

Evaluation of Opportunistic Contact Graph Routing in Random Mobility Environments

Marc Sanchez Net
Jet Propulsion Laboratory,
California Institute of Technology
Pasadena, CA
marc.sanchez.net@jpl.nasa.gov

Scott Burleigh
Jet Propulsion Laboratory,
California Institute of Technology
Pasadena, CA
scott.c.burleigh@jpl.nasa.gov

Abstract—Routing in networks where nodes move randomly is particularly challenging due their potentially unpredictable, and rapidly changing topology. Several routing algorithms have been presented in the literature to address the needs of such networks, most of them implementing variants of controlled network flooding in the hope of successful data delivery.

In this note, we compare the results of previous routing algorithms with Opportunistic Contact Graph Routing (OCGR), an enhanced version of Contact Graph Routing (CGR) that is suitable for networks where contacts cannot always be scheduled ahead of time. To perform the benchmark, we simulate a network of nodes moving in a certain space according to the Random Waypoint Mobility Model, and then take measurements of bundle delivery probability and overhead ratio as metrics of performance and cost respectively. Through this exercise, we demonstrate that the performance of OCGR is highly dependent on the type of network under consideration (e.g. very sparse vs. densely connected) and the assumed mobility model.

Index Terms—delay tolerant networks, opportunistic routing

I. INTRODUCTION

Routing in Delay Tolerant Networks (DTN) is a challenging problem due to topology variability over time. Interplanetary networks such as the Deep Space Network (DSN) are an example of DTN where contacts suffer long outages (due to planet occultations, for instance) and large transmission distances. However, the existence, or lack thereof, of a link can be known ahead of time with a high level of confidence as motion in the system obeys the laws of orbital mechanics. On the other hand, ad-hoc networks might be restricted to operate in vicinities of a certain scale (e.g., room, city), but nodes can come in and out of contact at given point in time without prior warning. Therefore, routing protocols that successfully address the needs of the DSN might not be suitable for ad-hoc networks, and vice versa.

Given this fact, a natural question to ask is how to route data through a DTN with heterogeneous nodes where some of them move in a scheduled manner and others do not (e.g. a set of orbiters and rovers at Mars exchanging data with Earth). To address it, at least two high level strategies can be envisioned: First, conceive a routing algorithm sufficiently smart to operate in both the scheduled and non-scheduled part of the network. This ensures that all nodes are interoperable as a single routing mechanism is shared across all them. Alternatively, regions that encompass nodes with similar mobility can be first

defined, each one with its specific routing protocol, and then a global inter-region routing mechanism can be established to connect them through a set of gateways.

OCGR is, to the best of our knowledge, the first attempt to pursue the former strategy. It extends the capabilities of CGR, a routing algorithm originally designed and optimized to route in DTNs with scheduled contacts, to handle contacts that are predicted with a certain confidence degree or discovered in real-time. In the latter case, we assume that the underlying link layer can notify the routing layer when a new contact starts. Therefore, while the specific goal of this paper is to benchmark the performance of OCGR against other ad-hoc routing mechanisms, its strategic objective is to start informing which alternative is better suited to address the needs of heterogeneous DTNs: A single routing protocol capable of gracefully handling multiple mobility environments, or a set of specialized routing algorithms in regions that are then interconnected.

II. LITERATURE REVIEW

A. Use Cases for Opportunistic Routing

Several use cases in the space context would benefit from a delay/disruption tolerant networking with opportunistic contacts. For instance, human exploration of planetary bodies such as the Moon or Mars will require a mix of static and mobile elements generating different data flow types [1]. While the main base camp could be serviced by a lander acting as a cell tower, science traverses could extend beyond the coverage area, thus requiring delay-tolerant communications to be implemented. Similarly, robotic exploration of rugged terrain environments such as Moon caves could also require opportunistic routing of data as no infrastructure would be pre-deployed [2]. Finally, use cases for opportunistic routing on Earth are well documented in the literature (see, for instance, [3]) and include mobile sensor networks (Internet of Things), vehicular networks, ad-hoc military deployment networks.

B. Network Mobility Models

To evaluate the performance of routing protocols in ad-hoc mobile networks, simulation environments where nodes move according to a given mobility model are typical in the literature. Reference [4] provides a great survey of those

models and, at the highest level, classifies them depending on whether nodes move freely with respect to each other or they make correlated movement decisions. For the purposes of this paper, we focus our attention on the former, specifically the *Random Waypoint Mobility Model*, which is said to be suitable for environments where nodes advance based on targets that are known a priori (e.g. Mars rovers moving across an area of geological interest, people walking in a museum [4]), and has a small set of configuration parameters.

In the *Random Waypoint Mobility Model*, nodes are initially placed at random in a square of given dimensions. At time zero, a subset of them determines the position of their next target, while the rest select a wait time, all at random. When the simulation starts, the nodes that have selected a target start moving towards it in a straight line and given speed, while the other wait until their timers expire. When that happens, they also select a destination target and start moving towards it. Finally, when a node reaches its target, it waits for a random period of time and then selects a new target before moving again.

While it is known that the *Random Waypoint Mobility Model* has some disadvantages (e.g. density waves, highly variable average neighbor statistic, among others [4]), we have chosen it for our initial study for its simplicity. However, it is known that the performance of certain protocols is highly dependent on the underlying mobility model [4]. Therefore, in the conclusion section we outline the next set of trials we are performing to test OCGR in other environments.

C. Routing Protocols for Mobile Ad-hoc Networks

Many routing protocols have been proposed in the literature for mobile ad-hoc networks. However, in this section we only provide a brief introduction for the three algorithms that will be used to benchmark OCGR.

Epidemic routing is the first candidate [3]. As its name indicates, its core idea is to spread messages through the network epidemiologically (i.e., by flooding) with the hope that at some point one of copies reaches destination. To control the flooding mechanism, network managers are allowed to set a threshold on a message's maximum hop count. If a router receives a message that exceeds this threshold, then it is immediately discarded instead of forwarded.

Spray and Wait is competing routing algorithm for mobile ad-hoc networks [5]. It consists of two phases: In the spray phase, the node that generates a packet, and any subsequent relays, create a total L copies of it (i.e., the network will never have more than L copies of a message at any point in time). Under conditions of IID node movement, this is optimally accomplished using a binary splitting mechanism by which a node having $n > 1$ copies of a message will forward $\lfloor n/2 \rfloor$ to a neighbor and keep $\lceil n/2 \rceil$ for itself, so long as the neighbor does not already have copies available. Finally, if the destination is not found during the spray phase, then all nodes that hold any of the L copies of the message can only deliver it directly to its destination.

PROPHET is yet another alternative for routing in mobile ad-hoc networks [6]. However, unlike the previous two alternatives, PROPHET uses information about the history of contacts encountered to estimate the probability of bundle delivery to destination by any given node. If this probability is greater than a configurable threshold, then a copy of the message is forwarded. Otherwise, the message is kept in memory and the delivery probabilities of the network nodes are updated and aged as time passes and nodes come in and out of contact.

While Epidemic routing, *Spray and Wait* and PROPHET are different in the execution of their routing procedures, they also share common elements. Most notable is the requirement to exchange state information whenever two nodes come into contact. This state information is of two types: The first type helps control the flooding mechanism and reduce unnecessary transmissions by notifying each node of the messages that have already been seen by (and are possibly stored at) its peer. Alternatively, the second type of state information summarizes past network topology dynamics in the hope that it will have predictive power over future node motion.

D. Contact Graph Routing and Opportunistic Contact Graph Routing

Contact Graph Routing (see [7], [8]) is a routing protocol conceived to estimate end-to-end paths for bundles using a pre-computed database of contact opportunities between all nodes in the network (henceforth termed *contact plan*). Each contact opportunity is characterized by a six element tuple: Origin node, end node, start time, end time, mean data rate and contact confidence level. While pre-computing this database might seem unrealistic for typical Earth-based networks, it is actually standard practice for spacecraft operations, the original users for which CGR was developed. Furthermore, since contact opportunities can be known with high precision using orbital propagators, CGR assigns a confidence level of 1.0 to all contacts by default.

CGR's routing procedure starts by using the information in the *contact plan* to construct a graph that captures the time-varying topology of the network. Then, bundles are routed by performing a shortest-path search through the graph, using best-case delivery time as the search's objective metric [8]. In other words, the algorithm minimizes the time at which the bundle would be delivered taking into consideration transmission time, propagation delay and contact outages in the network, but obviates other potential sources of latency such as queuing delays at intermediate nodes. Finally, CGR also attempts to avoid overflowing links by purposely avoiding routes that do not have enough data volume capacity to accommodate a bundle of given priority. Once again, this mechanism is optimistic in nature since no information about the current state of another node/link in the system can be assumed known.

Opportunistic Contact Graph Routing extends the capabilities of CGR by handling contacts whose existence is uncertain (i.e., cannot be pre-scheduled) [9]. In essence, OCGR attempts to retain the end-to-end path searching mechanism of CGR

(i.e. shortest path through time-varying graph) while providing the functionality required to add new contacts that were not known a priori and are either discovered in real-time or predicted [9]. To that end, each node using OCGR maintains two databases of contacts, the *contact plan* which contains contacts that are pre-scheduled and thus will happen with certainty, and the *contact history*, which stores contacts that are discovered. When a bundle requests a new route, the contact history is first used to predict future contacts (and their respective properties, start time, end time, etc.). These are assigned a confidence level depending on the amount of information available in the contact history to make the prediction. Next, the predicted contacts and the *contact plan* are merged and used to build a time-varying graph that represents the current and (partially predicted) future topology, just as in CGR. This graph is then used to compute the end-to-end route for the bundle using CGR's-inherited shortest-path algorithm. The resulting route is then assigned an overall delivery confidence level equal to the product of the confidences for all contacts in the route. Finally, this value is used as the arrival confidence level if the bundle is to be forwarded using the first contact in the route¹, which determines the immediate neighbor that the bundle needs to be transmitted to.

Three fundamental parameters are used to control the behavior of OCGR: First, the *base confidence* is used together with the size of the contact history to estimate the confidence of a predicted contact between any two given nodes [9]. Second, γ specifies the minimum confidence improvement in bundle delivery that a path needs to provide to be accepted as a valid candidate for routing. This is clearly reminiscent of PROPHET routing, where nodes hand a bundle over to a neighbor only if they think that it has a high enough likelihood to deliver it to its final destination. Finally, η controls the number of times a given node will send a bundle to its neighbors so as to ensure that it has reached an acceptable delivery confidence level. This parameter, in turn, is similar to Epidemic and *Spray and Wait* control parameters. It essentially tunes the number of copies of a given bundle that circulate through the network.

III. BENCHMARK SETUP

The benchmark setup is based on the ONE network simulator [10] and ION, a widely used implementation of DTN [11]. The interface between both programs was done by Rodolfi in [9], who also added part of the required functionality for OCGR to work (e.g., the contact history exchange mechanism). Next, we describe the configuration parameters used during this study: All scenarios were initially configured as a square of 2000 meters in length with a variable number of nodes in it. To keep the computational complexity of OCGR manageable, at most 15 nodes were considered in the system. Furthermore, each node was assumed to generate 1MB

¹While CGR computes end-to-end routes for bundles, it does not do source-routing. Indeed, as the bundle progresses through the network, each intermediate node will compute an end-to-end path, which may or may not be equal to the paths computed by nodes previously visited. Provisions to avoid cycles in the outcome of the routing procedure are also enforced [8].

messages with infinite time-to-live at a non-constant rate of 12.5 messages per second for the entire simulation, which lasted 5 hours.

A total of 60 network configurations were evaluated by varying three parameters: Node speed, node density and transmission range. Node speed was set to either 5km/h (walking pace) or 16km/h (slow driving conditions). Node density was varied from 5 to 15 nodes in increments of 2 nodes. Finally, the transmission range for any node was increased from 100 meters, to 250, 500, 750 and 1000 meters. Also, since the traffic pattern and motion of nodes was stochastic, a total of 20 simulations per network configuration were performed and all results were computed as the mean value across them. Therefore, a grand total of 1200 simulations were run.

A. Analysis of Network Configurations

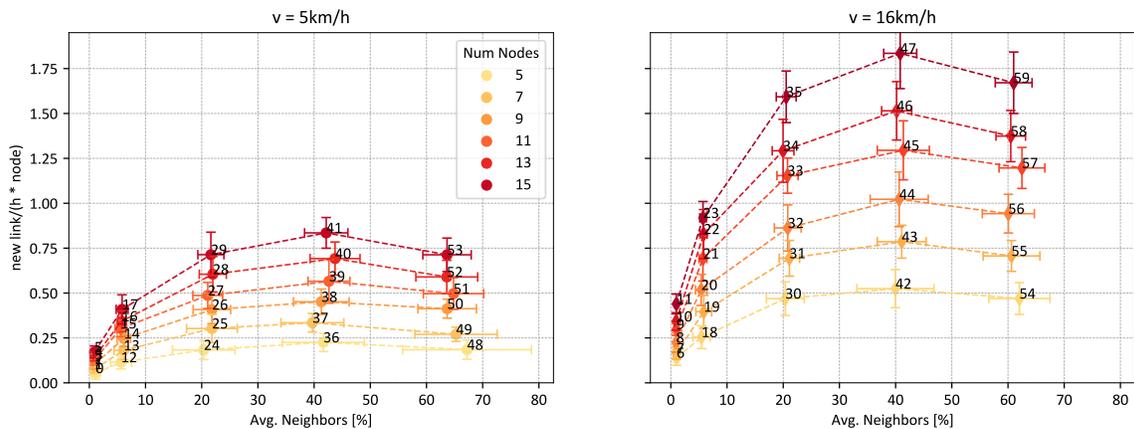
The performance of a routing algorithm in a mobile network primarily depends on three properties: Network sparsity, which measures, at any point in time and for any given node, how many nodes are within transmission range (i.e., are neighbors); network dynamicity, which measures the rate at which new connections are established in the system; and network stability, which quantifies how long a connection remains active as nodes move across the scenario. As we will see, these three properties are not independent from one another.

Figure 1a summarizes the network sparsity and dynamicity for all network configurations considered². To measure sparsity, we compute the average number of neighbors that a node has at any instant in time and normalize it by all pairwise links possible in the system. For network dynamicity, we compute the average number of links that node establishes every hour of simulation. Two plots are provided, one for nodes moving at a speed of 5km/h and another for 16km/h. Numbers next to each marker serve as identifiers for the network configuration considered, while error bars show the ± 1 standard deviation across 20 simulations. Finally, a single colored line in either plot is generated by varying the transmission range and keeping constant number of nodes and node speed, with increasing transmission range from left to right. So, for instance, the difference between networks configurations 42 and 54 is that the transmission range has been increased from 750 to 1000 meters.

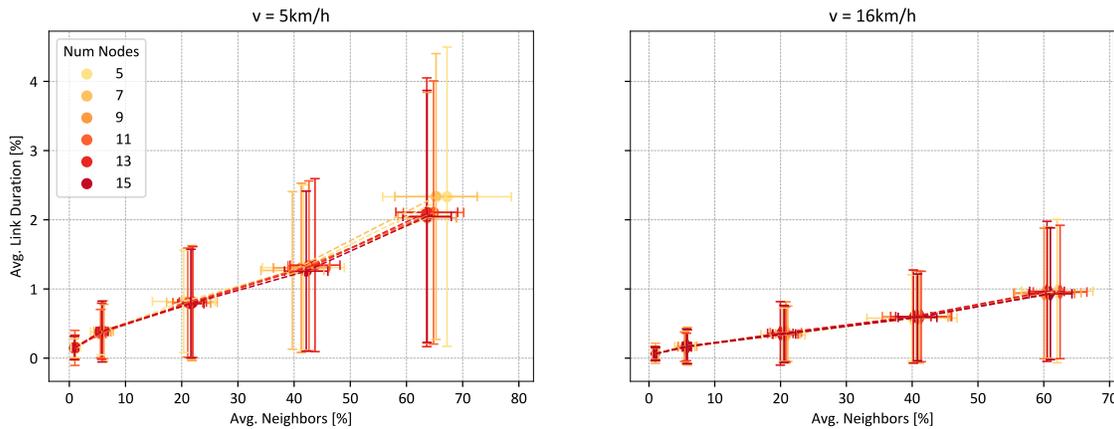
Figure 1b depicts the relationship between network stability and sparsity³. We observe, for instance, that the average link duration (normalized by simulation time) and mean number of neighbors exhibit a linear relationship, but the standard deviation increases as the network becomes denser. Therefore, the relationship between network dynamicity and network sparsity will have the same shape as in Figure 1a (i.e., an inverted parabola), albeit the error bars will be significantly larger. Also, note that the exact shape of all these curves depends on the underlying mobility model. However, some general trends are pervasive. Indeed, in a sparse network

²The number next to each marker identifies the network configuration tested. It references the top tier of Figure 2.

³Network configuration identifiers have been removed for the sake of clarity.



(a) Network Dynamicity vs. Sparsity



(b) Network Stability vs. Sparsity

Fig. 1: Summary of Network Configurations

where nodes are scattered and have low transmission range, nodes rarely encounter each other. Therefore, both the rate at which new connections are formed and their duration must necessarily be low (albeit the latter can be somewhat controlled by how fast they move). On the other hand, if nodes have a large transmission range, the network will necessarily be very dense as most nodes will be within range from each other all the time. Consequently, the rate of new connection will also be low, but the expected contact duration will be large. These intuitive reasonings are all well reflected in Figures 1a and 1b.

The combination of Figures 1a and 1b allows us to qualitatively partition the set of network configurations in three groups: *Sparse networks*, which comprises configurations 0 to 11; *connected networks*, encompassing networks 12 to 47; and *dense networks*, including configurations 53-59. *Sparse networks* have a reduced number of contact opportunities, their occurrence is rare, and when they do occur the contact is short-lived. Therefore, we do not expect any routing algorithm to deliver bundles to destination with high probability. *Dense networks* are the opposite: Nodes are within range of each other all the time and therefore delivering a bundle can be done in one or two hops. Therefore, the job of the routing

algorithm is easy, any neighbor it selects will probably be able to deliver the bundle. Finally, *connected networks* sit in between *sparse* and *dense networks*. They are the best type of network to compare the performance of routing algorithms, as bundle might traverse a large number of highly variable hops before reaching destination. Therefore, making correct routing decisions will impact overall system performance greatly.

IV. BENCHMARK RESULTS

To benchmark OCGR against Epidemic, *Spray and Wait* and PROPHET routing, we first need to define some evaluation metrics. Following previous studies (e.g. [3], [6]), we select bundle delivery probability and overhead ratio. The former is simply the probability that a bundle reaches its destination before the end of the simulation. Note that in practice we do not expect to find delivery probabilities equal to 1 for any scenario or router. Indeed, bundles generated just prior to the end of the simulation will not be delivered. On the other hand, the overhead ratio is defined as the ratio of bundles relayed (by any node to any other node) to bundles successfully delivered to destination, minus 1. Therefore, a higher overhead ratio will indicate more retransmissions required per bundle and worse

network flooding. Finally, we also record the average delivered bundle latency, in minutes, as a secondary evaluation metric.

Figure 2 summarizes the results of all simulations and networks configurations. It is divided in four tiers. The top tier is purely informational, it graphically depicts the set of parameters used in a given network evaluation. So, for instance, network configuration 11 assumed a transmission range of 100 meters, 15 nodes in the scenario and a speed of 16km/h. The second tier shows the bundle delivery probability as vertical bars grouped according to network configuration, while the third and fourth tiers display the overhead ratio and average delivered bundle latency, respectively.

Let us first concentrate on bundle delivery probability. Several interesting patterns emerge:

- For sparse networks, the delivery probabilities are low or moderate, never exceeding 0.75. Furthermore, for low node speed, OCGR seems to outperform all other routing algorithms, but this trend is reversed as soon as the speed of the nodes is increased. This suggest that OCGR shines in sparse and stable networks, i.e. networks where the number of routing alternatives is low and they vary slowly.
- For connected networks, the delivery probability transitions from less than 0.75 to almost 1 for all routing algorithms except for OCGR. Furthermore, for a given transmission range, the performance of OCGR decreases as the number of nodes in the system increases, irrespectively of the node speed.
- For densely connected networks, OCGR's performance almost matches the other routing algorithms. It is therefore impossible to favor any of the routers based solely on this metric.

Next, we consider the overhead ratio. For sparse and dense networks, we see that OCGR's performance is similar to the rest of routing algorithms. In fact, for dense networks OCGR seems to outperform all other algorithms, albeit the differences are not significant enough to be operationally relevant. On the other hand, OCGR's performance is significantly worse in the case of connected networks, especially as the number of nodes in the system increases. This fact, coupled with the decrease in bundle delivery probability, can probably be explained by the router making uninformed (almost random) decisions. This fact is backed by the exceedingly high average latency observed, which indicates that even if a bundle makes it to its final destination, it has traveled around the network for longer than with any other routing algorithm. In other words, we postulate that the *contact history* maintained by all nodes has no predictive power over future contacts. Consequently, the end-to-end paths inferred through the CGR-inherited graph search is meaningless and provides no improvement over randomly choosing the bundle's next hop.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have studied the performance of different routing algorithms in the context of mobile ad-hoc networks. Our original research goal was to understand whether OCGR,

an extension of CGR, can be used to service both networks with pre-scheduled and unpredictable contacts. Through simulation, we have compared OCGR against three well-known routing algorithms under a variety of network configurations, from very sparse to very dense topologies. Using bundle delivery probability and overhead ratio as the main comparison metrics, we have established that epidemic, *Spray and Wait* and PROPHET routers have similar performance when compared to each other, while OCGR can slightly overperform or largely under-perform depending on the network under consideration. In that sense, we have demonstrated that OCGR's attempt to route end-to-end bundles by maintaining a history of observed contacts, and using it to predict future ones, does not lead to meaningful routing decisions. This conclusion, however, is mostly conditioned by our choice of mobility model which, for this paper, was highly unstructured and random.

Relaxing our assumption of random waypoint motion is the primary area of future work. In particular, OCGR should be tested against mobility models where nodes exhibit periodic movement, even if imperfect. In that sense, we are working on testing OCGR using bus traces from the city of Rio de Janeiro. These are ideal since bus routes exhibit periodicity, they are typically closed loops with pre-defined stops. However, randomness in traffic conditions ensures that buses' positions cannot be pre-computed and contact opportunities cannot be pre-scheduled. Finally, another area of future work is related to understanding the impact of OCGR's configuration parameters in system performance. For this study, we have assumed reasonable values as a default based on previous studies.

ACKNOWLEDGMENT

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The authors would also like to acknowledge Kar-Ming Cheung, for his insights while performing this work.

REFERENCES

- [1] Cheung, Kar-Ming, et al. "Traffic Modeling for Deep Space Network in the Human Exploration Era." 14th International Conference on Space Operations. 2016.
- [2] Kesner, Samuel B., et al. "Mobility and power feasibility of a microbot team system for extraterrestrial cave exploration." Robotics and Automation, 2007 IEEE International Conference on. IEEE, 2007.
- [3] Vahdat, Amin, and David Becker. "Epidemic routing for partially connected ad hoc networks", 2000.
- [4] Camp, Tracy, Jeff Boleng, and Vanessa Davies. "A survey of mobility models for ad hoc network research." Wireless communications and mobile computing 2.5, 2002, pp. 483-502.
- [5] Spyropoulos, Thrasyvoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. "Spray and wait: an efficient routing scheme for intermittently connected mobile networks." Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking. ACM, 2005.
- [6] Grasic, Samo, et al. "The evolution of a DTN routing protocol-PROPHETv2." Proceedings of the 6th ACM workshop on Challenged networks. ACM, 2011.
- [7] Araniti, Giuseppe, et al. "Contact graph routing in DTN space networks: overview, enhancements and performance." IEEE Communications Magazine 53.3, 2015, pp. 38-46.

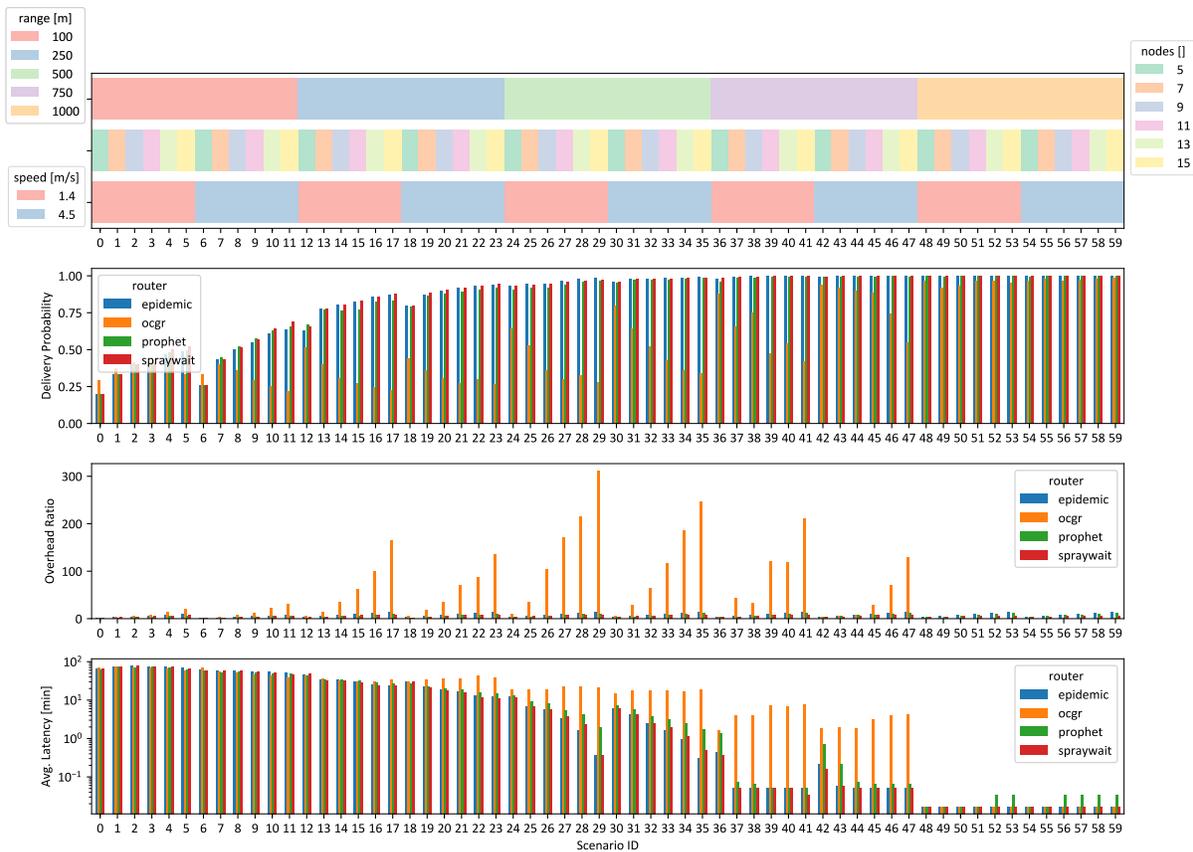


Fig. 2: Simulation Results

- [8] Fraire, J. A., Madoery, P., Burleigh, S., Feldmann, M., Finochietto, J., et al. "Assessing contact graph routing performance and reliability in distributed satellite constellations". *Journal of Computer Networks and Communications*, 2017.
- [9] Rodolfi, Michele. "DTN discovery and routing: from space applications to terrestrial networks". Masters Thesis, 2015.
- [10] Keranen, Ari, Jrg Ott, and Teemu Karkkainen. "The ONE simulator for DTN protocol evaluation." In *Proceedings of the 2nd international conference on simulation tools and techniques*, p. 55. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.
- [11] Burleigh, Scott. "Interplanetary overlay network: An implementation of the dtn bundle protocol.", 2007.



Marc Sanchez Net is currently a telecommunications engineer in the Communication Architectures and Research Section at JPL. His research interests include delay tolerant networking and its impact on distributed applications such as computational task sharing, spacecraft constellation management, as well as design of space communication systems in challenged environments such as the surface of the Moon. Marc received his PhD in 2017 from MIT, and also holds degrees in both telecommunications engineering and industrial engineering from Universitat Politècnica de Catalunya, Barcelona.

Marc Sanchez Net is currently a telecommunications engineer in the Communication Architectures and Research Section at JPL. His research interests include delay tolerant networking and its impact on distributed applications such as computational task sharing, spacecraft constellation management, as well as design of space communication systems in challenged environments such as the surface of the Moon. Marc received his PhD in 2017 from MIT, and also holds degrees in both telecommunications engineering and industrial engineering from Universitat Politècnica de Catalunya, Barcelona.



Scott Burleigh is a Principal Engineer at the Jet Propulsion Laboratory, California Institute of Technology, where he has been developing flight mission software since 1986. In 1988-1989 Mr. Burleigh developed one of the Laboratory's earliest Internet-enabled systems, a data distribution server that supported near-real-time analysis of science instrument data returned from the Voyager 2 encounter with the planet Neptune. Later, in the mid-1990s, Mr. Burleigh co-authored the specification for the CCSDS (Consultative Committee for Space Data Systems) File Delivery Protocol (CFDP), an international standard for file transfer over interplanetary distances. Mr. Burleigh developed the first implementation of CFDP, which was adapted for operational use on JPL's Deep Impact comet exploration mission. A member of the Delay-Tolerant Networking (DTN) Research Group of the Internet Research Task Force, Mr. Burleigh is a co-author of the DTN Architecture definition (Internet RFC 4838). He is also a co-author of the specification for the DTN Bundle Protocol (BP, Internet RFC 5050) supporting automated data forwarding through a network of intermittently connected nodes. In addition, he is a co-author of the specifications for the Licklider Transmission Protocol (LTP, Internet RFCs 5325 through 5327) supporting data block transmission reliability at the data link layer. Mr. Burleigh leads the development and maintenance of implementations of BP and LTP that are designed for integration into deep space mission flight software, with the long-term goal of enabling deployment of a delay-tolerant Solar System Internet. The initial exercise of these protocol implementations on an operational spacecraft occurred during the Deep Impact Network (DINET) experiment conducted in interplanetary space during October and November of 2008. This software is now in continuous operation on the International Space Station and is offered on SourceForge as open source code. Mr. Burleigh has received the NASA Exceptional Engineering Achievement Medal and four NASA Space Act Board Awards for his work on the design and implementation of these communication protocols.

Scott Burleigh is a Principal Engineer at the Jet Propulsion Laboratory, California Institute of Technology, where he has been developing flight mission software since 1986. In 1988-1989 Mr. Burleigh developed one of the Laboratory's earliest Internet-enabled systems, a data distribution server that supported near-real-time analysis of science instrument data returned from the Voyager 2 encounter with the planet Neptune. Later, in the mid-1990s, Mr. Burleigh co-authored the specification for the CCSDS (Consultative Committee for Space Data Systems) File Delivery Protocol (CFDP), an international standard for file transfer over interplanetary distances. Mr. Burleigh developed the first implementation of CFDP, which was adapted for operational use on JPL's Deep Impact comet exploration mission. A member of the Delay-Tolerant Networking (DTN) Research Group of the Internet Research Task Force, Mr. Burleigh is a co-author of the DTN Architecture definition (Internet RFC 4838). He is also a co-author of the specification for the DTN Bundle Protocol (BP, Internet RFC 5050) supporting automated data forwarding through a network of intermittently connected nodes. In addition, he is a co-author of the specifications for the Licklider Transmission Protocol (LTP, Internet RFCs 5325 through 5327) supporting data block transmission reliability at the data link layer. Mr. Burleigh leads the development and maintenance of implementations of BP and LTP that are designed for integration into deep space mission flight software, with the long-term goal of enabling deployment of a delay-tolerant Solar System Internet. The initial exercise of these protocol implementations on an operational spacecraft occurred during the Deep Impact Network (DINET) experiment conducted in interplanetary space during October and November of 2008. This software is now in continuous operation on the International Space Station and is offered on SourceForge as open source code. Mr. Burleigh has received the NASA Exceptional Engineering Achievement Medal and four NASA Space Act Board Awards for his work on the design and implementation of these communication protocols.