Hot Spot Selection in Rural Access Nanosatellite Networks

Marco Cello
University of Genoa
Via all'Opera Pia 13, 16145
Genova, Italy
marco.cello@unige.it

Mario Marchese
University of Genoa
Via all'Opera Pia 13, 16145
Genova, Italy
mario.marchese@unige.it

Fabio Patrone
University of Genoa
Via all'Opera Pia 13, 16145
Genova, Italy
f.patrone@edu.unige.it

ABSTRACT

Nanosatellites architectures have been proposed as a cost-effective solution to extend the network access in rural and remote areas. To guarantee a reliable service and a large coverage, a good number of nanosatellites and ground stations (or hot spot) must be deployed. During a data connection, a server on the Internet that wants to reply to the user on 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 a high delivery delay. In this paper, we propose "HotSel": an hot spot selection mechanism able to minimize the delivery time. HotSel is simple and practical, and outperforms two other selection mechanisms used as comparison.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.3 [**Computer-Communication Networks**]: Network Operations

Keywords

Nanosatellite; Delay Tolerant Networking; Congestion Control; Next-Hop selection

1. INTRODUCTION

A large amount of world population has not access to Internet since lives in underdeveloped countries or in remote areas which do not possess ICT infrastructure. The costs needed to connect these areas using cables and classical infrastructures are prohibitive compared with the yielded benefits. In literature, the problem to connect remote areas to the Internet has been principally studied through the use of inexpensive DTN mobile devices with rural kiosks [9, 10]. These architectures offer valid and inexpensive solutions, but suffer of severe performance limits due to the massive use of ground facilities. On the other hand, satellite communications [7, 8] 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) [5], and the use of drones in the new Facebook project called Internet.org [4].

Nanosatellites [1] have been recently proposed as a cost-effective solution to extend the network access in rural and remote areas. CubeSat [6], a kind of nanosatellite (10 cm cube with a mass up to 1.33 kg), is fabricated and launched into low-earth orbit using 0.1% of the cost of a classical LEO communication satellite. Rural and/or disconnected area will be connected through local gateways that will communicate in an opportunistic fashion with the nanosatellite constellation using the Delay Tolerant Networking (DTN) paradigm [2, 3].

Figure 1 shows a nanosatellites/DTN network scenario: in a rural area, a group of users or nodes $R_1, \ldots, R_N$ is connected with the node CS_1. Nodes CS_1 and CS_2, referred in the following as "cold spots" (CSs) are located in remote areas and act as Internet gateway for users. They transmit and receive data with nanosatellites SAT_1, SAT_2, SAT_3 that will carry data and send them to nodes HS_1 and HS_2, referred in the following as "hot spots" (HSs) that are connected to Internet. Hot spots will send the requests to the central node C of the constellation that will open the communication with servers on the Internet (e.g. node D). On the reverse path, the central node C, that wants to reply to the user on rural area has many hot spot alternatives to whom it can deliver data. Different hot spots can send data to destination with different delivery delay depending on the number, position and buffer occupancy of satellites they come in contact with. The problem of choosing the "best" hot spot becomes important because a wrong choice could lead a high delivery delay to destination. To do this, in this paper we propose "HotSel": an hot spot selection mechanism able to minimize the delivery time.

2. HOT SPOT SELECTION

In this section we describe "HotSel": a dynamic hot spot selection method implemented in the central node C. HotSel is not an heuristic algorithm, but it computes the optimal hot spot choice to minimize the delivery time of each bundle (i.e. the PDU of the DTN protocol) destined to the rural users. To do this, the central node needs to know, in each time instant, the current position of the satellites belonging to the orbit that it manages, and how HSs and SATs buffer occupancy will evolve, in order to predict how much data

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CHANTS'14, September 7, 2014, Maui, Hawaii, USA.
Copyright 2014 ACM 978-1-4503-3071-8/14/09 ...$15.00.
http://dx.doi.org/10.1145/2645672.2645687 .

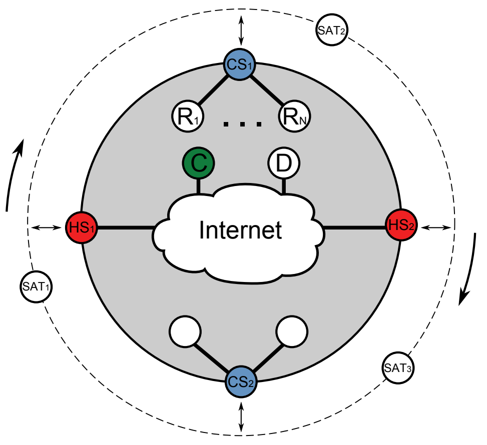


Figure 1: Nanosatellite network scenario.

will be loaded on each satellite and when the new bundle will be served. In particular, for each "candidate" HS, the method computes the number of satellites needed to upload the queued bundles and the new arrived bundle on satellites. To fulfill this aim, it analyzes each satellite one by one in order to estimate the satellite buffer occupancy evolution in the near future.

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 using HS i. The amount of data queued on i is updated considering the new arrived bundle B_J $(d_{i,J}+1)$; then the function $ComputeNumberSatellites$ computes the number S_i of satellites on the constellation that HS i uses to upload the data queued and destined to CS J $(d_{i,J})$. Delivery time $D_{i,J}$ is computed as:

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

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

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

The implementation of the function $ComputeNumberSatellites$ is reported in Figure 3. SAT k is selected as the first satellite that will come in contact with HS i (Line 2). The function enters in a do-while loop in which the satellite 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. Line 4 and 5 in Figure 3 define two variables: $nextHS(k)$, which represents the next HS with which the satellite k will come in contact and $\mathcal{CS}_k$, which is the set of cold spots that are located between the two HSs 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 i and from the others HSs and CSs, variable S_i is incremented by 1 and the next satellite is analyzed ($k \leftarrow k-1$ means that the satellite behind k is analyzed). The loop is terminated when all the bundles queued in i and destined to J have been virtually uploaded on S_i satellites ($d_{iJ} = 0$). The objective of function $UploadDataOnSAT$ is to take as input SAT k and HS i and simulate the movement of k along its path.

In case the next HS with which k comes in contact is exactly the analyzed 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 CS in the orbit portion in which k is located ($\mathcal{CS}_k \neq \emptyset$), SAT k downloads all the data destined to those CSs and then $d_{k,j} = 0$, $\forall j \in \mathcal{CS}_k$. $d_{k,j} = 0$, because in each orbit, each SAT k cannot carry more than Q bundles since it can download to the destination CS only this amount of data per orbit and the excess bundles 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. HotSel calculates which data the chosen HS i will upload to SAT k as follow:

$$\begin{aligned} & p_{i,j,k}, \qquad \forall j \in \mathcal{CS} && (3)\\ & s.t. && \\ & p_{i,j,k} \leq \min[d_{i,j}, Q - d_{k,j}], && (4)\\ & \sum_{\forall j \in \mathcal{CS}} p_{i,j,k} \leq Q; && (5) \end{aligned}$$

$d_{k,j}$ 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: -) 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); -) the total amount of data that HS i uploads to SAT k is bounded by Q (Eq. 5). Finally, HS and SAT buffer occupancies are updated:

$$\begin{aligned} d_{i,j} &\leftarrow d_{i,j} - p_{i,j,k}, && \forall j \in \mathcal{CS} && (6)\\ d_{k,j} &\leftarrow d_{k,j} + p_{i,j,k}, && \forall j \in \mathcal{CS}. && (7) \end{aligned}$$

$UploadDataOnSAT$ 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 next HS 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 HS in the orbit path, called HS l. In particular:

$$\begin{aligned} & p_{l,j,k}, \ \forall j \in \mathcal{CS} && (8)\\ & s.t. && \\ & p_{l,j,k} \leq \min[d_{l,j}, Q - d_{k,j}], && (9)\\ & \sum_{\forall j \in \mathcal{CS}} p_{l,j,k} \leq Q; && (10) \end{aligned}$$

The equations above are derived from Eq. (3), (4), (5) with the substitution of i with l. Then HS and SAT 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 (in Figure 1 would be moved after HS_2): $nextHS(k)$ 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 HS has always enough free space to store the messages received from central node; -) the buffer size of all SATs is big enough in order to allow storing the maximum possible amount of data, corresponding to Q bundles to be downloaded to each CS and Q bundles to be uploaded on SAT from each CS.

3. PERFORMANCE ANALYSIS

We implemented HotSel in Network Simulator 2 with the DTN module. We performed several simple simulations using two different scenarios: Scenario 1 is equal to that presented in Figure 1 with 2 hot spots (HS_1 and HS_2), 3 satellites (SAT_1, SAT_2 and SAT_3), 8 cold spots (CS_1-CS_4 located in the "north" portion of the orbit between HS_1 and HS_2, and CS_5-CS_8 located in the "south" portion between HS_2 and HS_1) and 2 rural nodes for each cold spot (N_1 and N_2 are linked to CS_1, N_3 and N_4 are linked to CS_2, ...). Scenario 2 is composed by 4 hot spots, 3 satellites and 8 cold spots.
In each scenario, different flows configurations are simulated. For each simulation we calculated the total delivery time using 3 different mechanisms for the hot spot selection: HotSel; static (all bundles destined to a specific CS are forwarded to specific fixed HS); and random (the HS choice is random). Each simulated traffic flow has central node as source node and a rural node as destination node. Each flow is composed of 500 bundles of 120 KB each. The orbit time of satellite is 675 s and the contact time is 32 s. In each contact, hot spots and cold spots can upload to satellite 7.4 MB of data. Each satellite can send the same data amount to the spots.
Scenario 1. Figure 5(a) shows that HotSel algorithm offers the same performance of the static choice when there is only one traffic flow (simulation N1), because all bundles are forwarded to the same HS in both cases, while using the random choice the performance decreases: in this case there is no advantage to using more HSs to upload data on SATs. Simulation N3-N13 refers to the case of two flows destined to nodes located in different orbit portions: node N_3 in the "north" and N_{13} in the "south". In the same way, the performance of static and HotSel are the same, since the optimal choice is statically assign HS_1 to the traffic destined to N_3 and HS_2 to the traffic destined to N_{13}. Simulation N2-N7, instead, refers to the case in which the two traffic flows are destined to nodes in the same orbit portion. In this case HotSel and random choice offer better performances than the static one, because both take advantage of the possibility to upload on SATs data destined to multiple CSs through all HSs. Last three simulations (N2-N4-N9-N11, N2-N5-N8-N15, N1-N3-N5-N7) are dedicated to the case with 4 traffic flows. In these cases HotSel performs better than the other two mechanisms. However, in all these simulations the performance differences between the HotSel and the Random method are not so significant (at most 4%).
Scenario 2. Increasing the number of hot spots we can observe a substantial performance improvement of HotSel algorithm as shown in Figure 5(b).
We chose to simulate only traffic that flows in uplink direction (from C to HSs to CSs) since only the bundles belonging to these traffic flows require the hot spot selection.

4. CONCLUSION

In this paper we present HotSel, an hot spot selection algorithm able to minimize data delivery time in a nanosatellite-DTN rural access networks. HotSel is implemented in the central node of the nanosatellite constellation and allows choosing which hot spot is the best to forward data destined to users. HotSel reduces the delivery time of the bundles in all the simulated traffic flow configurations (~15.6%). We compared the performance with two other simple mechanisms: static and random selection. Moreover, the results show that more the rural destination nodes are concentrated in adjacent areas, or more the HSs buffers are congested, higher the performance improvement. Load balancing and efficient central node architectures will be studied in future in order to cope with the bottleneck issues of central node C. Moreover more complex simulations, varying for example traffic distributions, will be the object of future work.

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

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

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

[6] 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.

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

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

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

[10] 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.

```
Input: $B_J$ - bundle destined to cold spot $J$; $\mathcal{HS}, \mathcal{CS}, \mathcal{SAT}$ - set of hot spots, cold spots and satellites, respectively;
Output: $i^o$ - hot spot $i$ that minimize the delivery time of bundle $B_J$;
foreach hot spot $i \in \mathcal{HS}$ do
    $d_{i,J} \leftarrow d_{i,J} + 1$;
    $S_i \leftarrow ComputeNumberSatellites\ (d_{i,J})$;
    $D_{i,J} \leftarrow w_{i,k} + (S_i - 1)\frac{W}{N} + t_{i,J}$;
$i^o = \operatorname{argmin}_{i \in \mathcal{HS}} D_{i,J}$;
```

Figure 2: HotSel implementation

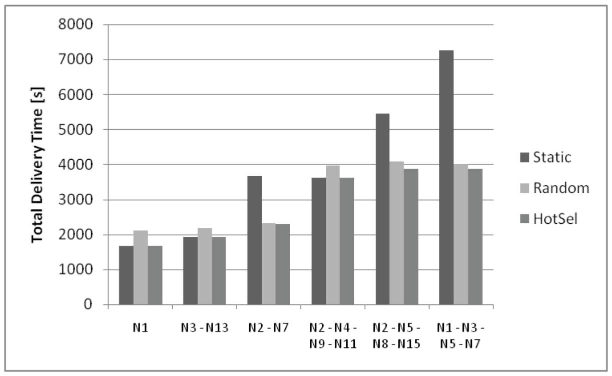
```
function $ComputeNumberSatellites\ (d_{i,J})$;
satellite $k \leftarrow$ first satellite comes in contact with $i$;
repeat
    $nextHS(k)$: next HS on the path of SAT $k$;
    $\mathcal{CS}_k$: subset of $\mathcal{CS}$ on the path of SAT $k$ between $nextHS(k) - 1$ and $nextHS(k)$ ;
    $UploadDataOnSAT\ (k, nextHS(k), \mathcal{CS}_k)$;
    $S_i \leftarrow S_i + 1$;
    $k \leftarrow k - 1$;
until $d_{iJ} = 0$ ;
return $S_i$;
```

Figure 3: Function ComputeNumberSatellites

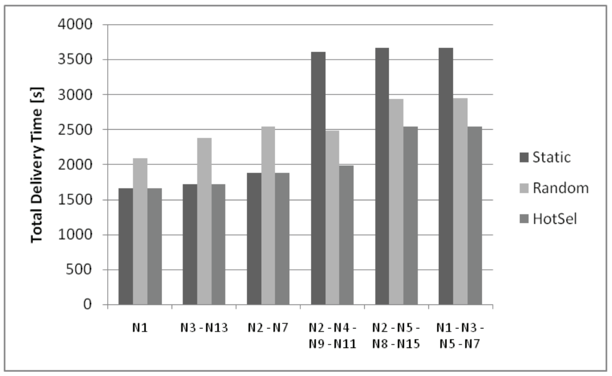
```
function $UploadDataOnSAT\ (k, nextHS(k), \mathcal{CS}_k)$;
if $nextHS(k) = i$ then
    if $\mathcal{CS}_k \neq \emptyset$ then
        $d_{k,j} = 0\ \forall j \in \mathcal{CS}_k$;
    $calculate\ p_{i,j,k}\ using\ Eqs.\ (3) - (5)$;
    $d_{i,j} \leftarrow d_{i,j} - p_{i,j,k},\ \forall j \in \mathcal{CS}$;
    $d_{k,j} \leftarrow d_{k,j} + p_{i,j,k},\ \forall j \in \mathcal{CS}$;
else
    if $\mathcal{CS}_k \neq \emptyset$ then
        $d_{k,j} = 0\ \forall j \in \mathcal{CS}_k$;
    $calculate\ p_{l,j,k}\ using\ Eqs.\ (8) - (10)$;
    $d_{l,j} \leftarrow d_{l,j} - p_{l,j,k},\ \forall j \in \mathcal{CS}$;
    $d_{k,j} \leftarrow d_{k,j} + p_{l,j,k},\ \forall j \in \mathcal{CS}$;

    $nextHS(k) \leftarrow nextHS(k) + 1$;
    $\mathcal{CS}_k \leftarrow$ subset of $\mathcal{CS}$ on the path of SAT $k$ between $nextHS(k) - 1$ and $nextHS(k)$ ;
    $UploadDataOnSAT\ (k, nextHS(k), \mathcal{CS}_k)$;
```

Figure 4: Function UploadDataOnSAT



(a) Scenario 1.



(b) Scenario 2.

Figure 5: Total delivery time varying the scenarios and traffic flows configuration.