Predictable Mobile Routing for Spacecraft Networks

Daniel Fischer, David Basin, Knut Eckstein, and Thomas Engel

Abstract—In predictable mobile networks, network nodes move in a predictable way and therefore have dynamically changing but predictable connectivity. We have developed a model that formalizes predictable dynamic topologies as sequences of static snapshots. We use this model to design and evaluate a predictable mobile-routing protocol based on link-state routing, whose performance is superior to its static and ad-hoc counterparts. Our routing protocol accounts for occurrences of additional, unpredictable changes, as well as their interaction with predictable changes.

We evaluate our protocol using simulations based on randomly generated topologies and spacecraft-network scenarios. In both cases, we show that our protocol outperforms traditional routing protocols and is well suited for routing in next-generation space networks.

Index Terms—routing, predictable mobility, spacecraft networks, space-link.

1 Introduction

Problem Context and Motivation: Spacecraft networks are a topic of growing importance within space agencies and industry. Existing and emerging applications call for efficient routing within an interconnected space environment [1].

Future spacecraft networks will exhibit characteristics different from existing terrestrial networks. Spacecraft networks are mobile networks with independent, heterogeneous nodes but without spontaneous movement. In regular operation, each spacecraft flies along a predictable trajectory that can be computed well in advance. Hence, spacecraft networks constitute a subclass of mobile ad-hoc networks that is characterized by its predictability. As in other networks, unpredictable changes may still occur and influence the evolution of the topology, in particular, when nodes or links fail.

Approach: We present a formal topology model for predictable mobile networks that describes the topology evolution by a sequence of static network-topology snapshots. In this way, we incorporate predictability while abstracting away from details such as flight dynamics for individual orbits and flight paths. We used this model to develop a general purpose Predictable Link-State Routing (PLSR) protocol that accounts for both predictable and unpredictable changes and their interaction.

We evaluate the performance of PLSR in two ways. First, we use simulations based on randomly generated topologies and mobility patterns to show that PLSR's performance exceeds that of LSR. We also make comparisons with OLSR. Second, we evaluate PLSR by simulating communication using data from future space-mission scenarios based on realistic mobility patterns and failure assumptions. In particular, we investigate two mid-term spacecraft-network scenarios that are under consideration by ESA, NASA, and other agencies [2].

1) A hybrid Low Earth Orbit (LEO)/Geostationary Earth Orbit (GEO) spacecraft network that supports Earth observation operations.
2) A communication infrastructure for a surface mission, such as a Mars rover.

We use the AGI Satellite Toolkit (STK) [3] to construct sequences of topology snapshots for the above scenarios. STK is a state-of-the art tool for solving location and inter-visibility problems associated with space scenarios. Afterwards, for both scenarios, we perform network traffic routing simulations using an implementation of PLSR in the ns-2 network simulator and we measure PLSR's performance against that of link-state routing.

Contributions: Our main contribution is a topology model and routing protocol for predictable mobile networks, in particular spacecraft networks. We prove our routing protocol correct and we demonstrate its superiority over alternative approaches in terms of protocol overhead and data throughput. Our results provide strong evidence that PLSR is well-suited for near-term and next-generation spacecraft networks.

Organization: In Section 2, we provide background on spacecraft networks and our network scenarios. In Sections 3 and 4, we introduce our snapshot topology model, and address topology information management, and distribution. In Section 5, we present our PLSR protocol. In Section 6, we describe how we generate snapshot sequences using a flight-dynamics simulator and, in Section 7, we perform a performance analysis in a setting with random node movement. In Sections 8 and 9, we evaluate our protocol using the two application scenarios described above. Finally, we review related

- *D. Fischer is with the European Space Agency, Darmstadt*
- *T. Engel is with the Department of Sciences, Technology and Communication, University of Luxembourg*
- *D. Basin is with the Institute of Information Security, ETH Zurich*
- *K. Eckstein is with European Space Agency*

work and draw conclusions in Section 10.

2 Spacecraft Networks

Satellite communication is traditionally one of two kinds: either (1) direct point-to-point links from control centers to spacecraft or (2) bent-pipe communication applications, where spacecraft relay a data stream. Neither approach uses network routing technologies. Earlier attempts to use more sophisticated spacecraft communication models in the form of Low Earth Orbit constellations with inter-spacecraft links [4], [5] had limited commercial success due to their inability to integrate the spacecraft network with other terrestrial networks [1].

Emerging application areas for space missions [2] require integrated multi-purpose spacecraft networks and constellations supporting spacecraft of different sizes, types, and flight dynamics. These spacecraft may also communicate with different terrestrial end-user terminals or networks, ranging from rovers on Mars to satellite telephones on Earth. Spacecraft routing solutions using the traditional communication methods introduced above would result in a setup that is solely based on manual commanding of the individual point-to-point links. As the number of nodes grows, such a non-autonomous setup becomes unmanageable from the perspective of planning and commanding. Hence, a topology model and a routing protocol for space networks must cope with a large variety of differently behaving nodes and links with the property of predictability.

Spacecraft networks are controlled and managed by a control center that is connected to the network via one or more ground stations. We will restrict our attention to the case of a single ground station in this paper. This is without loss of generality as multiple ground stations can always synchronize over a terrestrial network.

NASA, ESA, and other space agencies are planning the next generation of networked space infrastructures to support current and future space missions. NASA has identified a number of realistic mid-term scenarios [2], including LEO/GEO and Mars networks.

Spacecraft Network Properties

Spacecraft networks have the following properties, which we take into consideration when developing our topology model and routing protocol.

- **Predictable high mobility:** Spacecraft move at high velocities and the communication opportunities between them are therefore often very short and change rapidly. However, since spacecraft follow predictable flight paths, the network topology changes in a predictable way. Hence, one can compute in advance the time points when nodes or links come or go as well as changes in link properties.
- **Presence of one or more central processing units:** All spacecraft networks are controlled from ground control centers, which have substantial computational resources. This effectively allows one to redistribute tasks from the spacecraft to the ground.
- **Rare occurrences of unpredictable changes:** While unpredictable changes can occur at any time in spacecraft networks, e.g. due to solar flares or equipment failures, they are uncommon.

3 Topology Model

In this section, we present a formal topology model for predictable mobile networks that also handles unpredictable topology changes. While our work is motivated by spacecraft networks and their properties, our model is general and covers all predictable mobile networks.

Designing and reasoning about algorithms for spacecraft networks requires a topology model that describes networks with predictable and unpredictable changes, as described in the last section. We first formalize a topology model that handles predictable changes based on a notion of snapshots. A snapshot describes the static state of the network topology during a time interval and a snapshot sequence describes the expected evolution of this state over time. We formalize predictable changes by a snapshot transition function. In contrast, unpredictable changes must be discovered and corrected for, essentially by superimposing them upon the snapshot sequences.

3.1 Predictable Topology Changes

We specify the behavior of the network topology in the presence of predictable changes and introduce a model that characterizes predictability in mobile networks. The concept of snapshots was informally introduced in [6], where the authors describe the mobility of a LEO spacecraft network. They state that a snapshot represents the topology of the spacecraft network at a particular instance of time. We formalize the snapshot concept, associating it with a time interval, not a time instance.

Definition 1: Time Intervals and Transitions. A *time set* is a triple $(T, \leq, t_0)$. The elements of the set T are called *time points*, $\leq$ is a total order on T, and t_0 is the smallest element of T under $\leq$. We will often leave $\leq$ and t_0 implicit and simply speak about the time set T. For $t', t'' \in T$, we will use the notation $[t', t''[$, for $t' < t''$, to represent the *time interval* $\{t \in T \mid t' \leq t < t''\}$.

Let I be a countably infinite subset of T. We call I a *set of transition points* if for all $t \in T$ there are $t', t'' \in I$ with $t \in [t', t''[$. Since t_0 is also the initial time point, it follows that $t_0 \in I$. We will enumerate the elements of I as t_0, t_1, ..., where for all $i \in \mathbb{N}$, $t_i < t_{i+1}$. $\Diamond$

We can now describe the topology during a given time interval $[t_i, t_{i+1}[$.

Definition 2: Connectivity Graph and Snapshots. The predicted network topology during an interval $[t_i, t_{i+1}[$ is described by a directed *connectivity graph* $G_i = (V_i, E_i)$, where V_i is the set of network nodes and E_i is the set of active communication links.

For $t_i \in I$, a *snapshot* $S_i = \langle t_i, G_i \rangle$ represents the predicted topology in the time interval $[t_i, t_{i+1}[$. The predicted evolution of the topology over time is described by a *snapshot sequence* $S = \langle S_0, ... \rangle$. A transition point t_i, for $i > 0$, represents the time point when the topology changes from S_{i-1} to S_i. $\Diamond$

A sequence of snapshots describing predictable changes can be defined by an initial snapshot S_0 and a transition function δ that maps snapshots to snapshots. The transition function describes how the network connectivity is expected to change over time. Note that δ may be different for each network topology and sequence of snapshots. The snapshot sequence defined is $\langle S_0, S_1, S_2, ... \rangle$, where $\delta(S_i) = S_{i+1}$, for $i \in \mathbb{N}$.

Defining δ for a spacecraft network requires sophisticated calculations using the orbital mechanics and flight dynamics of the involved spacecraft. We abstract away from these considerations for now and focus on the nature of such snapshot transitions. In Section 6, we describe the functions δ that we use in the LEO/GEO spacecraft and Mars networks.

3.2 Unpredictable Topology Changes

Although predictable changes constitute the vast majority of the events that determine the topology dynamics in a spacecraft network, unpredictable changes, which cannot be anticipated, can also occur at any time, also between snapshot transition points.

Unpredictable changes are not covered by the transition function and must be discovered by the nodes using link-sensing. This means that nodes must check their neighborhood for unpredictable changes either proactively, at regular time intervals, or reactively.

Unpredictable changes transform the reality that is represented by a given snapshot. Let S_i be the current snapshot and let u be an unpredictable change that occurs within the interval $[t_i, t_{i+1}[$ associated with S_i that describes the addition or removal of an edge in the topology graph G_i. After u occurs, the snapshot S_i no longer correctly represents reality. The reality corresponds instead to a modified snapshot $\hat{S}_i$, whose topology graph $\hat{G}_i$ accounts for u. Said another way, the reality can be abstracted to a snapshot $\hat{S}_i$, where $\hat{S}_i$ differs from G_i only in the presence of the unpredicted change. Figure 1 illustrates this and that the deviation accumulates with each additional unpredictable change.

This raises two problems that must be resolved when designing protocols for predictable mobile networks. The first problem is related to the interaction of predictable and unpredictable changes at transition points. As the predictable snapshot sequence does not include unpredictable changes, such changes may be overwritten by the snapshot topology images. The second problem is the need for regular realignment of the original predicted snapshot sequence $\langle S_0, S_1, S_2, ... \rangle$ given by δ with the real sequence $\langle \hat{S}_0, \hat{S}_1, \hat{S}_2, ... \rangle$ to incorporate modifications due to past unpredictable changes. In this paper, we

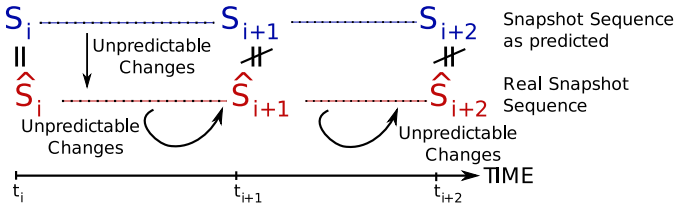

Fig. 1. Deviation between reality and the predicted snapshot sequence caused by unpredictable changes

provide solutions to both problems. Note that these problems (and opportunities) do not arise in mobile ad-hoc settings without the a-priori topology knowledge of node movement and communication opportunities since all changes are unpredictable.

4 Managing Topology Information

In predictable mobile networks, routing protocols can benefit from predictability only if the nodes that perform routing have access to information about upcoming snapshots. In this section, we describe the distribution and management of this topology information. Distribution is required because it is generally infeasible to store all topology management information in the nodes at the time of network initialization (snapshot S_0). We present a topology information management algorithm that supports our link-state based routing scheme.

4.1 Snapshot Distribution

Spacecraft memory is adequate for storing snapshot sequences of reasonable length [7]. Each node stores a sequence $\langle S_j,..,S_{j+k} \rangle$ of $k+1$ snapshots, with $k \geq 0$, and requires an update with a new sequence $\langle S_{j+k+1},... \rangle$ before time t_{j+k+1}. This update process must also modify snapshot updates to account for unpredictable changes that have affected previous snapshots.

Recall that spacecraft networks contain a ground station that has extensive computational resources and can therefore pre-compute sets of snapshots. Furthermore, by participating in the network as a node, the ground station also receives information on unpredictable changes and can incorporate them in future snapshots. To facilitate this, all unpredictable changes are time-stamped. This allows the nodes to distinguish between those unpredictable changes that have been used as an input for the snapshot computation and those that occur between the snapshot computation and the time point at which the computed snapshot becomes valid (i.e. the time point t_i for a snapshot S_i). Information on unpredictable changes that reach the ground station is stored in a data structure UC, which holds the changes together with their time stamps.

Providing an initial pre-computed snapshot sequence to all network nodes eliminates the need for an initialization phase in the routing protocol. Nodes that are

added to the network by a predictable change can also be equipped with an initial sequence of snapshots.

We will use the following definitions in our formalization of snapshot distribution and our routing algorithm.

Definition 3: Restricted Graphs and Neighborhood. Let $G = (V, E)$ be a graph and $V' \subseteq V$ a subset of nodes. Then $E\,|_{V'} = E \cap (V' \times V')$ denotes the *restriction of* E *to* V' and $G\,|_{V'} = (V', E\,|_{V'})$ denotes the *restriction of* G *to* V'. We define the *edges neighboring* v *in the graph* $G = (V, E)$ as $\{(x, y) \in E \mid x = v \vee y = v\}$. Similarly, the *neighbors of* v *in the graph* $G = (V, E)$ are $\{w \in V \mid (v, w) \in E \vee (w, v) \in E\}$. $\Diamond$

Definition 4: Transition-affected subgraph. Let $t_i \in I$ be a transition point and S_i the corresponding snapshot. We define $VT_i = \{v \in V_i \mid \exists (x, y) \in E_i \setminus E_{i+1} : (x = v) \vee (v = y)\} \cup \{v \in V_i \mid \exists (x, y) \in E_{i+1} \setminus E_i : (x = v) \vee (y = v)\}$. This is the set of nodes v in V_i for which the application of a snapshot transition at time t_{i+1} causes v to update its information on neighboring nodes or edges. We then define $GT_i = G_i\,|_{VT_i}$, the *restriction of* G *to* VT_i, as the *transition-affected subgraph* associated with a transition from snapshot S_i to S_{i+1}. $\Diamond$

Distribution Scheme: Let $\langle S_i, ..., S_{i+k} \rangle$ be a sequence of pre-computed snapshots that is to be distributed. Algorithm 1 describes, in high-level pseudocode, the steps taken by the ground station for each sequence of k snapshots. A new snapshot S_j is calculated from S_{j-1} using δ (line 3). The computed snapshot is then updated with the information stored on past unpredictable changes (line 4). Afterwards, a time stamp is added, indicating the time point at which the most recent unpredictable change has reached the ground station (line 6). Then, all entries are removed from the data structure `UC` (line 7), which holds all unpredictable changes received since the last computation. Finally, the new snapshot sequence is distributed to the other nodes in the network (lines 8–10).

Algorithm 1 Snapshot Sequence Calculation and Distribution Function `snap_calc`(i, k) for a ground station

Input: i (first snapshot index of the new sequence) and k (number of snapshots in the sequence).

```
1: var S_{i-1},...,S_{i+k} /* Snapshot variables, where S_{i-1} contains the last computed snapshot change */
2: for all j=0 to k do
3:     S_{i+j} ← δ(S_{i+j-1})
4:     S_{i+j} ← ApplyStoredUnpredictableChanges(S_{i+j})
5: end for
6: UpdateTimeStamp(⟨S_i, ..., S_{i+k}⟩)
7: DeleteAllChanges(UC)
8: for all Neighbors x of GS do
9:     Send (⟨S_i, ..., S_{i+k}⟩, x)
10: end for
```

Distribution Scheme Efficiency: The change in the topology $\delta(S_i)$ is limited to the subgraph GT_i of G_i. Adding or deleting an edge from E_i involves two nodes x and y and one edge (x, y) in GT_i and adds only a constant amount of space to the size of the update for S_{i+1}. For each node that is added or deleted from V_i, the number of edges that can change is limited to the number of neighboring edges, which is $2\,|V_i|$ edges, in the worst case. Given this, we can reduce the data overhead by propagating just the differences between the snapshots S_i and S_{i+1}. Note that instead of using flooding (as in Algorithm 1), one could alternatively use a more efficient multicast solution. Snapshot updates can be distributed to the nodes at any time before the snapshots are needed by the nodes.

5 Routing Protocol

We now specify our PLSR protocol for predictable mobile networks. We build it on top of (basic) link-state routing [8], which we briefly introduce.

5.1 Link-State Routing Algorithm

Each unpredictable change u describes the addition or deletion of a single edge in the topology graph. When a node v changes its Link-State Database (LSDB) to incorporate such a modification, we say that v *has updated its LSDB with the change* u. Since u affects only one edge in the network, only the adjacent nodes can discover the occurrence of u using link-sensing. The adjacent nodes will, after detecting the change, initiate a Link-State Advertisement (LSA), describing u. Note that more than one unpredictable change may occur at the same time point. For example, this occurs when a node with more than one adjacent edge fails.

If two consecutive unpredictable changes u and u' affect the same edge, then u' will reverse u. In this case, we say the unpredictable change u is *superseded* by u'.

Algorithm Description: Algorithm 2 shows the routing main loop of the basic link-state protocol for a node v in the network as introduced in [8]. Note that periodic beaconing is omitted for reasons of clarity. In the algorithm, if v detects a change on the link to another node x (line 4), it adjusts its own LSDB and creates an LSA message for the change (lines 5–6). The shortest path first (SPF) tree is then updated (line 7) and an LSA is flooded to all other nodes in the topology (lines 9–11). If v receives an LSA from a node w (line 13), it checks its freshness and the LSA is either processed (lines 15–19) or dropped (line 21). Processing an LSA includes updating the local LSDB with the changes, continuing the flooding, and storing the LSA for future freshness checks. This flooding mechanism rapidly propagates network changes through the network at the cost of increased traffic overhead.

5.2 Assumptions and Requirements

We adapt the basic link-state routing protocol from Algorithm 2 for use in predictable mobile environments

Algorithm 2 Basic link-state main loop for a node v

```
1: var x /* Variable x describes a change */
2: var lsa /* Variable lsa contains an LSA as described in Section 5.1*/
3: while true do
4:   if x = DetectChange() then
5:     UpdateLSDB(x) /* Updates LSDB with x */
6:     lsa ← CreateLSA(x) /* New LSA from x */
7:     ModifySPFTree()
8:     /* LSA Flooding */
9:     for all neighbors y of v /* v is the current node */ do
10:      Send (lsa, y) /* Send lsa to y */
11:    end for
12:  end if
13:  if (lsa, w) ← Receive() /* Receive lsa from w */ then
14:    if IsFresh(lsa) then
15:      UpdateLSDB(lsa)
16:      ModifySPFTree()
17:      for all neighbors s of v except w /* LSA Flooding */ do
18:        Send(lsa, s)
19:      end for
20:    else
21:      Drop(lsa) /* LSA not recent */
22:    end if
23:  end if
24: end while
```

by incorporating information from snapshot sequences. Before describing our adaptations, we state our assumptions and requirements for the protocol.

Assumptions: We design our routing algorithm to work correctly under the following assumptions:

1) Spacecraft clocks are synchronized.
2) The time period required for local node computations is negligible.
3) The network topology does not change too quickly. In particular, a flooding algorithm, initiated by a set of nodes $V \subseteq V_i$ during a snapshot S_i, reaches all other nodes $w \in S_j$, for some $j \in T$ and $j \geq i$.
4) The topology graph is always connected (also after both predictable and unpredictable changes).

Although clock synchronization is a common problem in distributed systems, spacecraft employ high-precision clocks as many of their commands are time critical. Therefore, only small clock deviations are expected and this assumption is reasonable. Assumption 2 is also reasonable given the capacities of modern spacecraft on-board computers. Restricting the network dynamics in Assumption 3 expresses the need for the stability of the network topology and its evolution in terms of predictable changes. In particular, it excludes anomalies such as a topology that grows faster than the time required for messages to spread through the network. We describe the reasons for Assumption 4 below.

Network Separation through Unpredictable Changes: Consider an unpredictable change u that separates the topology into a main topology that contains the ground station and an isolated topology. This isolated topology can neither propagate any further unpredictable changes that affect its edges nor can it receive snapshot-sequence update messages from the ground station. Observe that when the nodes in the separated subgraph run out of snapshots, all predictable changes must be treated as unpredictable changes.

We do not consider this pathological case in this paper since it is very unlikely to occur in spacecraft networks. For example, spacecraft networks use redundant nodes to avoid network separation. If the topology is such that this case has a non-negligible probability of occurring, one can adapt routing to handle it. We only sketch a solution here. Namely, we can equip all nodes with the capability of switching into a full, ad-hoc mode that becomes active as soon as the nodes of an isolated topology exhaust their snapshot information. When the connection to the ground station is restored and fresh snapshot sequences can be acquired, the nodes switch back to predictable mobile routing mode and notify the main topology about unpredictable changes that may have occurred during the time of separation.

Predictable and Unpredictable-change consistency: We require two notions of consistency.

Definition 5: Change Consistency.

Let S_i be the current snapshot. *Predictable-change consistency* concerning a predictable change at time t_{i+1} holds if there exists a $t \in T$ with $t_{i+1} < t < t_{i+2}$ and all nodes $v \in V_{i+1}$ have a consistent view of the predictable change at t_{i+1} by time point t. *Unpredictable-change consistency* concerning an unpredictable change u that occurs at time point t, with $t_i \leq t < t_{i+1}$, holds if at some time point $\hat{t}$ associated with some snapshot S_k, with $k > i$ and $t_k \leq \hat{t} < t_{k+1}$ one of the following has occurred: (1) all nodes $v \in V_k$ have updated their LSDB with the change u, (2) u has been superseded by another unpredictable change u' that occurs at a time point t' with $t \leq t' \leq \hat{t}$, or (3) all nodes that have updated their LSDB with the changes.$\Diamond$

Predictable-change consistency is achieved when the nodes have updated their LSDB with the last predictable change. Due to clock synchronization, the time interval of inconsistency is very small in practice and no communication is required to achieve predictable-change consistency. In contrast, for an unpredictable change u, it cannot be guaranteed that the propagation process is completed within in the same snapshot in which u occurs. The conditions state:

- either all nodes have learned of the change u or
- the change is no longer relevant as it has (1) been superseded or (2) all information on u has been deleted from the topology

Note that unpredictable-change consistency is also a requirement for a basic link-state algorithm.

The LSDBs of all network nodes reflect the reality at time t (in snapshot $\hat{S}_t$) only if predictable-change consistency holds for all previous snapshot transitions and unpredictable-change consistency holds for all unpredictable changes that have occurred prior to t. As unpredictable changes occur rarely in spacecraft networks, the nodes' LSDBs often reflect reality.

Protocol Requirements: Our requirements are twofold: (1) following a predictable change in the network topology, predictable-change consistency holds and (2) following an unpredictable change in the network topology, unpredictable-change consistency holds.

Meeting the first requirement increases the quality of the packet forwarding process since all routing decisions made by PLSR are correct whenever no unpredictable changes are currently propagating through the network. As predictable-change consistency holds following each snapshot transition S_i to S_{i+1}, the time interval $[t_{i+1}, t]$, for $t \in T$ with $t_{i+1} < t < t_{i+2}$, where the routing tables are inconsistent with the network topology is negligible. If packets are forwarded during $[t_{i+1}, t]$, then errors may arise as either the previous snapshot is used or the LSDB of the forwarding node is in an inconsistent state. However, if the forwarding process is delayed until the routing tables are rebuilt, then all packets will be correctly forwarded with respect to the current network topology. Note that this effect on the quality of packet forwarding is a property just of topologies where predictable-change consistency holds.

5.3 Routing Protocol Definition

Algorithm 3 presents our link-state routing algorithm for predictable mobile networks. This algorithm handles both predictable and unpredictable changes, as well as their interaction. Because unpredictable changes cause snapshots to deviate from reality, their interaction with predictable changes must be specifically addressed.

Predictable Changes: For each transition point t_{i+1}, GT_i is the transition-affected subgraph. We modify the core routing protocol to handle predictable changes and update the topology image at each transition point t_{i+1} (line 6). Information on snapshot changes for the next snapshot has already been received and hence the LSDBs can be updated locally at the transition point t_{i+1} (line 8). Since only nodes in GT_i are affected, a complete re-run of the Dijkstra algorithm is not required (only an incremental update) to compute the local SPF trees (line 13). However, if a node is added by a predictable change, this node must learn of any recent unpredictable changes from one of its neighbors. Therefore neighbors of the newly added node create an LSA for each unpredictable change that they have stored in their `LSDB_UC` and send them to the new node (line 10).

Unpredictable Changes: During the time interval $]t_i, t_{i+1}[$ within a snapshot S_i, PLSR operates as the basic link-state routing. Moreover, an unpredictable change that is detected during $]t_i, t_{i+1}[$, or is received via an

Algorithm 3 PLSR main loop for a node v.

```
1: var x /* describes a change */
2: var lsa /* LSA as described in Section 5.3 */
3: var S_i,...,S_{i+k} /* Snapshot variables, contain a snapshot
   sequence consisting of k snapshots updates */
4: var n, s, v, w /* Variables describe nodes */
5: while true do
6:   if CurrentTime() == NextTransitionPoint() then
7:     t ← CurrentTime()
8:     newNodes ← UpdateLSDB(LoadChanges(t))
9:     for all neighbors n in newNodes do
10:      SendAllChanges(LSDB_UC, n)
11:    end for
12:    LoadUnpredictableChanges()
13:    ModifySPFTree()
14:  end if
15:  if x ← DetectChange() then
16:    n ← GetNeighborNode(x)
17:    if n ∉LSDB then
18:      Send(LSDB, n) /*Send LSDB to node n*/
19:    end if
20:    UpdateLSDB(x) /* Updates LSDB */
21:    ModifySPFTree()
22:    UpdateLSDB_UC(x)
23:    lsa ← CreateLSA(x) /* New LSA from x */
24:    for all neighbors y of v do
25:      Send (lsa, y) /* Send lsa to y */
26:    end for
27:  end if
28:  /* Receive an LSA lsa from a node w */
29:  if (lsa, w) ← Receive() then
30:    if IsFresh(lsa) then
31:      UpdateLSDB(lsa)
32:      ModifySPFTree()
33:      UpdateLSDB_UC(lsa)
34:      for all neighbors s of v except w /* LSA
         Flooding */ do
35:        Send(lsa, s)
36:      end for
37:    end if
38:  end if
39:  /* Receive a new snapshot sequence from a node w*/
40:  if (⟨S_i, ..., S_{i+k}⟩, w) ← Receive() then
41:    timestamp ← StoreSequence(⟨S_i, ..., S_{i+k}⟩)
42:    DeleteOldEntriesFromLSDB_UC(timestamp)
43:    for all neighbors s of v except w do
44:      Send (⟨S_i, ..., S_{i+k}⟩, s)
45:    end for
46:  end if
47:  if newLSDB ← Receive() then
48:    SetLSDB(newLSDB)
49:  end if
50: end while
```

LSA, is stored in an additional data structure LSDB_UC (lines 22 and 33) together with a time stamp indicating the reception or detection time of the most recent unpredictable change. If an unpredictable change results in the addition of a new node (line 16), this node must be provided with the current LSDB (line 18).

Interaction: Interaction between predictable and unpredictable changes occurs at transition points when an unpredictable change directly affects an edge with a neighboring node in GT_i. In this situation, the order in which the changes are applied is important. We first apply the predictable changes to the LSDB (line 8) and then the unpredictable changes from the data structure LSDB_UC (line 12). Thus, the effects of unpredictable changes are preserved across snapshot transitions.

Reception of Snapshot Sequences: If a new snapshot sequence is received from another node (line 40), its time stamp is retrieved and the new sequence is stored for later use (line 41). Afterwards, obsolete entries are deleted from LSDB_UC (line 42) and the new sequence is further distributed (lines 43–45).

5.4 Protocol Correctness

We state the following theorem, which we proved in [9].

Theorem 1 (PLSR Protocol Correctness): The PLSR Protocol satisfies the requirements defined in Section 5.2.

6 Topology Generation

Before we evaluate the performance of PLSR, we describe how we generate the snapshot topologies for our simulations. First we explain how we generate random topologies. Afterwards, we describe how we generate topology information from real flight-dynamics data for our application scenario.

6.1 Random Topology Generation

Our tool for random topology generation simulates the movement of nodes using a random waypoint model [10]. To achieve this, we assign a maximum communication range to each node. The nodes then move randomly inside a simulated area. As long as two nodes are in communication range with each other, they have a communication link in the topology. The time points when a link is created or dropped become snapshot-transition points. In the random-movement scenario, we use the same propagation delays of either 33, 330 or 3300 ms for each link. Also, all links are assigned an identical cost value. We created a UDP packet transfer between two random network nodes for a fixed time interval and two different packet production rates (20 and 2000 packets per second). Each packet has a size of 999 bytes. We measured the traffic arriving on the destination node.

6.2 Topology Generation from Flight-Dynamics Data

The current state-of-the art tool for producing flight dynamics information is the commercial Satellite Tool Kit (STK) [3], which is used by many space agencies and commercial operators. STK can accurately simulate the movement of any body or spacecraft within the solar system or on celestial bodies. It can also model spacecraft antenna pointing, visibility widows, and other spacecraft properties. We use STK to compute the lines-of-sight and distance between network nodes.

Visibility and Communication Windows: As the distance between nodes in space networks can be substantial, propagation delays may affect the topology by reducing communication possibilities. To account for this, we define two key notions: *visibility windows* and *communication windows*. These are used by the flight-dynamics simulator and the topology generator when calculating snapshots.

Definition 6: Visibility and Communication Windows.

Let a and b be two nodes in the topology and let $t_1, t_2 \in T$ be two time points. We say that $[t_1, t_2[$ *is a visibility window between* a *and* b if the following two conditions hold.

1) During $[t_1, t_2[$, there is a continuous direct line-of-sight between a and b.
2) This time interval cannot be extended. Namely, there is no time interval $[t_1', t_2'[$, with $t_1', t_2' \in T$, such that $[t_1, t_2[\subsetneq [t_1', t_2'[$ and a continuous direct line-of-sight exists between a and b during $[t_1', t_2'[$.

Given a visibility window $[t_1, t_2[$, let $d \in \mathbb{R}$ be the propagation delay between the two spacecraft a and b at the time point t_2. Then, the corresponding *communication window* is defined as $[t_1, t_2 - d[$. $\Diamond$

The difference between a visibility window and the associated communication window is substantial when large distances are involved, such as between Earth and Mars. In our approach, the topology model that is used by the topology generator provides snapshots based on communication windows rather than visibility windows. Note that the communication window does not extend earlier in time than the visibility window because earlier communication attempts starting at time $t_1 - d_1$, where $d_1 \in \mathbb{R}$ represents the (non-negative) propagation delay at t_1, may still be blocked by celestial bodies.

We calculate the visibility window information for our scenarios together with distance information using STK. Afterwards, we process the flight-dynamics simulator output to derive a common input format for the topology generator. From this information, the topology generator computes the sequence of communication windows.

Snapshot Computation: Let $t, t' \in T$ be two time points. The topology generator computes a snapshot sequence from the communication windows provided as input for the time period $[t, t']$ as follows. Let V be the set of all nodes in the topology and let $W \subseteq T \times T \times (V \times V)$ be a set of communication windows within $[t,\ t']$. The snapshot generation algorithm takes W, t, and t' as input

and computes the sequence of snapshots $S = \langle S_0, ... \rangle$ and thus also the δ function that represents the topology evolution in $[t, t']$. Namely, for each communication window $(t_i, t_j, (a, b)) \in W$, it computes the respective edges (a, b) and (b, a) for the affected snapshot topology graphs $G_i, ..., G_{j-1}$ in the time period $[t_i, t_j]$.

Propagation Delay Metric: In the topology models for our two application scenarios, we use cost metrics that include scenario-specific cost factors, primarily delay. We extend the snapshot definition to store information on link-propagation delays by annotating each edge with a value denoting the delay.

6.3 Network Simulation

We use the generic network simulator ns-2 [11] for all our simulations. Since we are only interested in simulating the network or higher-layer behavior, we do not simulate the physical properties of the communication medium. Hence, our simulations are independent of the signal-transport medium at the physical layer, such as radio modulation or optical links. We can therefore use the standard ns-2 communication model with the following modifications. (1) Links have a delay label and nodes only recognize link changes after an idle time of twice the delay value for the affected link, representing the minimal round-trip delay. (2) Node connectivity information over time is taken from the snapshot-sequence output of the topology generator. Using this information, ns-2 can activate or deactivate links at transition points.

With this approach, we cleanly separate physical-medium properties and topology properties. This makes our simulations independent of the actual physical-layer media used in the network. Moreover, since the network simulator is separate from the flight-dynamics tools and the random-movement generator, no modeling of node movement is required in the network simulator. In our two application scenarios, this allows us to optimize the spacecraft constellation simply by making changes in the flight-dynamics tools and then directly observing their impact on the network model. In our simulations, we compare different properties of PLSR with those of LSR.[1]

LSR and OLSR: Optimized Link-State Routing (OLSR) [12] is an optimization of LSR well-suited for mobile ad-hoc networks. We do not experimentally compare PLSR to OLSR. OLSR is specialized to efficiently handle large, dense ad-hoc networks. Spacecraft networks have quite different characteristics. Namely, they consist of a small number of nodes that are sparsely distributed over a vast area, with dense concentrations only around planets. This leads to an asymmetric situation, where the topology may change rapidly around the planets while remaining stable for long time periods in other parts of the network. OLSR's enhancements, which benefit from dense, frequently changing topologies, do not work efficiently in such an environment and produce additional overhead. In our simulation summary, we return to this point and discuss the differences between LSR and OLSR in our evaluation scenarios.

Measurement Metrics: For all scenarios, our simulations measure routing message overhead and traffic throughput, which are the most relevant metrics for spacecraft networks. We do not measure the convergence time for unpredictable changes since the majority of the changes are predictable and therefore PLSR routing is still possible during convergence, even if routing is suboptimal. We have implemented the PLSR protocol, described in Algorithm 3, by adapting an implementation of the OLSR protocol for ns-2.

Snapshot Distribution: Our PLSR simulations include the upload of snapshot sequences from the ground station to spacecraft. In our simulations, transition information on each snapshot is encoded in 15 bytes of data. Since the number of snapshots and the required flooding messages differ for each simulation topology, the traffic overhead related to snapshot distribution also varies.

Unpredictable Changes: In the random-movement setting, the vast majority of changes are predictable, but an unpredictable change occurs with a 10% probability within an entire simulation run. In the application scenarios, we use a more realistic approach to model unpredictable changes. Spacecraft and link failures of varying degrees of severity frequently occur on spacecraft due to technical malfunctions or environmental problems (caused by temperature, radiation, and the like). These failures can affect nodes' routing capabilities. Most failures are non-critical in practice and we assume that failures rarely occur and are quickly repaired. We modeled the failure cases based on failure events that currently occur in existing spacecraft operations.

7 Random Movement Scenario

We now assess the performance of the PLSR protocol in a simulation setting with random node movement, independent of any specific spacecraft-network scenario. For these experiments, our topology generator simulates the random movement of nodes as described in Section 6.1. This random topology setting emphasizes the generic properties of the PLSR protocol but also leads to a large confidence interval (i.e. topologies with extremely low or high measurement results may exist). We therefore performed many simulation runs (each 10000 seconds of simulated time) for all configurations from Section 6.1 and averaged the results. We simulated both protocols, LSR and PLSR, on the same topologies. We performed simulation runs for random, connected topologies with 5, 10, 15, and 20 nodes. Note that these are realistic sizes for typical spacecraft networks.

1. For LSR, our starting point was the implementation distributed with ns-2. This implementation is for static networks and therefore does not address the failure-detection latency caused by the propagation delay. We therefore adapted the protocol implementation to include this latency. We further disabled periodic topology discovery messages since they are of limited use in the LEO/GEO inter-spacecraft network and only increase the routing overhead of LSR.

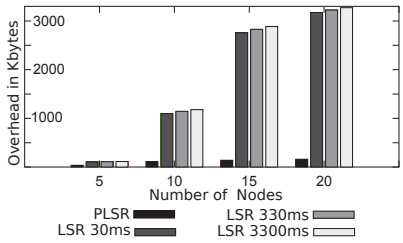

Fig. 2. Random Setting Routing Overhead

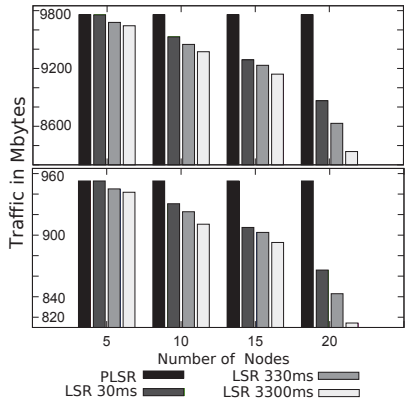

Fig. 3. Random Setting Traffic Throughput

Routing Overhead: We measured the overhead produced by PLSR and LSR. Figure 2 shows our results. Note that PLSR produces little overhead in all cases. The overhead produced stems from the snapshot updates that are required for PLSR. Although the size of these updates grows with the number of nodes, the growth is minimal and far below the growth rate of LSR overhead. LSR shows a strong increase of traffic overhead caused by flooded LSA messages. With 20 nodes, LSR already produces around 3 MBytes of traffic overhead. This would continue with an increasing number of nodes.

Traffic Throughput: Figure 3 shows the traffic throughput measurements for the two packet-production rates of 20 and 200 packets per second. The figure shows that PLSR provides constant maximum throughput since it has access to topology information and can reroute packets before a link actually goes down. Therefore, it is not impacted by variations in the propagation delay. We also see that the traffic throughput supported by LSR is reduced as the number of nodes increases. This behavior is similar for the different packet-production rates. The throughput degradation results from the fact that the number of changes increases with the number of nodes and traffic is lost during routing table synchronization of the nodes. Finally, we see that a high propagation delay causes more packets to be lost since the LSA messages need longer to reach the nodes.

Unpredictable Changes: We measured the effect of unpredictable changes on the PLSR protocol. We simulated LSR and PLSR for topologies with 10 and 20 nodes, using the same configurations as above. In addition, a number of unpredictable changes occur in each configuration. Figure 4 shows our results. In a 20 node topology, PLSR performs substantially better than LSR, even with many unpredictable changes. In fact, PLSR's overhead with less than 5 unpredictable changes is negligible. Note that we did not make measurements for more than 20 changes in the 10 nodes setting.

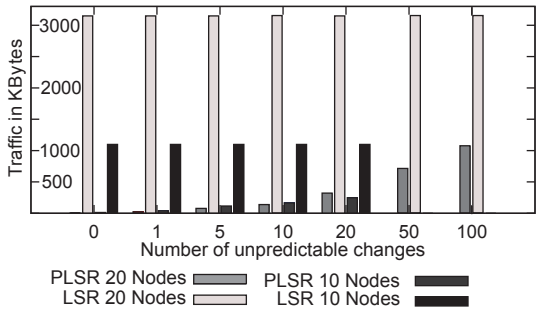

Fig. 4. Impact of Unpredictable Changes

8 LEO/GEO Spacecraft Hybrid Network

LEO spacecraft are increasingly used by Earth observation missions. Their high-resolution measurements generate substantial quantities of scientific payload data that must be sent to Earth. This is problematic as, due to their low orbit, LEO spacecraft have only small communication windows with ground stations.

The small communication windows constrain the direct downlink capacity of LEO spacecraft for data transmission to ground stations and result in a communication bottleneck. To address this problem, space agencies are examining the use of GEO spacecraft as stationary relay points. The GEO spacecraft will have long communication windows with the LEO spacecraft as well as continuous connectivity with the ground stations. This eliminates the communication bottleneck and provides a continuous failure-resistant, data-stream downlink for payload data from LEO spacecraft. This new, more complex, topology requires a fast switching, efficient routing scheme. It cannot be managed using traditional, manually commanded, point-to-point communication.

8.1 Properties and Infrastructure

We describe the participating entities and links that are present in the LEO/GEO scenario.

- **LEO spacecraft:** Five LEO spacecraft orbiting the Earth at ca. 600 km. They maintain links to all visible GEO spacecraft. Additional inter-LEO spacecraft communication can be used.
- **GEO spacecraft:** Three spacecraft in a geostationary orbit.
- **Ground Stations:** Three interconnected ground stations, one of which acts as a traffic sink.

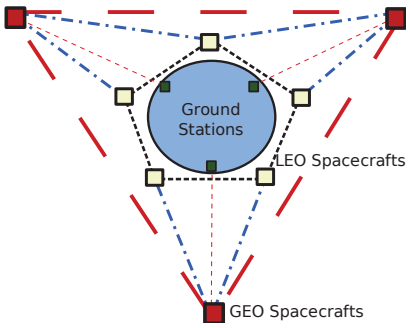

Fig. 5. LEO/GEO Network

We assume that all entities employ the same packet-switched networking protocol. LEO/GEO spacecraft networks are highly mobile, which results in a large number of snapshots. Furthermore, each ground station has constant visibility with at least one GEO spacecraft.

Figure 5 shows our constellation of the five LEO and three GEO spacecraft orbiting the Earth as well as some of the possible communication links. (For clarity, not all possible inter-LEO spacecraft links are shown.)

Unpredictable Changes: Should one GEO spacecraft fail and a continuous data relay no longer be possible, the LEO spacecraft in our model can use other LEO spacecraft in their communication range as data relays. This contingency situation provides a backup to avoid the loss of payload data.

8.2 Simulation

We simulate PLSR and LSR in the LEO/GEO scenario. We compute the topology model as described in Section 6 and then simulate the network dynamics as well as the behavior of PLSR and LSR in ns-2.

We do not compare PLSR to other, highly specialized, space-routing protocols since they are not generic and their use is limited to particular constellation setups.

For the LEO/GEO scenario, we choose a simulation time period of one day. This provides a reasonable number of snapshots (605). Also, the LEO spacecraft have repeated ground tracks (i.e., the trace of the satellite's path over the ground) after this time period.

Since we want to simulate the network behavior during the generation and transfer of spacecraft payload data, our traffic model contains five data streams, one originating from each LEO spacecraft. In ns-2, we implemented each stream as a CBR UDP traffic stream with a 100 byte packet transmitted every 50 ms. Note that the size of the traffic packets only affects the packet-to-bytes ratio but does not affect the simulation results as long as the packets are small enough to avoid fragmentation. The traffic stream starts at simulation time 1 second and stops at 86,300 seconds. For PLSR, we use the delay metric given in Section 6.2 to model the link costs.

Spacecraft Link Failures: We conduct measurements for nominal operations (that is, without any spacecraft failures) and for three failure configurations:

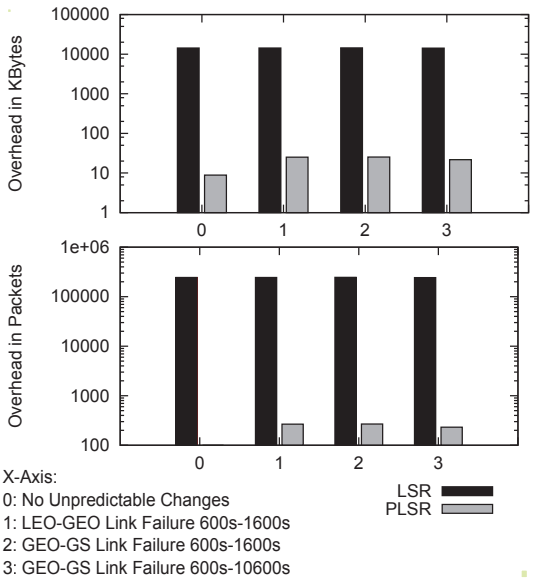

Fig. 6. LEO/GEO Network Routing Overhead

1) The failure of a LEO-GEO spacecraft link, which lasts for a duration of 1,000 seconds (≈ 17 minutes)
2) The failure of a GEO spacecraft to ground station link for a period of 1,000 seconds
3) The failure of a GEO spacecraft to ground station link for a period of 10,000 seconds (≈ 2.7 hours)

8.2.1 *Routing Overhead*

We compare PLSR's routing protocol overhead to that of LSR. Figure 6 shows the overhead measurements for nominal operations and the three failure configurations. The x-axis shows the simulation configurations and the y-axis shows the traffic overhead on a logarithmic scale.

As was expected, by exploiting network predictability, PLSR has a clear advantage over LSR. This advantage is largest for the simulation without unpredictable changes, where LSR produces about 2,000 times more packet overhead and 1,000 times more byte overhead. In this non-failure case, PLSR produces only the minimal overhead required to upload snapshot sequences and no overhead related to the distribution of LSA messages.

Impact of Failures: As seen in Figure 6, for all three failure configurations, unpredictable failures cause a slight increase in PLSR's routing overhead. This comes from the need to communicate these changes to all nodes. Still, PLSR's advantage is substantial.

For the failure configurations (2) and (3), the duration of an unpredictable failure does not significantly impact the overhead produced by PLSR. This is because PLSR stores the currently valid unpredictable changes at each node. At snapshot transition points, the nodes modify their new LSDBs with the unpredictable changes that they have currently stored. If an unpredictable change is invalidated by a predictable change or another unpredictable change, the information is deleted from the nodes. This obviates the need for generating additional

LSAs after each transition point. Also, once information on an unpredictable change has propagated through the network, no additional messages occur until another unpredictable change occurs.

The slight drop in PLSR routing overhead in configuration (3) results from the minimal network connectivity at the time when the link comes back up again. Fewer links are active at this time and therefore fewer LSA messages are required during flooding.

8.2.2 Traffic Throughput

We measure the stability of the payload-data streams and, in particular, the number of traffic packets and bytes successfully transmitted from the LEO spacecraft to the sink ground station. We conduct these measurements in the nominal (non-failure) situation as well as in the three failure configurations. Figure 7 shows our results.

In all configurations, PLSR has a higher throughput than LSR, although the differences are not substantial. The additional loss of packets in LSR only occurs in the time between a change and its discovery through link sensing as well as during the time required for the LSA messages to propagate the change. Therefore, in the LEO/GEO spacecraft scenario, the difference in throughput performance can be explained by the fact that the communication delays between the spacecraft are low, the LSA messages in LSR will quickly reach all nodes, and the routing tables converge.

In the absence of unpredictable changes, we see that PLSR provides maximum throughput and no packets are lost. This is because link breakdowns are known in advance (through the use of communication windows) and PLSR can react before a link goes down and choose another link if a route is affected.

In real-world scenarios, the inter-spacecraft links would be bandwidth-optimized to achieve maximum link saturation. Additional traffic on these links, caused by routing overhead, thereby reduces payload-data throughput. Therefore, in reality, the total traffic throughput in LSR would actually be less since traffic packets are dropped in favor of LSA packets if the links are saturated. In PLSR, this effect only takes place for rare snapshot updates and LSAs associated with unpredictable changes, and is negligible in the measurements. Note also that, should the control and data transfer links be separate, which is often the case in practice, the low bandwidth (4-6 Kbit/s) control links will be much more affected by the routing overhead.

9 Mars Network Communication

Rovers will play an increasingly important role in exploring the solar system and Mars in particular. They gather scientific data and communicate frequently with the Earth. Rovers have limited energy and computing resources. Their communication sessions and data transfer opportunities must therefore be optimized. This option introduces a topology complexity that challenges the

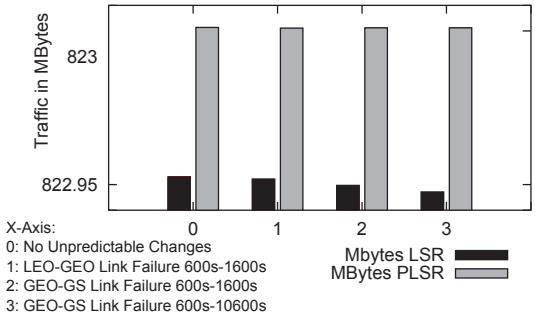

Fig. 7. LEO/GEO Network Traffic Throughput

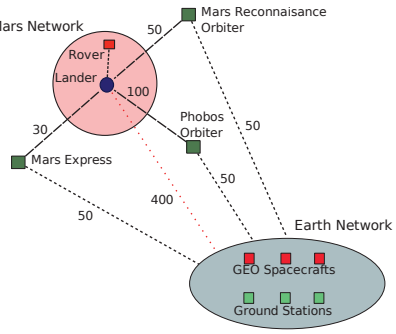

Fig. 8. Mars Network

manual commanding of point-to-point data transfer and requires an autonomous, fast reacting routing mechanism. While direct communication with a large antenna on Earth is possible, the costs are high both in terms of energy consumption and monetary cost. Hence, other communication options are preferred. An attractive option here is to use existing science orbiters around Mars with larger computational and energy-related resources to relay data between the rover and Earth.

9.1 Properties and Infrastructure

We describe below the participating entities in our Mars rover scenario. Although the rover mission that we use is fictive, it is based on the ExoMars mission [13]. Moreover, the Mars infrastructure components use assets that are part of present or planned missions. Note that the Mars orbiters that participate in the network have very different orbital configurations and therefore illustrate well the heterogeneity of spacecraft networks. The overall communication infrastructure is illustrated in Figure 8 and includes the following entities:

- **Mars rover:** The rover is mobile and maintains a link with the base station.
- **Lander base station:** The base station is used as a communications relay. It maintains links with the rover, the deep-space ground station, and all spacecraft orbiting Mars (when visible). Its direct link to Earth is costly in terms of energy.
- **Mars orbiters:** These orbiters maintain links to the Earth orbiting GEO spacecraft (if applicable, otherwise directly to the ground stations) and to the

lander.

- **Earth GEO constellations and ground stations:** The GEO spacecraft network is identical to the one used in the LEO/GEO spacecraft-network scenario (from Section 8.1) and is only present in two of the simulation configurations.

Mars networks have two additional special constellation-specific properties that we must also account for. The large distance between Earth and Mars causes long propagation delays on the interplanetary links. These delays lead to substantial deviations between the visibility and communication windows. This also delays the distribution of unpredictable changes and snapshot updates. Also, while the rover course follows a pre-determined path, its predictability is limited to approximately one Mars day. Therefore, frequent snapshot updates are required if connectivity is affected by rover movements.

In the Mars rover scenario, we use a more complex cost function than for the LEO/GEO scenario. The communication cost accounts for energy as well as actual monetary costs, such as renting a deep-space ground station or using the spacecraft of another agency as relays. We have approximated the resulting communication costs as shown in Figure 8. We also consider different failure configurations that may occur in real Mars operations. In particular, we investigate the impact of temporarily loosing the ability to communicate between a Mars orbiter and the Earth GEO network.

9.2 Simulation

In the Mars rover scenario, we choose a simulation time period of seven days. This period is longer than in the LEO/GEO scenario. The reason for this is since topology changes occur less frequently, paths between the rover and Earth may not always exist and longer propagation delays are present. As before, we assume that a sending antenna will automatically track the receiving antenna. We measure the routing overhead of the PLSR and LSR protocols and also the traffic throughput between the rover and the sink ground station.

All nodes run the routing protocol, including the ground stations and the Mars ground-based entities. We use two nominal operations configurations and one failure configuration for our throughput measurements:

1) A full network, including all entities described above with no unpredictable failures.
2) A reduced network without the Earth GEO spacecraft. Here, the ground stations on Earth communicate directly with the Mars orbiters. This configuration is used today for communication with Mars.
3) As in (1), but with 10 minutes Mars orbiter failure.

The Mars rover produces a continuous stream of CBR

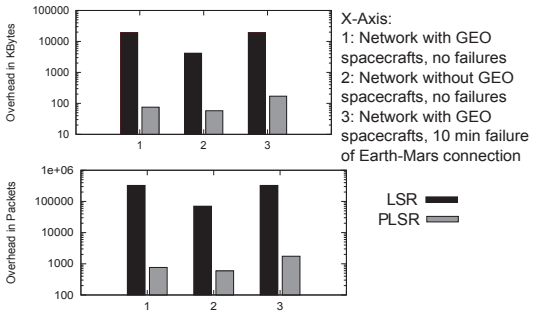

Fig. 9. Mars Network Routing Protocol Overhead

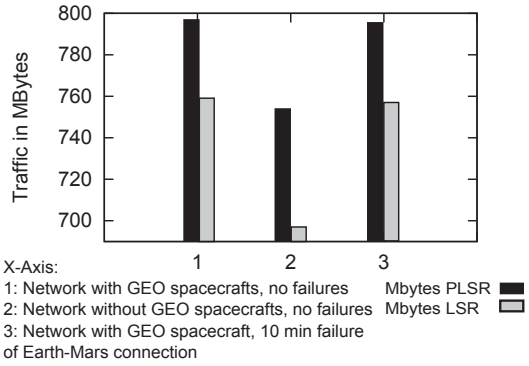

Fig. 10. Mars Network Traffic Throughput

UDP traffic packets (100 bytes, 50 ms interval).[2] The traffic stream starts at simulation time 1 second and stops at simulation time 600,000 seconds.

9.2.1 Routing Overhead

We compare PLSR's overhead, in terms of packets and bytes, to that of LSR. We perform simulations for the configurations (1)–(3) above. The results are shown in Figure 9. As before, the x-axis shows the different simulation configurations and the y-axis the overhead in kilobytes and packets.

We see that the performance difference for routing overhead is comparable to our results for the LEO/GEO scenario. This is expected since, as in the previous scenario, LSR still initiates a flooding process for every change, independent of the propagation delay. Moreover, the convergence time, compared to the LEO/GEO scenario is substantially larger (around 12 minutes) due to the presence of long-delay links. This affects the traffic throughput, which we discuss below. Another major difference is the increased number of hops that is required for communication in nominal mode.

9.2.2 Traffic Throughput

We measured the total traffic throughput of PLSR and LSR in the Mars rover communication scenario for all

2. Note that since a route between the rover and the sink ground station does not always exist (for a total of 184,778 seconds in configuration (1) and for a total of 203,650 seconds in configuration (2)), the rover does not produce traffic during these time periods. This ensures that packet loss is only related to routing failures.

three configurations. Figure 10 shows the results.

In comparison to the LEO/GEO scenario, PLSR performs significantly better than LSR: around 5% in (1), 7% in (2), and 5% in (3). This is the case even though the number of snapshots in the Mars scenario is lower than in the LEO/GEO scenario. One reason for this increased performance benefit is the presence of large propagation delays, which impair the communication of LSR link-state advertisements. Furthermore, following a link change, LSR will only re-route traffic once it is fully aware of the change. Thus the transport of payload data may be subject to routing failures until the change has been propagated to all nodes on the traffic path.

The reason that the performance difference between PLSR and LSR is not substantially larger is because route switches (which affect the long delay links between Earth and Mars) are infrequent. Therefore, additional overhead due to LSA messages is also limited in LSR.

The results for configuration (3) show that unpredictable changes that affect the high propagation-delay link between Mars and Earth reduce the performance of PLSR since these changes require a long time to propagate.

9.3 Simulation Summary

We draw conclusions from our three simulation scenarios. In all configurations, PLSR is superior to LSR in terms of throughput and overhead. Furthermore, large propagation delays have a significant negative impact on the efficiency of LSR, while PLSR is only affected in the presence of unpredictable changes. The number of nodes and the density of snapshots have a negative effect on PLSR, while LSR is not affected. Thus, we can expect that the advantage of PLSR with respect to LSR will further increase in more complex application scenarios.

Efficiency of OLSR: In all three scenarios, OLSR enhancements would not increase the performance of LSR. Due to the small number of sparsely distributed nodes, combined with fast link switching, multipoint relaying does not enhance LSR's efficiency. Furthermore, periodic LSA distribution, as used in OLSR (in contrast to instant distribution of LSAs upon detection of a link change), increases the packet loss in sparse networks.

10 Related Work and Conclusions

10.1 Related Work

While we use the snapshot approach, other approaches exist to model dynamically changing topologies. Borrel et al. [14] investigate Delay Tolerant Network (DTN) network classifications and introduce a notion of evolving graphs, which are essentially equivalent to snapshot sequences. Shao et al. [15] also discuss routing approaches for DTN networks based on evolving graphs, where a routing path between two nodes does not always exist and packets must then wait at intermediate nodes for links to become available. A similar approach is taken by Merungu et al. [16]. Snapshot sequences, however, provide better possibilities for decomposing the topology evolution into information that can be distributed by the ground stations. Ferreira *et. al* [17] present a combinatorial model for MANETs along with approaches for solving different routing-over-time problems. In deep-space communication, long propagation delays and intermittent connectivity links may be present in the network. The DTN approach [18] addresses this problem by introducing an overlay layer. DTNs are compatible with the connectionless packet-switched network that we assume. In [19], Jain *et. al* discuss routing problems in DTNs. While their focus is on routing with finite buffers, they also address DTN routing and flow control with complete knowledge. They do not, however, address unpredictable failures and their impact on the topology snapshots.

Several routing protocols have been proposed for Earth-orbiting spacecraft constellations, including datagram routing [20], optimal topological design [21], and routing tailored to Asynchronous Transfer Mode technology [22]. All of these protocols are limited to LEO constellations and have numerous architectural restrictions (such as a fixed constellation). These are based on the properties of LEO spacecraft networks that we introduced in Section 8.1. They lack the level of flexibility required in modern heterogeneous spacecraft networks.

In [23], the authors describe their ASCoT routing mechanism which provides a position-based routing architecture for space networks. Similar to the PLSR protocol, ASCoT exploits predictability. In contrast to PLSR, ASCoT treats all nodes as equal in terms of routing protocol behavior and proactively propagates connection information on current and future links between nodes. This increases ASCoT's overhead compared to PLSR.

10.2 Conclusions

We have presented a model for predictable mobile topologies and used it to design the PLSR routing protocol. Our protocol is correct and performs well compared to other routing protocols in an topology that has been generated using a random-waypoint model.

We have carried out the first detailed study of predictable routing for space networks. Using realistic simulations based on two application scenarios, we showed that PLSR is efficient and offers advantages over competing protocols. Our simulations are based on actual flight-dynamics data and show the superiority of PLSR over LSR. Together with the generic simulations, our results provide strong evidence of PLSR's general usability.

As future work, we would like to utilize additional scenario-specific factors to further optimize routing using PLSR, for example by accounting for the rover's energy budget or antenna-pointing information. With such extensions, PLSR would also support services for other layers than the network layer (such as the DTN bundle layer). For the Mars rover scenario, this would

allow the autonomous adaptation of the rover to its changing environment. Finally, since space assets are part of a critical infrastructure, we will also investigate how to best integrate security services with PLSR.

REFERENCES

[1] B. Evans, M. Werner, E. Lutz, M. Bousquet, G. E. Corazza, G. Maral, R. Rumeau, and E. Ferro, "Integration of satellite and terrestrial systems in future multimedia communicaitons," *IEEE Wireless Communications*, vol. 12, no. 5, pp. 72–80, October 2005.

[2] "Space Communications Architecture Working Group. NASA Space Communications and Navigation Architecture Recommendations for 2005-2030," NASA Technical Report, Washington D.C., May 2006.

[3] "Analytical Graphics Inc. Satellite Tool Kit Fundamentals," AGI Technical Report, Exon, PA, November 2007.

[4] R. J. Leopold and A. Miller, "The iridium communications system," *Potentials, IEEE*, vol. 12, no. 2, pp. 6–9, April 1993.

[5] D. P. Patterson, "Teledesic: a global broadband network," in *Proc. of the Aerospace Conference, 1998. Proceedings., IEEE*. IEEE, March 1998, pp. 547–552.

[6] V. V. Gounder, R. Prakash, and H. Abu-Amara, "Routing in LEO-based satellite networks," in *Wireless Communications and Systems, 1999 Emerging Technologies Symposium*. IEEE, April 1999, pp. 22.1–22.6.

[7] AtmelCooperation, "Tsc695f sparc 32-bit space processor - user manual," 2003.

[8] J. M. McQuillan, I. Richer, and E. C. Rosen, "The new routing algorithm for the ARPANET," IEEE/ACM *Transactions on Communications*, vol. 28, pp. 711–719, 1980.

[9] D. Fischer, D. Basin, and T. Engel, "Topology dynamics and routing for predictable mobile networks," in *Proceedings of the 16th International Conference on Network Protocols (ICNP), 2008*. IEEE, October 2008, pp. 207–217.

[10] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. New York, NY, USA: ACM, 1998, pp. 85–97.

[11] S. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd, P. Haldar, M. Handley, A. Helmy, J. Heidemann, P. Huang, S. Kumar, S. McCanne, R. Rejaie, P. Sharma, K. Varadhan, Y. Xu, H. Yu, and D. Zappala, "Improving simulation for network research," Technical Report, University of Southern California, March 1999.

[12] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized link state routing protocol for ad hoc networks," in *Proceedings of the 5th IEEE Multi Topic Conference, INMIC 2001*. IEEE, 2001, pp. 62–68.

[13] P. Baglioni, R. Fisackerly, B. Gardini, G. Gianfiglio, A. L. Praider, A. Santovincenzo, J. L. Vago, and M. Winnendael, "The Mars Exploration Plans of ESA," *IEEE Robotics & Automation Magazine*, vol. vol. 13, no. 2, pp. 83–89, June 2006.

[14] V. Borrel, M. H. Ammar, and E. W. Zegura, "Understanding the wireless and mobile network space: a routing-centered classification," in *CHANTS '07: Proceedings of the second ACM workshop on Challenged networks*. ACM, August 2007, pp. 11–18.

[15] Y. Shao and J. Wu, "Understanding the tolerance of dynamic networks: A routing-oriented approach," in *Proceedings of the 28th International Conference on Distributed Computing Systems Workshops, 2008. ICDCS '08*. IEEE, June 2008, pp. 180–185.

[16] S. Merugu, M. Ammar, and E. Zegura, "Routing in space and time in networks with predictable mobility," Georgia Institute of Technology, GA, Tech. Rep. GIT-CC-04-07, 2004.

[17] A. Ferreira, "Building a reference combinatorial model for manets," *Network, IEEE*, vol. 18, no. 5, pp. 24–29, October 2004.

[18] K. Fall, "A delay-tolerant network architecture for challenged internets," in *SIGCOMM '03: Proceedings of the 2003 Conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2003, pp. 27–34.

[19] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications. ACM, 2004, pp. 145–158.

[20] E. Ekici, I. Akyildiz, and M. Bender, "Datagram routing algorithm for LEO satellite networks," in *Proceedings of INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*. IEEE, August 2002, pp. 500–508.

[21] H. Chang, B. W. Kim, C. G. Lee, Y. Choi, S. L. Min, H. S. Yang, and C. S. Kim, "Topological design and routing for low-earth orbit satellitenetworks," in *Proceedings of the Global Telecommunications Conference, 1995. GLOBECOM*. IEEE, November 1995, pp. 529–535.

[22] M. Werner, "A dynamic routing concept for ATM-based satellite personal communication networks," *IEEE Journal on Selected Areas in Communications*, vol. vol. 15, no. 8, pp. 1636–1648, October 1997.

[23] O. Gnawali, M. Polyakovt, P. Bose, and R. Govindan, "Data-centric, position-based routing in space networks," in *Proceedings of the Aerospace Conference, 2005*. IEEE, March 2005, pp. 1322–1334.

Daniel Fischer is with the European Space Agency where he is working as a Data Systems Engineer. He received his Ph.D. from the University of Luxembourg in 2010. His research interests are satellite communication, security protocols and space networking. He is working on the specification of security protocols for ESAs satellite communication systems.

David Basin is a full professor and has the chair for Information Security at the Department of Computer Science, ETH Zurich since 2003. He is also the director of the ZISC, the Zurich Information Security Center. He received his Ph.D. from Cornell University in 1989, and his Habitation from the University of Saarbrcken in 1996. His research focuses on information security, in particular methods and tools for modeling, building, and validating secure and reliable systems.

Knut Eckstein graduated from the University of Stuttgart with an aerospace engineering degree in 1993 and doctoral degree (Dr.-Ing.) in 1997 for his research on high-performance computing for non-linear computational mechanics. After four years in consulting industry as a network security analyst and team leader, he joined the NATO C3 Agency to work on Network Security R&D. Since 2006, he works at the European Space Agency as a System Security Engineer. Dr Eckstein is a member of the ACM.

Thomas Engel is Professor for Computer Networks and Telecommunications at the University of Luxembourg. He studied Physics and Computer Science at the University of Saarbruecken, where he also received his Ph.D. in 1996. Prof. Dr. Engel is member of the European Security Research Advisory Board (ESRAB) and member of the Security Taskforce of the European Commission. He also is the coordinator of the European Integrated Project u-2010 on Next Generation Networks.