HotSel: A Hot Spot Selection Algorithm for Internet Access in Rural Areas through Nanosatellite Networks

Marco Cello, Mario Marchese and Fabio Patrone
Department of Electrical, Electronic, Telecommunications Engineering, and Naval Architecture (DITEN)
University of Genoa - Via All'Opera Pia, 13 - 16145 Genoa (Italy)
Email: marco.cello@unige.it, mario.marchese@unige.it, f.patrone@edu.unige.it

***Abstract*—Nowadays, to benefit from Internet access is so easy in some areas of the world as difficult in others. There are some different projects whose purpose is to extend Internet access to areas which do not have it yet. Each of these projects has unique characteristics that distinguish it from others (different transmission channels, different devices used, ...). One possible solution provides the use of a particular type of satellites called nanosatellites. Using these devices, we can have a solution to extend the network access in rural and remote areas which offers a good balance among performances, security and costs. In a network of this type ground stations (called hot spots) are deployed, which upload data destined to rural areas on nanosatellites and download data destined to Internet servers from nanosatellites. During a data connection, an Internet server that wants to reply to a request coming from a rural area has many hot spot alternatives to whom it can deliver data. The problem of choosing the "optimal" hot spot becomes important because a wrong choice could lead to a high delivery delay. We propose "HotSel": a hot spot selection algorithm able to minimize data delivery time. HotSel is simple and practical, and outperforms two other selection mechanisms used as comparison. All results have been collected by using Network Simulator 3 (NS3) and developing a module able to simulate a nanosatellite network.**

I. Introduction

Despite the technological progress gets used to the frequent emergence of new technologies and gives us new devices which implement new functionalities almost every day, there is a huge amount of world population (about 60%) that does not have access to the Internet since lives in countries or in remote areas where there is no ICT infrastructure. One of the main reasons is the cost needed to provide these areas with cables and communication infrastructures. These costs are often prohibitive compared with the yielded benefits. Another important reason is related to security and reliability, because, in many rural areas, undesirable situations which could damage telecommunication infrastructure occur quite frequently, such as natural disasters as earthquakes or political events as wars.

In literature, the problem to connect remote areas to the Internet has been principally tackled through the use of inexpensive Delay Tolerant Network (DTN) mobile devices [5], [14], [15], [17]. These architectures offer valid and cheap solutions, but suffer of severe performance limits due to the massive use of ground facilities, which include mechanical backhauls such as buses and cars to transport data [17]. Satellite communications [7], [10], [11], [13] are another way to provide Internet access in these areas, but current satellite technologies require high costs in the construction, launch and maintenance. Other solutions involve the use of a network of balloons travelling at an altitude of about 20 km (Google's Project Loon) [8], the use of drones, such as in the new Facebook's project called Internet.org [6], and the utilization of a huge number of LEO satellites in the SpaceX and partners' project [18] and in the Virgin, Qualcomm, and other partners' project called Oneweb [12].

Nanosatellites have been recently proposed as a cost-effective solution to extend the network access in rural and remote areas [1]. A good example is CubeSat [9], a kind of nanosatellite shaped as a 10 cm sided cube with a mass up to 1.33 kg. The cost needed to fabricate and launch a nanosatellite is about 0.1% of the cost needed to fabricate and launch a classical LEO communication satellite. In this solution rural and/or remote areas are connected through local gateways that communicate in an opportunistic fashion with the nanosatellite constellation by using the DTN paradigm [2], [4].

We have decided to use nanosatellites in our scenario because their cost is lower than other satellites one, and because they offer better performance than other solutions which provide the use of fix or mobile ground infrastructure. However, HotSel can be used in a generic satellite network, independently from the satellite type.

II. Use case scenario

Figure 1 shows a nanosatellites/DTN network scenario. There are different type of nodes:

- **Rural nodes**: $R_1, \ldots, R_N, R_{N+1}, \ldots, R_M$ represent the users situated in rural and remote areas who would like to have access to the Internet.
- **Cold Spots**: CS_1 and CS_2 are the rural ground stations that, on one hand, collect all data from rural nodes in order to upload them on nanosatellites, and, on the other hand, download data destined to rural users from nanosatellites.
- **Hot Spots**: HS_1 and HS_2 are the Internet ground stations that, on one hand, download data from nanosatellites and forward them to the nodes on the Internet (e.g. the Internet servers), and, on the other hand, upload data on nanosatellites which bring them to rural nodes.

978-1-4799-5952-5/15/$31.00 ©2015 IEEE

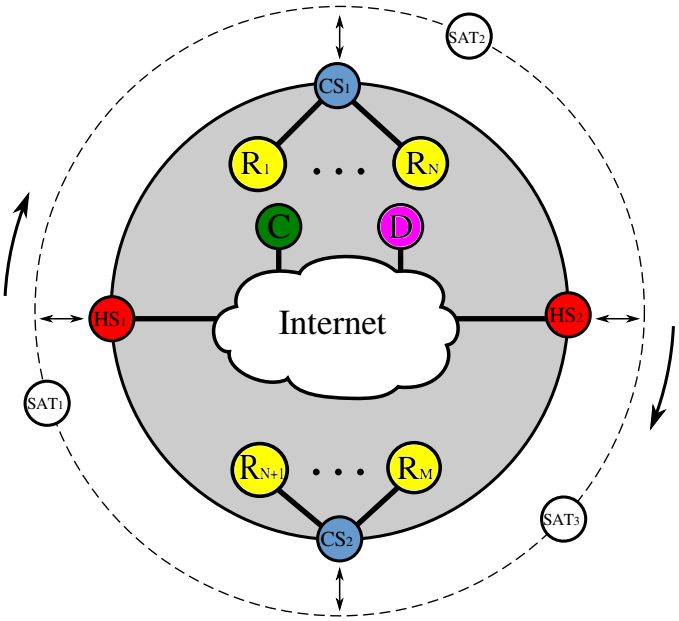

Figure 1: Nanosatellite network scenario.

- **Nanosatellites**: SAT_1, SAT_2, and SAT_3 are the nanosatellites which upload and download data from ground stations (both hot and cold spots). They change their position along their orbit;
- **Internet nodes**: D is an Internet node (e.g. Web Server, Mail Server, ...) which acts as the destination of all rural nodes requests. It represents an Internet Server.
- **Central node**: C is the central node of the DTN-based nanosatellites network whose purpose is to receive all rural nodes requests in order to collect all responses and to forward each of them to the proper hot spot.

When a rural user R_1 wants to browse the Internet or connect to a mail server, it sends its request to the cold spot that manages the remote region where it is situated (CS_1). CS_1 waits the next nanosatellite passage (SAT_1 in the example in Figure 1). If the nanosatellite buffer is not full, it accepts and transports the request. When it comes in contact with the first hot spot on its route (HS_2), it delivers the request to HS_2 that forwards the message to the central node C. C reads the message to know which is the destination node (e.g. D) and delivers the request to it. On the reverse path, D sends back the response to the central node. C selects one of the hot spots under its control and forwards the response to it. The selected hot spot uploads the message on a nanosatellite that delivers it to the proper cold spot which finally delivers the response to the rural node.

Differently from classical satellite networks, the communication path between rural nodes and the central node is a disconnected path, because all links are not up at the same time. This may be considered a structural constraint. The DTN architecture provides long term information storage on intermediate nodes so tackling link disruptions, very long delays, and intermittent connectivity. The action is carried out through an overlay protocol, called Bundle Protocol (BP) [16], developed on top of either transport (such as TCP and UDP) and lower layer (such as Bluetooth and Ethernet) protocols. BP data unit is called "bundle". It is a message that encapsulates application layer protocol data units. The DTN paradigm and BP are employed just to cope with this problem.

The central node may have many hot spot alternatives to whom it can deliver response data. Figure 1 shows two possible choices: HS_1 and HS_2. Different hot spots can send data to rural destinations with different delivery delay depending on the number, position and buffer occupancy of the nanosatellites they will come in contact with. The problem of choosing the "best" hot spot becomes important because the choice will impact on the delivery delay.

III. Hot Spot Selection

In this section we describe "HotSel": a dynamic hot spot selection method implemented in the central node C. HotSel computes the optimal hot spot choice to minimize the delivery time of each bundle destined to the rural users. To do this, the central node needs to know, in each time instant, the current position of the nanosatellites belonging to the orbit under its control[1], and how hot spots and nanosatellites buffer occupancy will evolve[2], in order to predict how much data will be loaded on each satellite and when the new bundle will be served. In particular, at each bundle arrival, for each "candidate" hot spot, the method computes the number of nanosatellites necessary to upload the queued bundles and the new arrived one. To fulfil this aim, it analyses each nanosatellite in order to estimate the nanosatellite buffer occupancy evolution in the near future.

Input: B_J - bundle destined to CS_J; $\mathcal{HS}, \mathcal{CS}, \mathcal{SAT}$ - set of hot spots, cold spots and nanosatellites, respectively;
Output: i^o - HS_i that minimizes the delivery time of bundle B_J;

1 **foreach** hot spot $i \in \mathcal{HS}$ **do**
2 $\quad d_{i,J} \leftarrow d_{i,J} + 1$;
3 $\quad S_i \leftarrow ComputeNumberNanosatellites\ (d_{i,J})$;
4 $\quad D_{i,J} \leftarrow w_{i,k} + (S_i - 1)\frac{W}{N} + t_{i,J}$;
5 $i^o = \underset{i \in \mathcal{HS}}{\operatorname{argmin}}\, D_{i,J}$;

Figure 2: HotSel implementation

HotSel implementation is shown in Figure 2. A bundle B_J needs to be transmitted to the $CS_{J \in \mathcal{CS}}$. For each HS_i, HotSel computes the delivery time $D_{i,J}$ needed to transmit B_J by using HS_i. The amount of data queued on HS_i is updated considering the new arrived bundle B_J ($d_{i,J}$+1). Then the function $ComputeNumberNanosatellites$ computes the number S_i of nanosatellites on the constellation that HS_i uses to upload the queued data destined to CS_J ($d_{i,J}$). Delivery time $D_{i,J}$ is computed as:

$$D_{i,J} = w_{i,k} + (S_i - 1)\frac{W}{N} + t_{i,J}\,, \tag{1}$$

[1] it is able to do this because it knows the nanosatellites initial position and their altitude, hence it knows how they are moving

[2] through a proper protocol which allows nanosatellites to send their buffer occupancy information to the hot spots whenever they come in contact with each other

1 **function** $ComputeNumberNanosatellites\ (d_{i,J})$;
2 nanosatellite $k \leftarrow$ first nanosatellite comes in contact with HS_i;
3 **repeat**
4 $nextHS(k)$: next HS on the path of SAT_k;
5 $\mathcal{CS}_k$: subset of $\mathcal{CS}$ on the path of SAT_k between $nextHS(k)-1$ and $nextHS(k)$;
6 $UploadDataOnSAT\ (k, nextHS(k), \mathcal{CS}_k)$;
7 $S_i \leftarrow S_i + 1$;
8 $k \leftarrow k - 1$;
9 **until** $d_{iJ} = 0$;
10 return S_i;

Figure 3: Function ComputeNumberNanosatellites

1 **function** $UploadDataOnSAT\ (k, nextHS(k), \mathcal{CS}_k)$;
2 **if** $nextHS(k) = i$ **then**
3 **if** $\mathcal{CS}_k \neq \emptyset$ **then**
4 $d_{k,j} = 0\ \forall j \in \mathcal{CS}_k$;
5 $calculate\ p_{i,j,k}\ using\ Eqs.\ (3)-(5)$;
6 $d_{i,j} \leftarrow d_{i,j} - p_{i,j,k},\ \forall j \in \mathcal{CS}$;
7 $d_{k,j} \leftarrow d_{k,j} + p_{i,j,k},\ \forall j \in \mathcal{CS}$;
8 **else**
9 **if** $\mathcal{CS}_k \neq \emptyset$ **then**
10 $d_{k,j} = 0\ \forall j \in \mathcal{CS}_k$;
11 $calculate\ p_{l,j,k}\ using\ Eqs.\ (8)-(10)$;
12 $d_{l,j} \leftarrow d_{l,j} - p_{l,j,k},\ \forall j \in \mathcal{CS}$;
13 $d_{k,j} \leftarrow d_{k,j} + p_{l,j,k},\ \forall j \in \mathcal{CS}$;
14
15 $nextHS(k) \leftarrow nextHS(k) + 1$;
16 $\mathcal{CS}_k \leftarrow$ subset of $\mathcal{CS}$ on the path of SAT_k between $nextHS(k)-1$ and $nextHS(k)$;
17 $UploadDataOnSAT\ (k, nextHS(k), \mathcal{CS}_k)$;

Figure 4: Function UploadDataOnSAT

where $w_{i,k}$ is the flight time from the current position to HS_i of the first SAT_k that will come in contact with i. $\frac{W}{N}$ is the average flight time between two nanosatellites. $t_{i,J}$ is the flight time of each SAT between its contact with HS_i and CS_J. HotSel iterates on all hot spots. The optimal HS_i^o that minimizes the delivery time for bundle B_J is:

$$i^o = \underset{i \in \mathcal{HS}}{\operatorname{argmin}}\, D_{i,J}. \tag{2}$$

The implementation of the function $ComputeNumberNanosatellites$ is reported in Figure 3. SAT_k is selected as the first nanosatellite that will come in contact with HS_i (Line 2). The function enters in a do-while loop in which SAT_k is virtually moved on its path until it comes in contact with HS_i (by using the function $UploadDataOnSAT$ shown in Figure 4). During the virtual movement, if SAT_k comes in contact with other hot spots and cold spots, the relative queues will be updated according to the data stored in the nodes. Lines 4 and 5 in Figure 3 define two variables: $nextHS(k)$, which represents the next hot spot with which SAT_k will come in contact, and $\mathcal{CS}_k$, which is the set of cold spots that are located between the two hot spots where SAT_k is currently located. We indicate with $nextHS(k)-1$ the hot spot before $nextHS(k)$ on the clockwise orbit path. Referring to Figure 1 and assuming SAT_k as SAT_3: $nextHS(k) = HS_1$, $\mathcal{CS}_k = CS_2$ and $nextHS(k)-1 = HS_2$. When SAT_k has virtually received the data from HS_i and from the other hot and cold spots, the variable S_i is incremented by 1 and the next nanosatellite is analysed ($k \leftarrow k-1$ means that the nanosatellite behind k is analysed). The loop is terminated when all the bundles queued in HS_i and destined to CS_J have been virtually uploaded on S_i nanosatellites ($d_{i,J} = 0$). The objective of function $UploadDataOnSAT$ is simulating the movement of k along its path having SAT_k and HS_i as input.

In case the next hot spot with which SAT_k comes in contact is exactly the analysed HS_i (as SAT_1 in Figure 1 with HS_1), lines 3-7 in Figure 4 are processed. If there is at least one cold spot in the orbit portion in which SAT_k is located ($\mathcal{CS}_k \neq \emptyset$), SAT_k downloads all the data destined to those cold spots and then sets $d_{k,j} = 0,\ \forall j \in \mathcal{CS}_k$. $d_{k,j} = 0$, means that, in each orbit, each SAT_k cannot carry more than Q bundles and can download to the destination cold spot only this amount of data per orbit. The bundles in excess would remain in the buffer at least for an entire orbit.

After this operation, SAT_k is virtually moved until HS_i comes in contact with it. HotSel calculates which data the chosen HS_i will upload on SAT_k as follows:

$$\begin{aligned} &p_{i,j,k}, \qquad \forall j \in \mathcal{CS} \\ &s.t. \end{aligned} \tag{3}$$

$$p_{i,j,k} \leq \min[d_{i,j}, Q - d_{k,j}], \tag{4}$$

$$\sum_{\forall j \in \mathcal{CS}} p_{i,j,k} \leq Q; \tag{5}$$

$d_{k,j}$ is the amount of data already stored on SAT_k and destined to CS_j. $p_{i,j,k}$ is the amount of data that HS_i uploads on SAT_k constrained to: 1) HS_i cannot upload more data than those it has stored and it cannot upload more than Q–$d_{k,j}$ data destined to CS_j. In this way SAT_k will not carry more than Q data destined to CS_j and it will empty the buffer dedicated to CS_j (Eq. 4). 2) The total amount of data that HS_i uploads to SAT_k is bounded by Q (Eq. 5). Finally, hot spot and nanosatellite buffer occupancies are updated:

$$d_{i,j} \leftarrow d_{i,j} - p_{i,j,k}, \qquad \forall j \in \mathcal{CS} \tag{6}$$

$$d_{k,j} \leftarrow d_{k,j} + p_{i,j,k}, \qquad \forall j \in \mathcal{CS}. \tag{7}$$

$UploadDataOnSAT$ terminates and variable S_i is incremented by one. If $d_{i,J} \neq 0$ after the update of Eq. 6, the procedure is repeated from Line 4 until $d_{i,J} = 0$.

Alternatively, if the next hot spot with which SAT_k comes in contact is not HS_i, as SAT_2 in Figure 1 with HS_1 (Lines 8-17 in Figure 4), HotSel proceeds making the calculations described before, but considering the previous hot spot in the orbit path, called HS_l. In particular:

$$\begin{aligned} &p_{l,j,k},\ \forall j \in \mathcal{CS} \\ &s.t. \end{aligned} \tag{8}$$

$$p_{l,j,k} \leq \min[d_{l,j}, Q - d_{k,j}], \tag{9}$$

$$\sum_{\forall j \in \mathcal{CS}} p_{l,j,k} \leq Q; \tag{10}$$

The equations above are derived from Eq. (3), (4) and (5) with the substitution of i with l. Then HS_l and SAT_k buffer

occupancy are updated:

$$d_{l,j} \leftarrow d_{l,j} - p_{l,j,k}, \qquad \forall j \in \mathcal{CS} \qquad (11)$$
$$d_{k,j} \leftarrow d_{k,j} + p_{l,j,k}, \qquad \forall j \in \mathcal{CS}. \qquad (12)$$

Finally, SAT_k is virtually moved on the next orbit portion (it would be moved after HS_2 in Figure 1): $nextHS(k)$ and $\mathcal{CS}_k$ are updated and $UploadDataOnSAT$ is called recursively.

To make this algorithm usable in any nanosatellite network topology is necessary that:

- the buffer of each hot spot has always enough free space to store the messages received from the central node;
- the buffer size of all nanosatellites is big enough in order to allow storing the maximum possible amount of data, corresponding to Q bundles, for each cold spot.

IV. Performance Analysis

Differently from [3], we have decided to implement HotSel in Network Simulator 3 (NS3) because it allows managing more functionalities than its predecessor NS2. We have developed a DTN NS3 module [3] which includes the following DTN characteristics:

- Delay and disruption tolerance: each DTN node is able to carry out a communication even though there is no persistent path between source and destination, storing each bundle to forward until the link with the next DTN hop is up and there is the possibility to transmit it;
- Heterogeneity: each DTN node is able to carry out a communication even though the nodes on the path between source and destination are not based on the same technologies and standards;
- Endpoints identification: each DTN node has an unique identifier which is used for routing.

This module provides also an implementation of bundle header, even though it is a shorter version than the one defined in [16]. The module implements our proposed hot spot selection algorithm.

We performed a set of tests by using two different scenarios:

1) **Scenario 1**: it is composed of 4 hot spots (HS_1-HS_4), 4 nanosatellites (SAT_1-SAT_4), 8 cold spots (CS_1-CS_8) and 2 rural nodes for each cold spot (R_1 and R_2 are linked to CS_1, R_3 and R_4 are linked to CS_2, ...). Its topology is shown in Figure 5.
2) **Scenario 2**: it is composed of 4 hot spots (HS_1-HS_4), 8 nanosatellites (SAT_1-SAT_8), 16 cold spots (CS_1-CS_{16}) and 2 rural nodes for each cold spot (R_1-R_{32}). Its topology is shown in Figure 6.

In both scenarios, all ground stations (both hot spots and cold spots) are equally spaced, and also the distance between two consecutive nanosatellites is constant, assuming that nanosatellites speed remains constant for the entire orbit. We define as orbit portion each part of the orbit between two consecutive hot spots.

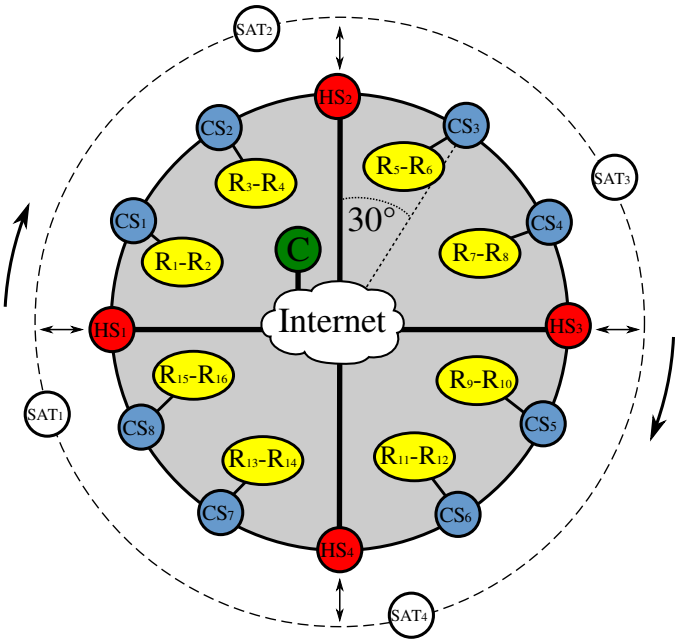

Figure 5: Scenario 1

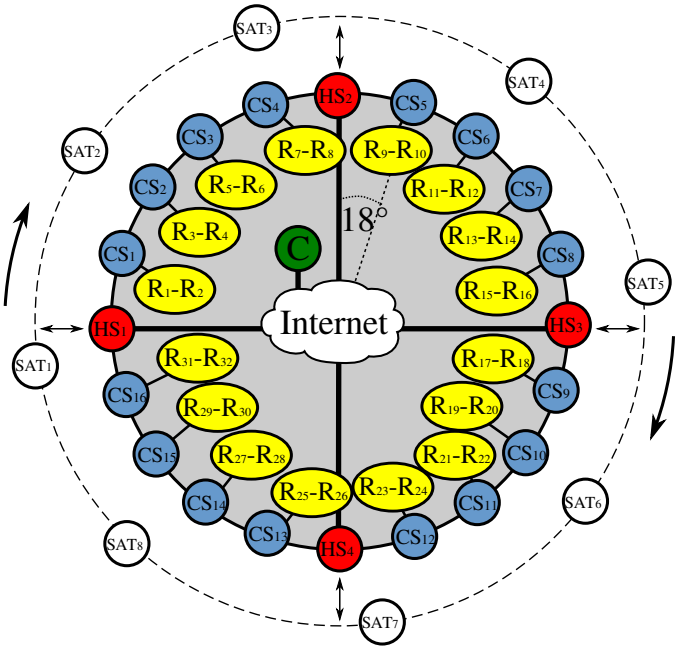

Figure 6: Scenario 2

For each scenario, we have made different simulations by varying the number of traffic flows, in order to test HotSel performance in different possible and realistic situations. Each simulated traffic flow has the central node as source node and a rural node as destination node, and it is composed by 500 bundles of 50 KB each. The following formalism has been

[3] the source code will be available soon on our website *http://www.scnl.diten.unige.it/software#NS3DTN*

used to define the rural nodes where the traffic is addressed: the notation R_x-R_y-R_z-R_t means that there are 4 traffic flows with R_x, R_y, R_z, and R_t as destination nodes. Using this module, we are able to set more realistic parameters about the nanosatellite orbit than ones in [3]. The nanosatellites altitude is 200 km, consequently the orbit time is about 90 minutes and the contact time is about 256 s. The transmission rate of satellite links is 230 Kbps, so ground stations can upload on nanosatellites and nanosatellites can download to ground stations about 7 MB of data in each contact (satellite links are full duplex). This amount is an underestimation, because we have set margin times at the beginning and end of contacts.

The parameter used to evaluate the performance is the Average Delivery Time (ADT), defined as:

$$ADT = \frac{\sum_{n=1}^{N} \left(T_n^{RX} - T_n^{TX}\right)}{N} \tag{13}$$

where N is the number of bundles per traffic flow (500 in our simulations), T_n^{RX} is the time instant when the $n-th$ bundle is received, and T_n^{TX} is the time instant when the $n-th$ bundle is transmitted.

For each simulation, we computed ADT by using three different mechanisms for the hot spot selection:

- **HotSel**;
- **Static**: all bundles destined to a specific rural area are forwarded to a specific and fixed hot spot;
- **Random**: the hot spot choice is random.

To better quantify the performance improvement achievable by using HotSel, we have decided to test four different network load configurations for each scenario:

1) **One portion (1P)**: all traffic flow destination nodes are located in the same orbit portion. For Scenario 1 and Scenario 2, 1P means to use the configurations named R_1-R_3 and R_1-R_5, respectively.
2) **Two consecutive portions (2CP)**: all traffic flow destination nodes are located in two consecutive orbit portions. For Scenario 1 and Scenario 2, 2CP means to use the configurations named R_1-R_3-R_5-R_7 and R_1-R_5-R_9-R_{15}, respectively.
3) **Two not consecutive portions (2NCP)**: all traffic flow destination nodes are located in two not consecutive orbit portions (in our scenarios they are opposite portions). For Scenario 1 and Scenario 2, 2NCP means to use the configurations named R_1-R_3-R_9-R_{11} and R_1-R_5-R_{17}-R_{21}, respectively.
4) **All portions (AP)**: the traffic flow destination nodes are equally distributed among all orbit portions. For Scenario 1 and Scenario 2, AP means to use the configurations named R_1-R_5-R_9-R_{13} and R_1-R_9-R_{17}-R_{25}, respectively.

Scenario 1. Figure 7 shows that in all simulations, HotSel outperforms both Static and Random Choice selections. The performances achievable by using HotSel are very good for two reasons: 1) when there are not or there are few bundles stored in the hot spots and destined to the destination cold spot, HotSel chooses the "nearest" hot spot to the destination cold spot; 2) if there is a congestion situation at the "nearest" hot spot to the destination cold spot, HotSel can upload on nanosatellites bundles belonging to all traffic flows through all hot spots, so increasing the amount of data carried by each nanosatellite during each orbit. This obviously cannot be done both by Static and by Random Choice. The only exception is in simulation AP, where Static Choice offers the same performance of HotSel. The motivation is that when the congestion level of all hot spots is the same, HotSel always chooses the same hot spot of Static Choice, which is the "nearest" hot spot to the destination cold spot. The mean performance improvement by using HotSel compared to Static Choice in this scenario is about 18%, while compared to Random Choice is 23%.

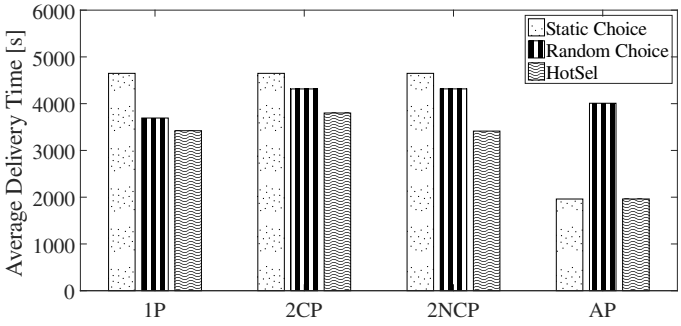

Figure 7: Average delivery time varying traffic flows configuration (Scenario 1).

Scenario 2. Figure 8 shows that HotSel offers better performance than the other two selection algorithms also in this scenario, with the same exception for simulation AP. The mean performance improvement by using HotSel compared to Static Choice in this scenario is about 10%, while compared to Random Choice is 42%.

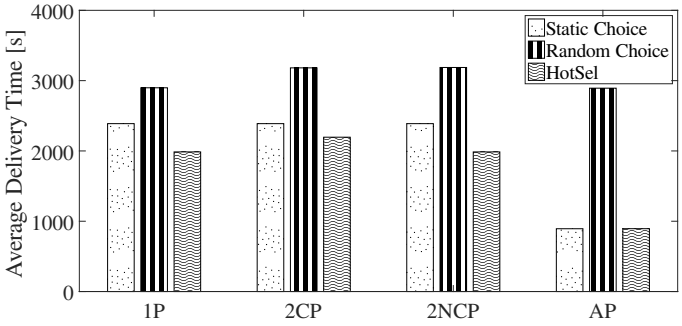

Figure 8: Average delivery time varying traffic flows configuration (Scenario 2).

In general, Random choice does not take the optimal choice for all bundles, and does not consider the presence of congestion situations, so it always offers worst performance than HotSel. Static Choice offers excellent performance when the destination nodes are equally distributed among the orbit portions (simulations AP) but it is inefficient in all other distribution cases. Obviously, the performance offered for all schemes also depends on the number of nanosatellites, because increasing this number, the distance (and consequently the flight time) between one nanosatellite and the following

decreases. This is clear comparing the performance in Scenario 1 (Figure 7) with the one in Scenario 2 (Figure 8).

Finally, we have modified the network topology adding some not DTN nodes in the Internet part of the network as background traffic sources, in order to simulate a more realistic network with possible congestion situations on links between the central node and hot spots. However, we have not reported the obtained results in this paper because they are exactly the same of the simulations without background traffic. The main reason of this behaviour is that the biggest part of delivery times concerns the time that bundles have to wait in the hot spots buffers until a nanosatellite uploads them, which remains the same regardless of the presence of other traffic flows.

V. Conclusions

In this paper we have presented HotSel, a hot spot selection algorithm able to minimize data delivery time in a nanosatellite-DTN rural access network. HotSel is implemented in the central node of a nanosatellite constellation and allows choosing which hot spot is the best to forward data destined to users located in remote areas. HotSel reduces bundles delivery times in all simulated scenarios and traffic flows configurations assuring a gain ranging between 0% and 27% compared with Static Choice, and between 7% and 70% compared with Random Choice.

Efficient multi central node architecture and relative load balancing will be studied in future in order to cope with the drawbacks issues related to the use of a single central node. Future research will be devoted also to carry on more complex simulations, by varying, for example, traffic distributions. Moreover, the DTN module will be extended in order to simulate a multi-orbit nanosatellite network, providing to nanosatellites the possibility to communicate with each other. A more deep and detailed description about the architectural aspects of the network and the communication among rural nodes and Internet serves will be included in a future work as not belonging to the main topic of this article.

References

[1] S. Burleigh. Nanosatellites for universal network access. In *Proceedings of the 2013 ACM MobiCom workshop on Lowest cost denominator networking for universal access*, pages 33–34. ACM, 2013.

[2] C. Caini, H. Cruickshank, S. Farrell, and M. Marchese. Delay- and disruption-tolerant networking (dtn): an alternative solution for future satellite networking applications. *Proceedings of the IEEE*, 99(11):1980–1997, 2011.

[3] M. Cello, M. Marchese, and F. Patrone. Hot spot selection in rural access nanosatellite networks. In *Proceedings of the 9th ACM MobiCom workshop on Challenged networks*, pages 69–72. ACM, 2014.

[4] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-tolerant networking architecture. *RFC4838, April*, 2007.

[5] A. Doria, M. Uden, and D. P. Pandey. Providing connectivity to the saami nomadic community. In *Proceedings of the 2nd Int. Conf. on Open Collaborative Design for Sustainable Innovation*, Dec 2002.

[6] Facebook and partners. Facebook and partner's project internet.org. http://www.internet.org/, 2014.

[7] Globalstar. Globalstar network. http://eu.globalstar.com/en/index.php?cid=3300, 1991.

[8] Google. Google project loon. http://www.google.com/loon/, 2013.

[9] H. Heidt, J. Puig-Suari, A. Moore, S. Nakasuka, and R. J. Twiggs. Cubesat: A new generation of picosatellite for education and industry low-cost space experimentation. In *Proceedings of 14th Annual/USU Conference on Small Satellites*, Aug 2000.

[10] Inmarsat. Inmarsat satellites. http://www.inmarsat.com/about-us/our-satellites, 1990.

[11] Iridium. Iridium global network. http://www.iridium.com/About/IridiumGlobalNetwork.aspx, 1998.

[12] OneWeb. Oneweb's project. http://www.oneweb.world/, 2015.

[13] Orbcomm. Orbcomm networks. http://www.orbcomm.com/networks, 1991.

[14] A. S. Pentland, R. Fletcher, and A. Hasson. Daknet: Rethinking connectivity in developing nations. *Computer*, 37(1):78–83, Jan. 2004.

[15] B. Raman and K. Chebrolu. Experiences in using wifi for rural internet in india. *Communications Magazine, IEEE*, 45(1):104–110, 2007.

[16] K. L. Scott and S. Burleigh. Bundle protocol specification. *RFC5050, November*, 2007.

[17] A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S. Keshav. Low-cost communication for rural internet kiosks using mechanical backhaul. In *Proc. of MobiCom '06*, pages 334–345. ACM, 2006.

[18] SpaceX and partners. Spacex and partner's project. http://www.spacex.com/, 2015.