Routing in Satellite Networks

Repo : https://github.com/pranavpage/routing-satcom

Pranav Page
Dept. of Electrical Engineering
IIT Bombay
Mumbai, India
18D070020

Kaustubh Bhargao
Dept. of Electrical Engineering
IIT Bombay
Mumbai, India
18D070014

Hrishikesh Baviskar
Dept. of Electrical Engineering
IIT Bombay
Mumbai, India
18D070048

***Abstract*—We study contemporary routing techniques in the context of Low Earth Orbit satellite constellations. Adaptive and distributed routing techniques are discussed, with our own claims on improvement of some of them in the aspects of congestion control and satellite failures. Implementation of a modified routing algorithm is presented**

***Index Terms*—LEO satellite communication, Distributed routing, Congestion control, Adaptive routing methods, paper survey, implementation**

I. Introduction to Satellite Communication

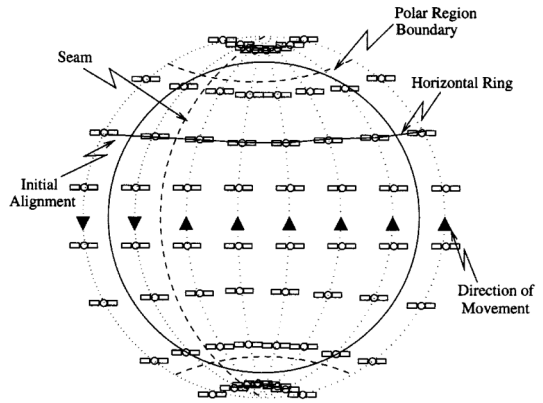

Fig. 1. Topology of the satellite constellation, showing orbital planes, the seam and the polar region boundary

The LEO satellites follow the Walker Constellation (parameters $M \times N/N/0°$). It consists of N polar orbits (planes) with M satellites in each, making a total number of $M \times N$ satellites. The Polar planes cross each other only at the poles and are separated from each other with $360°/(2 \times N)$ angular distance. Satellites in the same plane are equidistant with an angular distance of $360°/M$. The location of a satellite can be describes in two ways. i) Geographical location: of a saetllite S is given by $[longitude_S, latitude_S]$. ii) Logical locations: The logical location of a satellite S is given by $< p, s >$ where p $(p = 0, ...N-1)$ denotes the plane number and s $(s = 0, ..., M-1)$ denotes the satellite number. Thus, the location of a satellite relative to the network is completely specified by these two numbers. In practice, the polar orbits do not cross the pole

The link between intraplane satellites is called Intraplane Satellite Link, or Intraplane ISL. The intraplane ISLs are maintained at all times. Since these satllites are always at the same distance from their intraplane neighbours, the propagation delay on the intraplane ISLs is always fixed. The link between interplane satellites is called as Interplane Satellite Link, or Interplane ISL. The interplane ISL is shut down in the polar region and only kept operational in non-polar region. This is because of two reasons. First, satellites switch their positions when they cross the pole and the position map changes. And secondly, many satellites come in close proximity near the polar region and the ISL range changes rapidly. The common Round Trip Time for the LEO satellites at an altitude of 1000 km is 6.671 ms.

II. Problem Statement

We aim to propose a novel strategy for routing in satellite communication networks. To understand what challenges are faced by the same, we first consider what distinguishes routing in satellite communication networks from terrestrial, point-to-point routing. We first consider the problem of routing a packet from a **source** satellite to a **destination** satellite. The uplink-downlink is yet to be considered. For now, we assume that only a LEO satellite constellation is available, and we aim to look at routing strategies for multi-layer satellite constellations (which will include MEO and GEO satellites).
Terrestrial routing suffers from conditions on the ground, such as obstacles and/or geographical features which might prevent some lines of sight (in wireless communication). Satellite communication in LEO constellations suffers from no such problems, as even if an object blocks the satellite for a moment, the satellite moves very fast and will soon be visible again. Some drawbacks of using a satellite constellation are high transmission delay, high transmission loss, which leads to high BER, and the fast-changing topology of the nodes in the network relative to the ground. There is also an uneven distribution of dataflow, and some nodes might get clogged due to congestion. The satellites themselves have limited on-board processing and storage facilities, which might exacerbate the congestion and result in dropped packets if routing is done

improperly.
Due to the limited storage, we look at decentralized policies for routing, similar in structure to distributed terrestrial routing policies. The problem statement reduces to finding a distributed algorithm to find the shortest path from one node to another in a LEO satellite constellation.

III. Relevant Works

A. ATM-based Dynamic Routing (DT-DVTR)

This routing algorithm [1] is based on the Asynchronous Transfer Mode of operation of data transmission. This is a connection-oriented mode of transmission, with packets from the source node to the destination node following the same path (packet sequence is maintained). The authors propose that for meeting QoS requirements and dealing with the highly dynamic (yet predictable) topology of the satellite constellation, a connection-oriented routing scheme would perform well.
Discrete Time Dynamic Virtual Topology Routing makes use of the deterministic topology changes and generates a fixed number of topology slices, on which routing is performed. If the period of orbit is T, the topology can be considered as a repeating K length sequence, each slice separated by time $\Delta T = \frac{T}{K}$. If the total nodes in the constellation are N, there are $\binom{N}{2}$ source-destination pairs. Each pair gets assigned a set $P_w(k)$ of up to m distinct loopless paths. Each path $p(k) \in P_w(k)$ consists of a unique sequence of nodes $(i,j)_k$ such that $\delta^{p(k)}_{(i,j)_k} = 1$ iff $(i,j)_k$ is in path $p(k)$, 0 otherwise.
The problem reduces to finding the path $p^*(k)$ with the minimum cost $C_{p^*(k)} = \min_{p(k)} \sum_{i,j} c_{ij}(k)\delta^{p(k)}_{(i,j)_k}$. Selecting the paths in $P_w(k)$ is done by choosing the top m distinct shortest paths, using an iterative modified Dijkstra's algorithm, which eliminates nodes which are already included in some other path.
As this work did not focus on distributed computations, it contributed little to our development. However, the concept of discrete snapshots of the topology of the system is fascinating, and might prove to be useful.

B. Distributed Routing

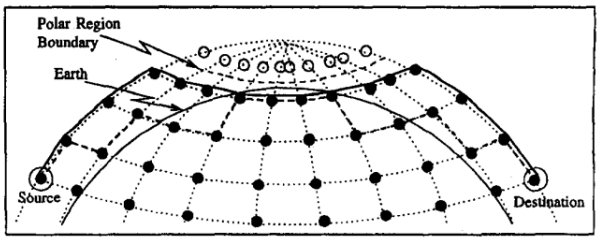

Fig. 2. Minimum propagation delay path not passing through the polar region

This algorithm [2] [3] involves setting up nodes of the constellation as virtual satellite locations. Thus, packets are routed between logical locations. The algorithm focuses on datagram routing, as described in [2] which is a connectionless framework. This has an immediate advantage over the ATM-based algorithm discussed in III-A in terms of path-handover, i.e , when a satellite passes out of visibility (below the horizon) and the ground station has to establish another connection with the new satellite. In ATM-based strategies, the routing has to be performed to re-establish a Virtual Circuit in order to send packets.
A node is represented by (p, s), where p is the plane number and s is the satellite number. Thus, p decides the longitude, while s decides the latitude. Each node has two neighbours in the same plane, and two in adjacent planes (only one if the node is located next to the seam, see I). The decision of which link to send the packet on is decided by the logical locations of the source and destination. This is based on the fact that closer to the poles, inter-plane ISLs are shorter than those at lower latitudes (see figure). The position of the two nodes with respect to the seam is also taken into consideration, with the path going around the Earth or through the polar region, based on the lengths of the two paths. The idea is that a decision map can be used to decide the next hop on every node, based solely on the logical locations of the current node and the destination.
The algorithm works in three phases, namely direction estimation, direction enhancement, and congestion resolution. Direction estimation does not look at lengths of the links, it works as a simple minimum number of hops algorithm. d_h, n_h and d_v, n_v are the direction and number of hops in horizontal and vertical directions respectively. These are calculated by the direction estimation phase.
The direction enhancement phase actually decides on which link to send the packet. The decision is made based on the minimum hop metrics calculated by the DE phase. The decision map of the satellite network can be precalculated and stored in every node. This phase also marks directions as primary or secondary. If all packets follow primary directions, then they are routed on the minimum propagation delay path. The congestion resolution phase does not get traffic info from neighbouring nodes. The decision has to be made on the basis of the size of the outgoing buffers on that link. A threshold χ is used to decide whether a link is congested. If the primary link is congested, packets are sent to the secondary link. If both are congested, then the packet is sent to the primary link.

C. Priority based Adaptive Routing

This algorithm [4] deals with making a choice between the multiple shortest paths available in the network. A priority mechanism is proposed, which takes into account the past utilization and buffering at each link. For example, one priority metric that can be used is $\mu = \alpha u_r + \beta l_q + \delta n_d$, where u_r is the utilization ratio of the link, l_q is the average queue length, n_d is the number of packets dropped per second. α, β, δ are design parameters that are to be tuned according to traffic requirements and network topology.
The information belonging to previous times should be

considered of less importance, and therefore, an aging mechanism is also defined as $\mu = \mu_0 \left(1 - \frac{t}{2t_a}\right) + \mu_n \frac{t}{2t_a}$, where μ_n is the priority metric calculated at time n.
This work gave insights into congestion control, and how to design priority metrics for the same.

D. Approximate Optimal Routing Algorithm

This algorithm [5] also involves topology snapshots, i.e, it is a time virtualization algorithm. It aims to reduce the computational complexity of routing using Dijkstra's algorithm.
The time interval between each adjacent time slice is $\Delta T = T/n$, where T is the orbital period of the satellite. The network topology remains unchanged for $t \in [k\Delta T, (k+1)\Delta T]$. Let the network topology be expressed as $G(k)$ and the correlation matrix be $X(k)$. The value of $c_{i,j}$ in the correlation matrix is ∞ if nodes i and j cannot communicate with each other. If there are M satellites, $G(k)$ has M nodes and $X(k)$ is of size $M \times M$, and Dijkstra's algorithm takes $\mathcal{O}(M^2)$ time. If there are N ground stations as well, the time complexity becomes $\mathcal{O}((N+M)^2)$. The idea is to use local topology information to reduce the size of $X(k)$. If nodes p and q want to communicate with each other, the algorithm first finds the midpoint k of the two nodes (in spherical geometry). If D_{kp} is the distance between k and p, then the algorithm searches for a gateway i in the range $D_{ki} \leq D_{kp} + r$ where r is the compensation of the search distance (can be increased from 0). If n such gateways are found and total m satellites are visible from these gateways, a $(n+m) \times (n+m)$ sized sub-correlation matrix can be established with values from $X(k)$. The algorithm proceeds by running Dijkstra's algorithm based on the sub-correlation matrix.
This work gives an idea about how to use local topology information to reduce computational complexity, but does not say anything about congestion control. It is also centralized.

E. Distributed Dynamic Programming

This seminal paper [6] discusses an asynchronous distributed algorithm for a broad class of dynamic programming problems. The class is described below.
Let S and C be two sets referring to state space and control space respectively. For each $x \in S$, we have $U(x) \subset C$. Let F be the set of all real-valued functions $J : S \to [-\infty, \infty]$. Let $H : S \times C \times F \to [-\infty, \infty]$ be a monotone mapping in J in the sense that $J_1 \leq J_2 \Rightarrow H(x,u,J_1) \leq H(x,u,J_2)$. Given a subset $\bar{F} \subset F$, the problem is to find $J^* \in \bar{F}$ such that $J^*(x) = \inf_{u \in U(x)} H(x,u,J^*)$.
For the shortest path problem, if $\mathcal{N}$ is the set of nodes and $\mathcal{L}$ is the set of links, $N(x)$ denotes neighbours of x, then the identifications $S = C = \mathcal{N}$, $U(x) = N(x)$, $\bar{F} = F$, $J^*(x) = d_x^*$, $H(x,u,J) = a_{xu} + J(u)$ transform the abstract problem to the shortest path problem. Here, a_{xu} is the length of the link from x to u. d_x^* is the shortest path distance to node 1 from node x.
The computation algorithm can be described as a process over n nodes that perform computations. At each instant, the node can be in one of three states : compute, transmit, or idle. In the compute stage, node x computes a new estimate of d_x^*. It transmits this new estimate to nodes in $N(x)$. Thus, each node has a buffer per neighbour, denoted as B_{ij}, where it stores the latest transmission from j, as well as a buffer B_{ii} where it stores its own estimate. The paper provides proof for the claim that as $t \to \infty$, the estimate stored in each node approaches the optimal solution.

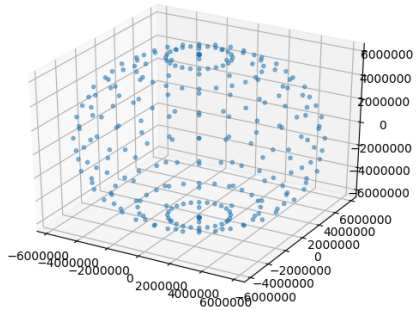

Fig. 3. Satellite Constellation with P = 12, N = 24

F. Miscellaneous

Some other works that were looked at, but did not show direct relevance to the topic are as follows: the Dynamic Source Routing algorithm [7], which is a connection-based routing scheme, and thus, suffers from the typical drawbacks such as path handover, initialisation time, etc. The paper [8] describes a probabilistic routing technique similar to the one that we plan to implement, but with a simpler priority metric. This might provide us with a benchmark to test our implementation against, apart from the traditional Dijkstra, Bellman-Ford and Darting algorithms.

IV. Implementation

We implemented the distributed routing algorithm as described in [3] in Python 3, aimed at looking to improve the algorithm. To actually study the performance of the algorithm, the constellation was to be made first. The code in which the constellation is built is included in the repository here.
The figure in 3 displays a Low Earth Orbit satellite constellation with P = 12 orbital planes, and N = 24 satellites per orbital plane. The orbital planes are equidistant from each other and span the complete 360^o, thus achieving a spacing of 15^o between adjacent planes. The constellation is located at an altitude of 600 km above the Earth's surface. Note that this constellation denotes an array of virtual nodes. The actual satellites would be moving all the time, but ignoring the handover and the displacement of the satellites between the nodes, the virtual constellation would be the one considered by us.
The figure 4 shows the constellation along with the direction

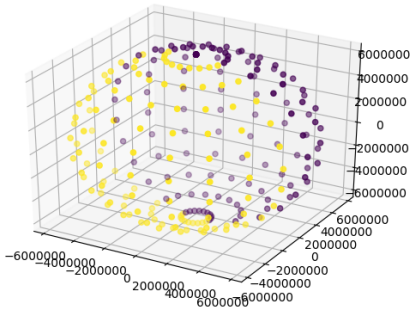

Fig. 4. Satellite Constellation with P = 12, N = 24 and with hemisphere division. The yellow nodes rotate North, while the purple nodes rotate South.

TABLE I
CALCULATION OF DIRECTIONS FOR $P^H_{S_c \to S_n}$

S_c & S_n in the Same Hemisphere					
ps_c ps_n			ss_c ss_n		
<	>	=	<	>	=
d_h=+1	d_h=-1	d_h=0	d_v=-1	d_v=+1	d_v=0

S_c & S_n in Different Hemispheres					
ps_c ps_n			$(M-1-ss_c)$		ss_n
<	>	=	<	>	=
d_h=-1	d_h=+1	-	d_v=+1	d_v=-1	d_v=0

Fig. 5. Direction calculation for P^H

of rotation. We also consider a polar region boundary at the latitude of 75^o. There are no inter-plane hops in the polar region.

The implementation of the distributed routing algorithm is done according to the papers [3] [2]. We explain in brief, using tables from the papers the steps in the algorithm.

A. Direction Estimation

This step considers the decision of where to send the packet on the next hop. Two different paths are considered, P^H and P^V, where P^H is the optimal among all paths which do not cross the polar regions, and P^V is the optimal path among all paths which do cross the polar regions. This phase chooses which path (P^H or P^V) would be best by comparing the number of hops in both. Each path can be described by four metrics, namely d_h, n_h, d_v, n_v where (d_h, n_h) describe the direction in the horizontal direction and the number of horizontal hops respectively.

Direction calculation for the two paths can be seen in 5 and 6. The number of hops for each path can be calculated by considering the position of the nodes on a case by case basis (e.g Eastern hemisphere and Western hemisphere), and is implemented in the code and described in the paper [3]. In the end, the two paths are compared, and the path with minimum $(n_v + n_h)$ is returned.

TABLE II
CALCULATION OF DIRECTIONS FOR $P^V_{S_c \to S_n}$

S_c & S_n in Eastern Hemisphere					
ps_c ps_n			$2(s_{S_n}+s_{S_c}+1)$		M
<	>	=	<	>	=
d_h=-1	d_h=+1	d_h=+1	d_v=+1	d_v=-1	d_v=+1

S_c & S_n in Western Hemisphere					
ps_c ps_n			$2(s_{S_c}+s_{S_n}+1)$		$3M$
<	>	=	<	>	=
d_h=-1	d_h=+1	d_h=+1	d_v=+1	d_v=-1	d_v=-1

S_c in Eastern & S_n in Western Hemisphere					
ps_c ps_n			$2(s_{S_n}-s_{S_c})$		M
<	>	=	<	>	=
d_h=+1	d_h=-1	d_h=0	d_v=-1	d_v=+1	d_v=+1

S_c in Western & S_n in Eastern Hemisphere					
ps_c ps_n			$2(s_{S_c}-s_{S_n})$		M
<	>	=	<	>	=
d_h=+1	d_h=-1	d_h=0	d_v=+1	d_v=-1	d_v=-1

Fig. 6. Direction calculation for P^V

B. Direction Enhancement

This phase actually takes into account the unequal interplane ISLs in the network, and the transport of packets in the polar region. In this phase, the directions given by d_h and d_v are labelled primary or secondary, depending on the position of the packet source and destination.

If the source is in the polar region, then d_v is marked as the primary direction. If d_v is also zero, then the destination must also be in the polar region. The packet is then routed to the nearest node outside the polar region, then horizontal hops are done.

If the source is in the last horizontal ring before the polar region, then d_h is marked as primary. Thus, horizontal hops are given priority, which is justified as the interplane ISLs are shorter in this last ring. If d_h is zero, then d_v is marked as the primary direction.

Here we differ slightly from the paper implementation. The paper recommends using a decision map to look up the next hop when the source is in the 'middle' of the mesh. A decision map would take up storage space which can be used by the satellite to increase its buffers and processing capability. Instead, we propose that the packet be routed on the basis of the relative difference between the latitudes of the source and destination, i.e, if the source is at a higher latitude, then hop horizontally till the packet reaches the orbital plane of the destination satellite. If not, then continue hopping vertically till the packet reaches the horizontal ring of the destination satellite.

C. Results

We implemented the above steps in Python 3 and generated packets with random start nodes and stop nodes. The code is designed to also generate logs in the terminal of packet hops (for debugging). All constellation parameters such as P, N,

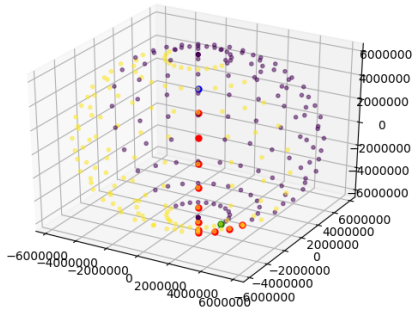

Fig. 7. Packet routing 0 : (11 , 12) → (8, 22)

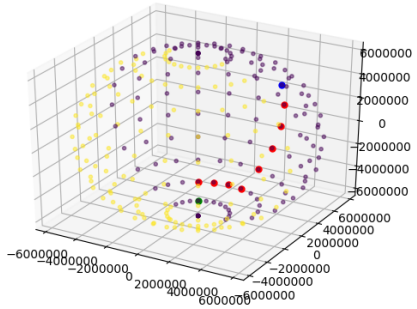

Fig. 8. Packet routing 1 : (8 , 12) → (5, 5)

etc. have been declared as global variables and can accordingly be changed. A sample output routing for a packet is given at the end of the report.

It is seen that routing using the simplified version of the algorithm also works. Figures 7, 8, 9, 10, 11 show different paths taken by different packets originating from different

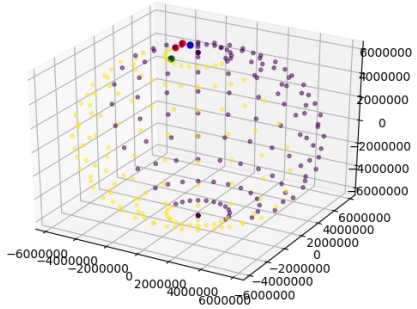

Fig. 9. Packet routing 2 : (9 , 5) → (9, 1)

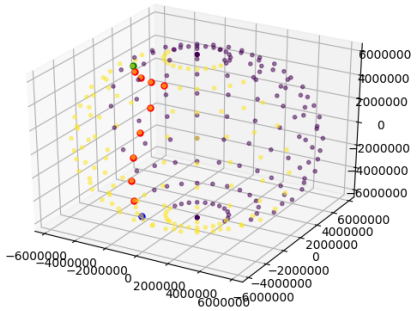

Fig. 10. Packet routing 3 : (1 , 22) → (6, 16)

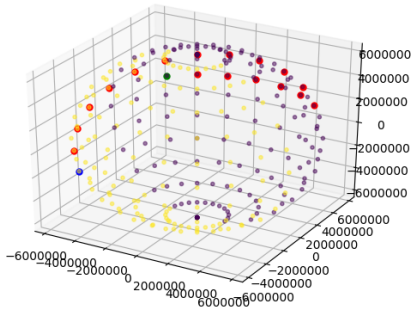

Fig. 11. Packet routing 4 : (10 , 5) → (3, 17)

regions. The green dot denoted the origin of the packet, and the blue dot denotes the destination of the packet. An interesting case is seen in Figure 10, where the origin of the packet is very close to the polar region, but not actually in it. Thus, horizontal hops are preferred to make use of the small interplane ISLs. Figure 11 also shows a case when going through a polar region is preferred over going around it. Figure 8 shows a case in which the origin is in the polar region, and thus only intraplane ISLs are permitted till the packet exits the polar region. Then the packet travels along the smallest possible horizontal ring till it reaches the orbital plane of the destination.

V. Future Work

Simulations could be done in ns-3 for actually comparing delay/throughput with contemporary routing algorithms. The problem of congestion control has not been addressed yet, so that could be another avenue to explore. The problem of link handover could also be potentially interesting in this context. Currently the simulation of the modified algorithm was run in Python 3, but to actually compare processing times with other algorithms like Dijkstra's or Bellman-Ford's, code optimization could be done, as well as a C implementation could be developed.

VI. Individual Contribution

Each of us have read most of the relevant papers. Pranav has focused on the distributed routing problem and possible solutions, included in III-B and III-E. Kaustubh has focused on the virtual topology snapshots-based algorithms, namely III-A and III-D. Hrishikesh has looked at priority metrics in III-C.

References

[1] M. Werner, C. Delucchi, H.-J. Vogel, G. Maral, and J.-J. De Ridder, "Atm-based routing in leo/meo satellite networks with intersatellite links," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 1, pp. 69–82, 1997.

[2] E. Ekici, I. Akyildiz, and M. Bender, "Datagram routing algorithm for leo satellite networks," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, vol. 2, 2000, pp. 500–508 vol.2.

[3] ——, "A distributed routing algorithm for datagram traffic in leo satellite networks," *IEEE/ACM Transactions on Networking*, vol. 9, no. 2, pp. 137–147, 2001.

[4] O. Korcak and F. Alagoz, "Analysis of priority-based adaptive routing in satellite networks," in *2005 2nd International Symposium on Wireless Communication Systems*, 2005, pp. 629–633.

[5] Y. Liu and L. Zhu, "A suboptimal routing algorithm for massive leo satellite networks," in *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, 2018, pp. 1–5.

[6] D. Bertsekas, "Distributed dynamic programming," *IEEE Transactions on Automatic Control*, vol. 27, no. 3, pp. 610–616, 1982.

[7] W. Peng, Y. Jian, C. Zhi-gang, and W. Jing-lin, "Dynamic source routing algorithm in low-earth orbit satellite constellation," in *2006 International Conference on Communication Technology*, 2006, pp. 1–4.

[8] X. Liu, Z. Jiang, C. Liu, S. He, C. Li, Y. Yang, and A. Men, "A low-complexity probabilistic routing algorithm for polar orbits satellite constellation networks," in *2015 IEEE/CIC International Conference on Communications in China (ICCC)*, 2015, pp. 1–5.

VII. Example output

```
testing (1 , 22) -> (6, 16)
dv=1, dh=1, nv=6, nh=5, hops=11, primary=[1, 0], secondary = [0, 1]

(2 , 22) -> (6, 16), hops = 1
dv=1, dh=1, nv=6, nh=4, hops=10, primary=[1, 0], secondary = [0, 1]

(3 , 22) -> (6, 16), hops = 2
dv=1, dh=1, nv=6, nh=3, hops=9, primary=[1, 0], secondary = [0, 1]

(4 , 22) -> (6, 16), hops = 3
dv=1, dh=1, nv=6, nh=2, hops=8, primary=[1, 0], secondary = [0, 1]

(5 , 22) -> (6, 16), hops = 4
dv=1, dh=1, nv=6, nh=1, hops=7, primary=[1, 0], secondary = [0, 1]

(6 , 22) -> (6, 16), hops = 5
dv=1, dh=0, nv=6, nh=0, hops=6, primary=[0, 1], secondary = [0, 0]

(6 , 21) -> (6, 16), hops = 6
dv=1, dh=0, nv=5, nh=0, hops=5, primary=[0, 1], secondary = [0, 0]

(6 , 20) -> (6, 16), hops = 7
dv=1, dh=0, nv=4, nh=0, hops=4, primary=[0, 1], secondary = [0, 0]

(6 , 19) -> (6, 16), hops = 8
dv=1, dh=0, nv=3, nh=0, hops=3, primary=[0, 1], secondary = [0, 0]

(6 , 18) -> (6, 16), hops = 9
dv=1, dh=0, nv=2, nh=0, hops=2, primary=[0, 1], secondary = [0, 0]

(6 , 17) -> (6, 16), hops = 10
dv=1, dh=0, nv=1, nh=0, hops=1, primary=[0, 1], secondar

(6 , 16) -> (6, 16), hops = 11
```