting (**GR**), the third using group reporting and NOOP FD (**GR+FD**), the fourth using group reporting and NOOP ND (**GR+ND**), and the fifth using group reporting and NOOP IC (**GR+IC**). We analyzed all scenarios for the 16-area region and, for simplicity, only **WN**, **GR**, and **GR+IC** for the 25-area regions. We used the same set of fifteen different shifts (i.e., work periods) to test all scenarios. Each of these shifts has a sequence of eight randomly generated occurrences, where each occurrence consists of the area where it is located and its type.

2) Experimental Results: Figure 8 has examples of the effectiveness test results we obtained for the average arrival time of all patrols. As these times were simulated, we present them as "time units" (t.u.).

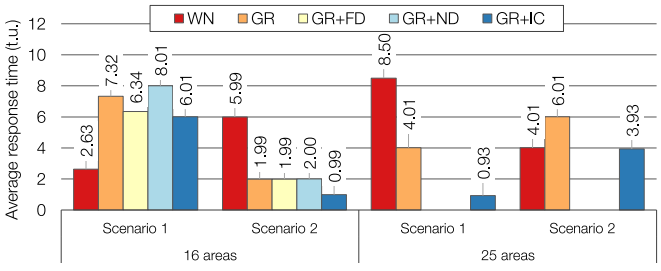

Fig. 8. Examples of tested scenarios for **TimeAR** (lower is better).

We can see that there are cases, such as 16-area Scenario 2 and 25-area Scenario 1, where adding group reporting by itself or with an allocation algorithm improves **TimeAR**. However, there are also cases, such as 16-area Scenario 1 and 25-area Scenario 2, where **WN** presents better results.

When considering all tested 16-area scenarios, we observed that in 68.33% of cases, **GR** outperformed **WN**, with **GR+FD**, **GR+ND**, and **GR+IC** being faster than **WN** in 68.33%, 70.00%, and 78.33% of cases, respectively. For 25-area tests, we see that **GR** and **GR+IC** presented better results than **WN** in 57.98% and 89.08% of cases, respectively. Furthermore, we point out that only 5.83%, 5.00%, 5.83%, 3.33%, 8.40%, and 0.84% of cases presented results that had response times over twice that of **WR** for 16-area **GR**, **GR+FD**, **GR+ND**, and **GR+IC** and 25-area **GR** and **GR+IC**, respectively.

Figure 9 shows the average results considering all tested scenarios (i.e., fifteen shifts with eight occurrences each). We see that for both 16 areas and 25 areas, **TimeAR** is, on average, better when using one of the NOOP strategies than with **WN**. Moreover, for 16 areas, **GR**, **GR+FD**, **GR+ND**, and **GR+IC** decrease **TimeAR** to less than half when compared to **WN** in 32.50%, 44.17%, 46.67%, and 43.33% of cases, respectively, while **GR** and **GR+IC** do the same in 15.13% and 52.10% of cases, respectively.

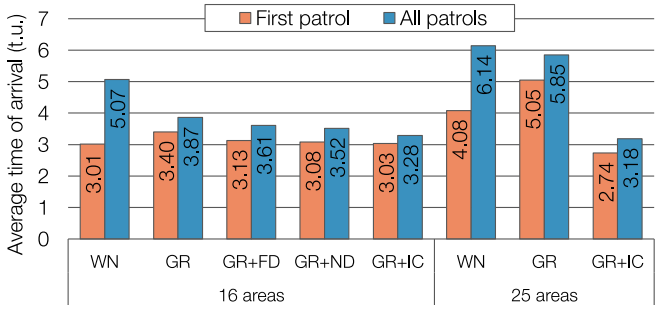

Fig. 9. Average time of patrol arrival (lower is better).

On the other hand, when we look only at **TimeFR**, we see that on average **WN** presents the best results. Nevertheless, we point out that using NOOP usually does not lead to a large increase in this metric, given that in 56.67% (**GR**), 60.83% (**GR+FD**), 65.00% (**GR+ND**), and 59.17% (**GR+IC**) of cases for 16 areas and in 55.83% (**GR**) and 71.67% (**GR+IC**) of cases for 25 areas, the value of **TimeFR** is up to 1.1× that of the result for **WN**.

Figure 10 shows that using NOOP increased the **EffePatrol** metric in all tested scenarios when compared to the **WN** baseline. In particular, we highlight the case of **GR+IC** for 16 areas, which reached 52.64% of effective patrolling time while **WN** presented only 13.87%. For the 25-area region, **EffePatrol** for **WN** was 8.7% and we see that adding simply **GR** or **GR+IC** more than triples the percentage of time that patrols are in operation.

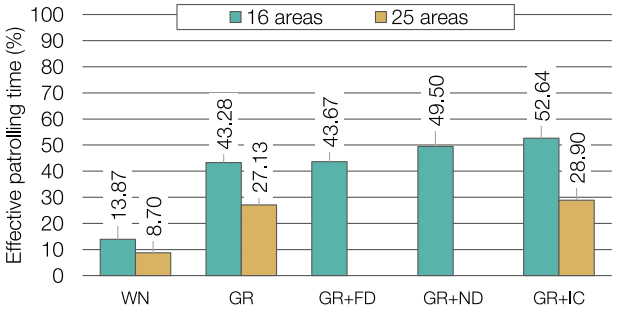

Fig. 10. Effective patrolling time (higher is better).

B. Scalability Tests

This section describes the tests performed to check the responsiveness of NOOP Monitor in the cases where there is increased occurrence frequency, a scenario that can be made possible if NOOP is widely adopted. The time interval between the Monitor recording the occurrence and sending the message to the patrol groups (dispatch) was used as a metric. In a scenario where NOOP needs to handle a large amount of data, the point of the system that would be stressed the most would be the Monitor; thus, this is the main component that must be tested to evaluate the scalability of the system. We note that in order to verify the scalability of NOOP, the tested frequencies are far above those seen in the real world.

1) Experimental Setup: For the scalability test, we analyzed the following frequencies: 3,600, 7,200, 36,000, 72,000, and 720,000 occurrences per hour, that is, periods of 1,000, 500, 100, 50, and 5 milliseconds between occurrences, respectively. We ran ten simulations for all periods, five with ten occurrences in a row, and five with 30 occurrences in a row. The area's size and the number of patrols do not interfere with this metric, so they do not interfere with the test result. We used the second scenario of the Effectiveness Tests (**GR**), which only uses notification by groups.

2) Experimental Results: Figure 11 shows the results of the scalability tests. The vertical axis shows the average time interval between the occurrence record and the dispatch measured in milliseconds with the error bars indicating the standard error. The horizontal axis shows the occurrence periods from 1000 to 5 ms, first with ten and then with 30 occurrence messages. Each point corresponds to one test. We observed that regardless of the frequency of occurrences, the process takes between 100 and 200 milliseconds in almost all cases except one outlier, which could have been caused by changes in the network condition.

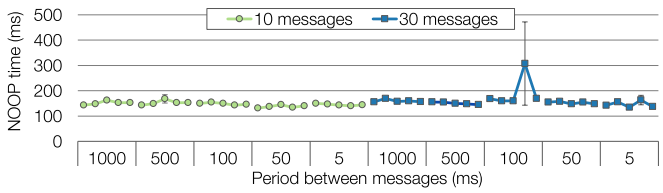

Fig. 11. Scalability tests.

VII. Conclusions and Future Work

Public security is an increasing societal concern, and, in this context, we see that smart cities can provide citizens with mechanisms that make them safer. Therefore, this paper presents NOOP, an open-source system for incident reporting and patrol allocation.

Our tests indicated that having such a system effectively reduces the response time of occurrences and increases the effectiveness of patrolling. The tests also showed that the reallocation algorithm should be elaborated according to the region's characteristics and size; therefore, they should be reassessed at each implementation. The architecture of NOOP makes it easy to test and implement such algorithms for each specific region. In future investigations, it is possible to test and discuss the best patrolling strategies and the best algorithms to implement them, as well as other ways to model regions. Implementing a queuing mechanism for unanswered occurrences is also a possibility.

Moreover, group notification proved to be an essential feature, as it alone reduced response time and increased the effectiveness of patrolling. Future work can test different compositions for the groups, taking into account, for example, other context information when defining them. Such context information could be captured by sensors integrated with the patrols and sent through the smartphones already used in the NOOP operation. The types of instruments and vehicles available to the patrol are possibilities for this type of information.

Finally, the scalability tests demonstrated that NOOP could be used in situations with very frequent occurrences, thus being robust enough to act in real-world scenarios.

References

[1] N. Dlodlo, P. Mbecke, M. Mofolo, and M. Mhlanga, "The Internet of Things in Community Safety and Crime Prevention for South Africa," in *Proc. CISSE 2012 & CISSE 2013*, 2015, pp. 531–537.

[2] Sao Paulo Municipal Secretariat for Urban Safety, "SP+SEGURA: aplicativo conta com a colaboração da população," Feb. 2019, https://www.prefeitura.sp.gov.br/cidade/secretarias/seguranca_urbana/noticias/?p=261057. [In Portuguese, access. Dec. 18, 2019].

[3] J. Hochstetler, L. Hochstetler, and S. Fu, "An Optimal Police Patrol Planning Strategy for Smart City Safety," in *Proc. HPCC/SmartCity/DSS*, Dec. 2016, pp. 1256–1263.

[4] Instituto Kuidamos, "Segurança pública 5.0," 2020, https://kuidamos.com/seguranca-publica-5-0/. [In Portuguese, access. Aug. 30, 2020].

[5] M. Endler and F. S. e Silva, "Past, Present and Future of the ContextNet IoMT Middleware," *OJIOT*, vol. 4, no. 1, pp. 7–23, 2018.

[6] D. M. Batista, A. Goldman, R. Hirata, F. Kon, F. M. Costa, and M. Endler, "InterSCity: Addressing Future Internet research challenges for Smart Cities," in *Proc. NOF*, Nov. 2016, pp. 1–6.

[7] C. Du and S. Zhu, "Research on Urban Public Safety Emergency Management Early Warning System based on Technologies for the Internet of Things," *Procedia Eng.*, vol. 45, pp. 748–754, 2012.

[8] J. F. Gonçalves, F. J. da S. e Silva, R. O. Vasconcelos, G. L. B. Baptista, and M. Endler, "A Security Infrastructure for Massive Mobile Data Distribution," in *Proc. MobiWac*, 2013, pp. 41–50.

[9] S. Dunnet, J. Leigh, and L. Jackson, "Optimising police dispatch for incident response in real time," *JORS*, vol. 70, pp. 269–279, 2018.

[10] L. David, R. Vasconcelos, L. Alves, R. André, and M. Endler, "A DDS-based middleware for scalable tracking, communication and collaboration of mobile nodes," *JISA*, vol. 4, no. 16, 2013.

[11] J. Fernandes and T. Falheiros, "Estímulo à sistematização de informações da segurança pública: o caso do 2o Batalhão de Polícia Militar do Estado do Rio de Janeiro," *Cadernos de Segurança Pública*, vol. 11, no. 11, 2019, [In Portuguese].