DroneSensor - drone-based sensor data collection

Ana Rita da Silva Pires
rita.silva.pires@tecnico.ulisboa.pt

Instituto Superior Tecnico, Lisboa, Portugal

Abstract— **Delay Tolerant Networks result from the need to transfer information on wireless mobile networks in situations where traditional networks tend to fail. The main characteristic of these networks is the lack of continuous communication between their nodes and the difficulty of ensuring the delivery of the information to the recipient. This type of network is used especially in situations where the mobile nodes of the network are widely dispersed. This thesis proposes an implementation of a low cost system that allows the acquisition of sensor data installed in areas without infrastructure, using drones.**

I. INTRODUCTION

Delay-tolerant networks result from the need to transfer information on wireless mobile networks - MANETs - in situations where communication is sporadic, and it is not possible to guarantee connectivity between the intervening nodes [1]. The main features of DTN are the lack of continuous communication between its nodes and the difficulty of ensuring the delivery of information to the recipient. This type of networks are used especially in situations where the mobile nodes of the network are widely dispersed [2].

There are several domains and examples of application of DTN, with the most common applications being vehicle fleet management, rural infrastructure communication networks, and search and rescue operations [3].

The very peculiar properties of DTN, such as limited resources, connection loss, and delayed delivery of information cause some problems with their creation and configuration. These problems involve resource allocation, buffer space, energy expenditure, reliability, and security.

II. RELATED WORK

A. Delay Tolerant Networks

The DTN concept initially emerged as an approach to communication in space or on an interplanetary scale. In this type of environment, the existence of a large latency, sometimes measured in hours or even days, and the frequent loss of connection is unavoidable. However, these problems also apply to sites not so distant but in which there is no infrastructure that allows communication using the previously existing protocols [3]. The very peculiar properties of DTN, such as limited resources, frequent loss of connection and delay in the delivery of information cause some problems with their creation and configuration, hence it is not possible to use protocols used in the so called regular networks. These problems involve resource allocation, buffer space, energy expenditure, reliability, and security.

B. Routing Protocols

Since the focus of DTN is to be as efficient as possible, and due to their specific characteristics mentioned earlier, typical ad-hoc protocols, such as the Mobile Ad-hoc Networks protocols, fail to establish paths in DTN. To apply the concept referred above, there is a great choice in terms of routing protocols. These are divided essentially into two families [4]:

- Flooding Families
- Forwarding Families

The protocols of the Flooding Families work in a replication scheme (creation of some copies of the original message or package, and subsequent delivery). Each node has a copy of each message, which is transmitted to a set of nodes named relays. All nodes keep copies, which are stored in a buffer, and will be transmitted on the established connection. Replication-based protocols allow better message delivery rates when compared to routing-based protocols, and do not require prior knowledge of the network topology. The major disadvantage of the protocols in this family is related to the storage space that each node has to contain, since it keeps copies of the messages to send.

In Forwarding Families, the information about the network topology is used to choose the best path, and then the message is routed node after node to the destination. The protocols belonging to this family require some knowledge about the topology of the network, in order to send only one copy of the message by the best path, and it is not necessary to use replication, saving storage space.

Bundle Protocol is a protocol suitable in both flooding and forwarding families, depending on which routing mode we choose to implement. This protocol packs blocks of data into packets, and transmits them using store-and-forward. Using this protocol allows the data to be stored for extended periods of time until it can be transmitted, even in situations where network connectivity is not constant and there are bandwidth breaks. However, delivery of the package to the destination node is guaranteed [5].

This protocol requires the use of a Convergence-Layer adapter to send and receive bundles using existing connections, networks, or protocols. [6]

The IBR-DTN implementation of this protocol includes four distinct routing modules, which are managed by the Base Router, and can be used concurrently. The first module, Static, routes between static routes defined a priori in the configuration file, which it assumes always available. The Neighbor module forwards the packets to the neighboring nodes discovered by the Discovery Agent. If IPND - IP Neighbor Discovery is used, the nodes present in the same subnet are reached. There is also an implementation of the Epidemic Routing protocol, defined in the Epidemic module. This type of forwarding is a variation of flooding families protocols, referred above. It aims to forward bundles to all participating nodes, and in case the resources are unlimited, guarantees the possible delivery of a given bundle. When a bundle is delivered to the destination, it is deleted from the network so that duplicate bundles are not received. To do this we use the information contained in summary vectors. The last module specified in this implementation is the Retransmission module, which as its name indicates, retransmits bundles in case of an error in its sending and consequent arrival at the destination. The re-transmission depends on the nature of the error occurred, permanent or temporary.

These modules receive the Discovery Agent events and interact with the device storage. The four routing modules described above allow the use of various routing protocols according to the user's preference.

- **Default routing** - Only delivers bundles to neighbors and static nodes.
- **Epidemic routing** - Send to all available neighbors.
- **Flooding routing** - Like the epidemic, but does not send the own summary vector to the neighbors.
- **Prophet routing** - Forward based on the probability of finding other nodes.
- **No routing**

C. Tecnology

1) Sensors: A sensor is a device capable of converting a physical quantity to be measured in a signal that can be read, displayed, stored and processed. There are several types of sensors, with very different characteristics, depending on the intended use.

The most common sensors depend on another unit, such as a micro-controller, that receives, processes and sends the acquired data. There are sensors for various applications, such as temperature, humidity, movement, among others. As an example of this type of sensors we have one of the most popular temperature sensors, the LM35, which is a 3-pin integrated circuit (+5V, GND and output). This particular sensor detects temperatures between -55 C and 150 C, with an error of approximately 0.5C [7].

2) Micro-controllers: Micro-controllers are required to do all the gathering and processing of information. The main features that make these devices the best option at the expense of, for example, a mobile device are related to its cost, low power consumption, weight and flexibility in the use of operating systems and programming languages. In addition, they allow the addition of modules for the most diverse situations, such as communication modules, GPS modules, among others.

Rasperry Pi is a micro-computer running a LINUX-based operating system. As well as working as a micro-controller, it has some processing power and is therefore eligible for more complex data processing tasks. Its memory is expandable through the use of a larger memory card, or even a USB storage device [8].

3) Drones: For the proposed project, the drone will have as function the collection of data provided by the sensors. To do this, it is necessary to have autonomy to go through the sensor network. In addition, in order to avoid collisions, the presence of a camera will be advantageous and bearing in mind that the route must be programmable, so its API must be available. For data collection to be possible, the drone must still support extra weight so that the micro-controller can be added to it. At present, the market for this type of devices offers a wide variety of products.

The DJI drones, popular for the model Phantom, have characteristics similar to the needed ones. A Software Development Kit is available which allows the user to obtain images of the route and also define the route that the drone should follow. In addition to the Phantom models, there is Matrice 100, which is a fully programmable drone suitable for development. This model is prepared to be added external devices (sensors, controllers, cameras, among others). Its autonomy depends on the weight added to the basis weight of the drone [9].

III. PROPOSED SOLUTION

The proposed project consists in the use of DTN to collect data from pre-established sensors using drones and delay-tolerant networks with optimization of the route and use of suitable network protocols. For this, three types of nodes are considered: collector nodes, responsible for acquiring data from the sensors aggregated to them, the mobile node, attached to a drone, that moves and collects the data acquired by the collector nodes, and finally the manager node that keeps a copy of the received data, processes it and allows its visualization by the user.

Despite the various applications already mentioned, this network will be implemented within the Instituto Superior Tecnico building on Taguspark Campus. Once implemented, it can be adapted to other situations, such as the monitoring of agricultural land and gardens.

Three types of nodes are considered: collector node, mobile node and manager node. The collector nodes consist of a set of sensors and a micro-controller that manages the information given by the sensors, and sends through a WiFi module. The mobile node consists of a drone with a micro-controller added and powered by a battery, and has the function of collecting data next to the collector nodes. This node, as its name implies, will be in motion in order to be within range of each of the collector nodes. Finally, the manager node consists of a computer that handles the information sent by the mobile node and allows its visualization. The figure 1 presents a general diagram of the network and the communication between the different nodes.

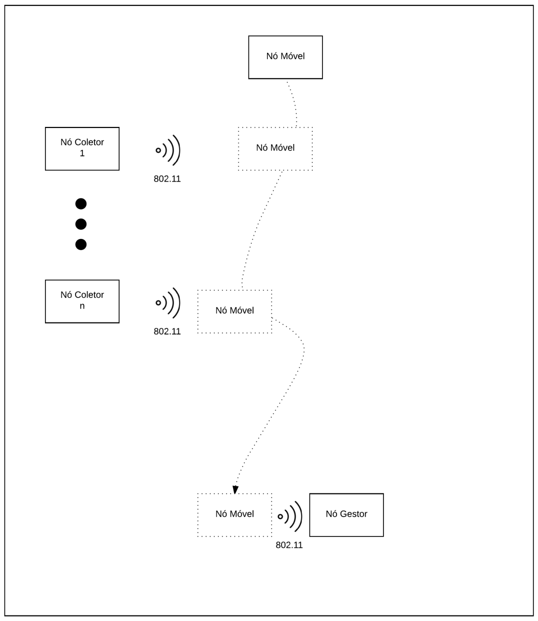

Fig. 1. Network Diagram

A. Software Architecture

1) Collector node: The collector node, as previously mentioned, is responsible for collecting the data obtained through sensors in contact with the outside. These sensors collect data from the environment, and store it locally in a file. This file contains the date and time of the acquisition, the node identifier, and the data collected from the various sensors. Periodically a acquisition is performed and a line is added to the file. The figure 2 illustrates the software architecture of this type of node. Taking into account that the clocks of the various collector nodes are not synchronized with each other, the time of collection is not precise and may differ by several seconds, which is not relevant for the type of data acquisition made.

2) Mobile node: The mobile node receives data from the collector nodes and sends it to the manager node. Its function is only to receive the files generated by the collector nodes and deliver them to the manager node after communication between them is established, as shown in figure 3.

3) Manager node: The manager node is responsible for processing the data contained in the file generated by the collector nodes. This node receives the files from the mobile node and inserts them into the database. The database can be accessed on this same node through a local page. The figure 4 illustrates the software architecture of this node.

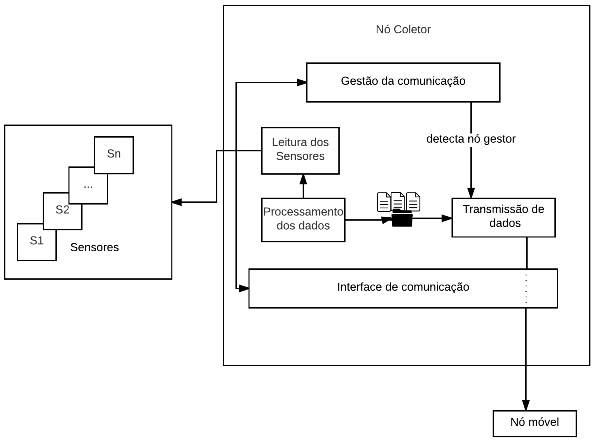

Fig. 2. Software diagram of collector node

B. Network architecture

The nodes form an ad-hoc network with the name ”dsensor”, and use static IPs in the 10.0.0.0/24 range. In this way the nodes only communicate with each other, reducing the security risks. In the wireless interface of each of the intervening nodes in the network runs the IBR-DTN protocol.

1) IBR-DTN Protocol Implementation: Along with the ad-hoc network configured in all network devices, they run an implementation of the DTN protocol developed by the IBR-Institut fr Betriebssysteme und Rechnerverbund, and so-called IBR-DTN. In order to be able to use it, a daemon is installed and later configured according to the user's preference. The name of the node in question is defined, which has the form dtn://nodeName.dtn, through which it can be found by the remaining nodes of the network. The base protocol used in this DTN, the Bundle Protocol, requires a Convergence-Layer Adapter, as referred to in the II-B section. In this specific case, TCP-CL is used, which uses TCP to send and receive the bundles. A TCP connection between the nodes is established, and a header that defines the connection parameters of TCP-CL is later exchanged. The identifier of the endpoint is also sent, which in this case will be the node identifier, which is the name itself. After this configuration the nodes can exchange bundles with each other [6]. Given that one of the nodes is constantly moving, ”discovery beacons” are sent every 5 seconds so that each node announces itself to new potential neighbors, and the connection has a keep-alive timeout of 10 seconds. Among the various possible routing strategies mentioned in the II-B section, the default routing option was chosen, which delivers the bundles to the neighbors and to the static nodes. This way we can define which nodes can connect to the network, and receive bundles. The definition of the static nodes is made on the configuration file and the first line of each static connection refers to the IP address of the node, followed

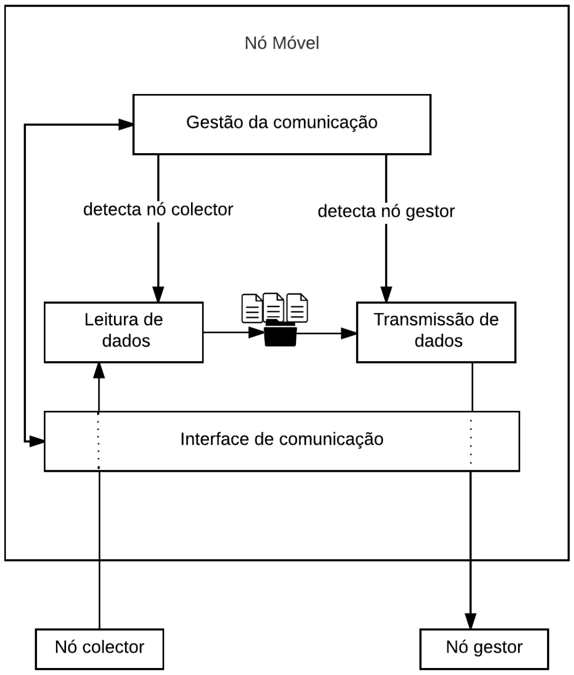

Fig. 3. Software diagram of mobile node

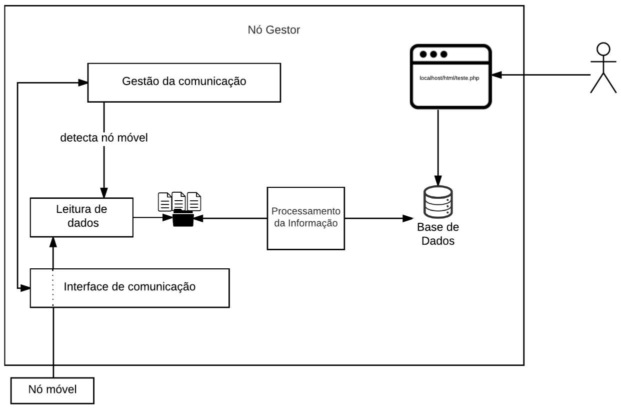

Fig. 4. Software diagram of manager node

by the port that receives the bundles, in this case 4556, which refers to the default port for DTN communication, the name of the node, the protocol used and finally the flags that allow immediate connection to the node without an internet connection.

IV. IMPLEMENTATION

Based on the architecture presented in the previous section, a test system was implemented to prove the feasibility of the solution. This section will present the details of the implementation, the decisions made and the results obtained.

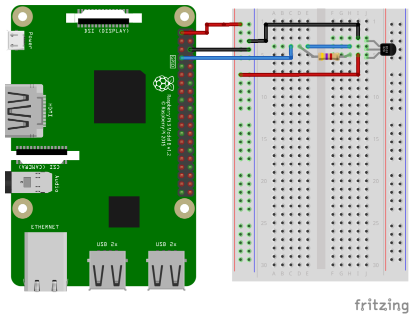

Fig. 5. Collector nodes hardware

A. *Hardware Architecture*

To test the proposed solution we used two collector nodes, a mobile node and a manager node. The collector nodes were simulated using Raspberry Pi 3 micro-controllers with Raspbian Jessie operating system, and temperature and humidity sensors connected through the device's GPIO interface. To simulate the mobile nodes and manager were used laptop computers with the Ubuntu 15.10 operating system. For the network to work properly, all nodes require a Wi-Fi module configurable in ad-hoc mode. The manager node has a mySQL database, a local Apache server, and PHP installed so that it can present the data correctly. For reasons of processor compatibility, Raspberry Pi 3 were used, since they have an ARMv7 processor that does not present any problems with the use of the IBR-DTN protocol, which is not the case with the model 1B.

1) Collector node: The collector node, being the one responsible for collecting environmental data, has a fixed position known to the user. Two collector nodes with different positions were used, and with different sensors. Collector node 1, with identifier node1, collects the temperature. Figure 5 shows the wiring diagram of the temperature sensor used in this node.

The second collector node, node2, has a similar assembly but for a sensor that measures the humidity of the environment. To obtain data close to a real situation, the collector nodes were placed at a distance of approximately 135m, as can be seen from figure 6. Although these nodes, represented in red, do not communicate with each other, this distance guarantees that there is disconnection between both collectors and the mobile node during its movement.

2) Mobile node: The mobile node, with identifier node253 and represented in green in the figure 6, was simulated by a laptop for reasons of simplification in terms of visualization of the connections in real time. A Raspberry Pi similar to the ones used in the collector nodes could have been used, but a battery would be required to feed it. This node makes the route between the collector nodes along the path indicated in the figure 6. Its starting point was next to node2, followed to node1, and made the reverse route to connect to the manager node, located at the starting point.

3) Manager node: The manager node, node3, represented in blue in figure 6, is also represented by a portable computer, located at the same place as node2. Like collector nodes, too

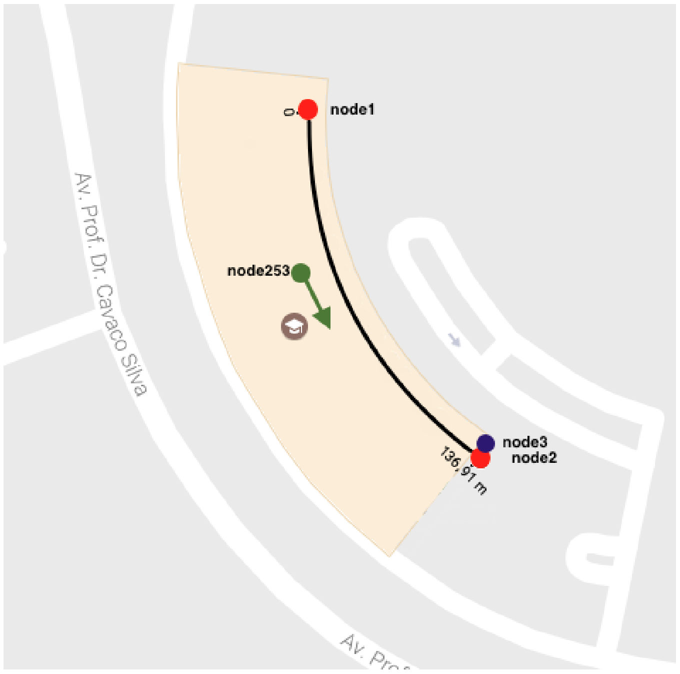

Fig. 6. Nodes location

it has a fixed and known location, since it is the node used by the user to consult the collected data.

B. *Nodes configuration*

The configuration files of the nodes at the level of the IBR-DTN protocol, referenced in section III-B.1 are similar for all nodes of the network, changing only the identifier of the node being configured and connections to the rest of the nodes. The collector nodes and manager only connect to the mobile node, node253, and node253 connects to all since it is the one who makes the reception and delivery of the data between both types of node.

The acquisition of data from the sensors connected to the collector nodes was done using programming language C, using the specific libraries for access and control of the GPIO pins of Raspberry Pi. This allows the reading of the data collected by the sensor, which are written in a new line of the file that will be sent when connecting to the mobile node.

The manager node, upon receiving the files from each of the collector nodes, reads its contents and inserts the rows into a local mySQL database. This database contains the same fields as the file created by the collector nodes, and through a local PHP page the data is made available to the user, as shown in table I. These data were acquired at one-minute intervals, and at the moment we can verify that the moisture detected by node2 was 0%, and in node1 the temperature was close to 30C. Given that the arrival of the files to the manager node does not happen periodically, since it is dependent on the route of the mobile node, the files already present in the manager node are read every minute and if there is any change, the new lines are added to the database.

TABLE I
COLLECTED DATA EXAMPLE

Date	Time	Node	Temperature	Humidity
2017-04-18	18:09:12	Node2	29.184	10.03
2017-04-18	18:10:12	Node2	29.234	10.01
2017-04-18	18:11:12	Node2	29.312	10.10
2017-04-18	18:12:12	Node2	29.417	10.07
2017-04-18	18:09:14	Node1	30.187	09.27
2017-04-18	18:10:14	Node1	30.375	09.81
2017-04-18	18:11:14	Node1	30.312	09.45

TABLE II
SIZE AND TIME OF FILE TRANSFER

Size (kB)	Time (s)
[0-100]	<1
[100-500]	1<t<2
[500-1000]	2<t<3
[1000-3000]	3<t<4
[3000-5000]	4<t<5
[5000-7000]	5<t<7
>7000	t>7

C. *Tests*

The implementation tests took place at the Instituto Superior Tcnico, campus Taguspark. The figure 6 shows the location of each of the nodes inside the building. In addition to the distance perceivable by the image, node1 is on floor 0 of the building, while node2 is on the second floor.

As the network nodes are not connected to the Internet, there may be a difference in the time of the various nodes. Taking into account the type of data collected, it is not considered problematic because the differences are of few minutes and the temperature or humidity, in this case, do not vary significantly in that time interval.

To perform the test, the system was simulated in its entirety. Collector nodes began collecting data and the mobile node began its movement. These tests were carried out for several days, at different times and periodicity of collection of the nodes themselves, and of communication between the mobile node and the different collectors. The data used to calculate the transmission time of the files were collected from the log files of the intervening nodes.

Tests were performed on files of varying sizes to evaluate the packet delivery rate and data transmission speed. The transfer of files of the same size was done 20 times, and the average value of transmission time calculated (table **??**).

The WiFi range of the nodes is about 25 meters, so at a speed of 2m/s the mobile node takes about 12 seconds to lose the connection.

According to the collected samples, a file with about 51 bytes takes a less than 1 second from the time it is queued for sending, until it reaches the target node. The time spent on the connection between the nodes is around 2 seconds, so there's about 10 seconds left to transfer a file.

The table II and the graph of the figure 7 show the relationship between the file size in kB and the transmission time of the file. Since logs only display the time in the HH: MM: SS format, it is not possible to calculate the transfer

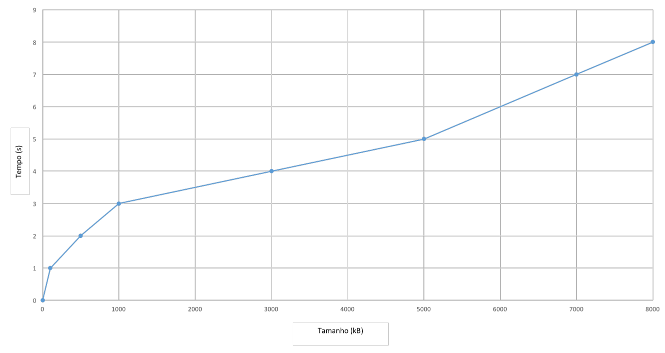

Fig. 7. Time/Size relation

time accurately for values less than one second. In this way, the file sizes as well as the transfer times were organized in intervals.

By analyzing the graphic of the figure 7, we can conclude that for files larger than 8000kB the sending time is close to the limit of the connection time between the nodes. Thus, taking as a limit a file of 8000kB we can collect about 82000 samples from 3 different sensors. This translates to approximately one sample every second for 24 hours.

For data whose change is not significant from second to second, there's no need to collect data at this interval. Samples collected every 10 minutes for 24 hours result in a file with about 8kB, which, according to the average times obtained in the **??** table, takes less than one second to transmit. In this way, we can guarantee a delivery rate of 100%, except in cases of connection interruption due to software or hardware problems.

V. CONCLUSIONS

The main goal of this work is the application of DTNs to the data collection of sensors, using drones. The research presented in this document, as well as the designed architecture, served as a basis for the application of this project in a functional and versatile way, using low-cost and open-source software. Although its realization has been made in controlled environment, its characteristics allow its application in different contexts, with changes of the used sensors, if it is necessary.

With the tests carried out on the implementation, it is proved the viability of the solution both in the scenario tested and in the hypothetical scenario that uses drones to move the mobile node.

The limitations found in this solution are related to the size of the transferred files, since the speed of the drone can be higher than the one reached in the tests, reducing the time that mobile node is in the range of collector nodes. This limitation can be mitigated by changing the periodicity of collecting data by the collector nodes and/or collecting more frequently by the mobile node.

In conclusion, the solution presented is a viable and low-cost option, applicable in the most varied contexts.

In the future, more functionality can be added to this system, as some features can be improved. The motion control of the drone can be incorporated into the network manager node through the data query page and a mobile application can also be developed for access to these functionalities. Another point that can be explored is the power supply of the collector nodes. Renewable energy can be used through solar panels to minimize the energy consumption of the nodes, allowing the number of nodes and sensors to be increased without causing a major impact on the cost of energy.

REFERENCES

[1] R. J. D'Souza and J. Jose, "Routing Approaches in Delay Tolerant Networks: A Survey," *International Journal of Computer Applications IJCA*, vol. 1, no. 17, pp. 9–15, 2010. [Online]. Available: http://www.ijcaonline.org/archives/volume1/number17/370-557

[2] J. Shen, S. Moh, and I. Chung, "Routing Protocols in Delay Tolerant Networks : A Comparative Survey," pp. 1577–1580, 2008.

[3] P. R. Pereira, A. Casaca, J. J. P. C. Rodrigues, V. N. G. J. Soares, J. Triay, and C. Cervello-Pastor, "From Delay-Tolerant Networks to Vehicular Delay-Tolerant Networks," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 1166–1182, 2012. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6007076

[4] E. Jones and P. Ward, "Routing strategies for delay-tolerant networks," *. . . to ACM Computer Communication Review (CCR . . .*, 2006. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Routing+Strategies+fo Tolerant+Networks#0

[5] K. Scott, "Bundle Protocol Specification," https://tools.ietf.org/html/rfc5050, [Accessed 28-Maio-2016].

[6] "DTN TCP Convergence-Layer Protocol," https://tools.ietf.org/html/rfc7242, [Accessed 5-Abril-2017].

[7] T. Instruments, "LM35 Precision Centigrade Temperature Sensors," http://www.ti.com/lit/ds/symlink/lm35.pdf, [Accessed 31-Maio-2016].

[8] "Raspberry Pi," https://www.raspberrypi.org, [Accessed 31-Maio-2016].

[9] "DJI," http://www.dji.com/, [Accessed 31-Maio-2016].