Characterizing Topological Bottlenecks for Data Delivery in CTP using Simulation-Based Stress Testing with Natural Selection

Doina Bucur

University of Groningen, The Netherlands

Giovanni Iacca

INCAS^3, The Netherlands

Pieter-Tjerk de Boer

University of Twente, The Netherlands

Abstract

Routing protocols for ad-hoc networks, e.g., the Collection Tree Protocol (CTP), are designed with simple node-local behaviour, but are deployed on testbeds with uncontrollable physical topology; exhaustively verifying the protocol on all possible topologies at design time is not tractable. We obtain topological insights on CTP performance, to answer the question: Which topological patterns cause CTP data routing to fail? We stress-test CTP with a quantitative testing method which searches for topologies using evolutionary algorithms combined with protocol simulation. The method iteratively generates new test topologies, such that the execution of the protocol over these topologies shows increasingly worse data-delivery ratios (DDR). We obtain a large set of example topologies of different network sizes up to 50 nodes, network densities, data rates, table sizes, and radio-frequency noise models, which, although connected, trigger a data delivery of nearly zero. We summarize these topologies into three types of *topological problems*, the root cause of which is the presence of certain asymmetric links and cycles, combined with a certain size of the routing table. We verify causality, i.e., show that randomly generated topologies having these particular features do cause low DDR in CTP. This testing methodology, while computationally intensive, is sound, fully automated and has better coverage over the corner cases of protocol behaviour than testing a protocol over manually crafted or random topologies.

Keywords: Routing, Collection Tree Protocol, Performance evaluation, Data delivery ratio

1. Introduction

This paper aims to answer the following questions:

(Low performance) How low can *data-delivery ratios* (DDR) fall for a given ad-hoc *collection routing* protocol? Can physical topologies be created such that the calculation of the shortest-path tree unexpectedly does not converge? We focus on the TinyOS [14] implementation of the Collection Tree Protocol (CTP) [10], and small-to-medium-size connected networks of various network densities, link (a)symmetry, data rates, noise models, and other deployment variables.

(Data sets) Can a testing method search through the theoretical domain of connected topologies effectively, and locate sets of test topologies over which CTP exhibits DDRs close to 0% ?

Email address: `d.bucur@rug.nl` (Doina Bucur)

(Predictive topological metrics) Can a general, predictive model be made for topologies of low DDR?

(Protocol fixes) If problematic topologies are found, can simple fixes be demonstrated? Note that here we focus on the testing procedure, and not on protocol redesign or methods for topology control.

Our method is akin to stress-testing a protocol. It is experimental, and based in *simulation coupled with a stochastic optimization algorithm*. It is standard to use optimization techniques to *construct* the optimum design of a system: to find that path on which a mountain can be climbed such that the path minimizes a given cost function, e.g., climbing time. In our method, optimization is used, on the contrary, to *destruct* the system, i.e., to search for that particular mountain which a given climber cannot conquer. In this metaphor, the climber is the protocol under study, and the mountain is a network configuration (i.e., physical topology) over which the protocol has low performance. We obtain a large number of such "mountains" as a set of evidence for low performance (or counterexamples for good performance), which enables us to form conclusions about the weaknesses of the protocol.

The object of this study is an ad-hoc collection routing protocol. Such protocols are designed to organize networks of wireless computing nodes into a collaborative, data-collecting system with good global performance, such as data delivery ratio or energy consumption. We focus the study on CTP and its main implementation for TinyOS; CTP is among the most successful protocols for collection routing [10]. As performance factor, we evaluate its data delivery. Of a relatively recent design, CTP is stated to reach its *reliability* goal, i.e., "99.9% delivery in a network with high quality links", and its *efficiency* goal, i.e., it achieves reliability and robustness "while sending few packets", according to evaluation in a number of testbeds of high-quality topologies, including topologies with some link asymmetry [10].

On the other hand, recent reports from deployments of collection routing protocols (CTP and precursors, e.g., MintRoute [26], MultiHop, and MultiHopLQI [24]) also encountered poor performance which could specifically be attributed to the behaviour of the routing software. In general, it was stated that "despite providing a simple service that is fundamental to so many systems, and being in use for almost a decade, collection protocols can suffer from poor performance. [It] is unclear why collection performs well in controlled situations yet poorly in practice, even at low data rates" [10]. This is not surprising: it is difficult to design a distributed data-routing algorithm which provides a guaranteed performance threshold for *any* given network configuration. This is due to the difficulty of *exhaustively* testing (e.g., 20 nodes connected by stable asymmetric links in a network of 50% density can form 10^{113} different topologies).

Importantly, the GreenOrbs and CitySee large-scale testbeds obtained concrete evidence of the negative effect of critical bottleneck nodes and links upon the effective network connectivity, and thus upon the data delivery ratios from nodes in disconnected areas of the network [17, 18]. Our work brings further theoretical evidence for this negative effect of bottlenecks.

Summary of contribution and article structure. We update an evolutionary framework for protocol testing first presented in [5]; the framework is based on the principle of natural selection, and is presented in Section 3. This study provides the following new knowledge of CTP performance:

- We experimentally obtain topological evidence of low delivery ratios for CTP in networks of up to 30 nodes, of different network densities, symmetric or asymmetric links, different data rates, sizes of the neighbour and routing tables, and dynamic environmental noise. We find that topologies with symmetric links and high signal-to-noise ratio do maintain high DDR. However, there exist connected topologies with bottleneck nodes and a certain pattern of link asymmetry which pose problems to CTP; over these, the outcome is heavily *probabilistic*, with a high chance that DDR is close to zero. The cause of this outcome is to a large extent the overloading of crucial bottleneck nodes with heavy beacon traffic from neighbour nodes situated in unresolved routing cycles. In the presence of heavy RF noise, less surprisingly, bottleneck links of low signal-to-noise ratio will lower DDR to zero, regardless of the symmetry of the links.

- We experimentally obtain large sets of test topologies which are evidence for low DDR performance for CTP. The experiments use our evolutionary framework and are described in Section 4. We share these test sets.

- We analyze, using fine-grained logging, the execution of CTP over those topologies which trigger low delivery (in Section 5), and give examples of bottleneck situations causing heavy data retransmission and thus a degradation of link cost estimations, which leads to incorrect routing choices. We briefly discuss possible fixes. We observe the topological patterns which are the root causes, formalize them via graph metrics, and verify that these metrics are sufficient causes to "break" the protocol (in Section 6).

Compared to previous work evaluating the performance of collection routing, the advantages of our method are: (i) The evidence of poor protocol performance is found here purely via computational rather than costly deployment means, and fully automatically, at protocol design time. (ii) A larger, theoretical body of evidence is obtained in comparison with recent studies such as [17, 18], which first located topological bottlenecks in CTP over a large-scale, long-term testbed. Here, we show that bottlenecks may also occur in small networks, that CTP performance outcome is heavily probabilistic, and that link asymmetry is the most dangerous factor in overloading nodes. (iii) We quantify topological metrics to generalize from the performance evidence obtained.

2. Background on performance evaluation for ad-hoc collection routing

Collection routing [9] designs a best-effort, shortest-path multihop routing of individual, unaddressed datagrams from any node in the network to any root (or *sink*) node of the network. For a collection protocol, a *functional* performance metric assesses the ability of the protocol to deliver data packets to the sink. The *data delivery ratio* (DDR) is the main functional performance metric; it is calculated as the fraction of data packets sent by all non-sink nodes received at the sink node at least once. High DDR over 90% is reported in the literature for CTP from testbed evaluations [10].

2.1. Distance-vector routing: theoretical guarantee of convergence under certain assumptions

Collection routing protocols for wireless sensor networks (WSNs) implement distance-vector routing, which aims to obtain *shortest paths* to any sink node. A node chooses the first node on this path (its so-called *parent*, or next-hop node), based on an assessment of the routing *cost* (or *distance*) of forwarding the packet to the sink via the path starting in that parent. The set of shortest paths forms one *collection tree* per network sink.

The basic mechanism of distance-vector routing is a distributed, asynchronous Bellman-Ford algorithm. Each node maintains a *neighbour table* memorizing the cost of direct outgoing links to all neighbours. An entry in the neighbour table $j \in \text{neigh}(i)$ of node i has the form $\langle j : c_{ij} \rangle$, where node j is reachable by i with a single transmission, and c_{ij} is the cost of a data transmission from i to j. Assessing the cost of individual links in the network is done by a *link estimator* logic using one-hop broadcasts of routing packets (called *beacons*) from each node in the network.

Each node also maintains a *routing table*, with an entry for each sink; for node i, a given *sink* node has an entry of the form $sink : \langle j : C_j \rangle$, where j is the parent node for i on the path to $sink$, C_j is the total cost of the path $j \rightarrow sink$, and $C_i = c_{ij} + C_j$ is the total cost of the path $i \rightarrow sink$. To construct such routing table entries, the beacon frames contain the sender node's own routing table, which then the receiver node uses to update its own table: the next hop j is chosen to be that neighbour node with the smallest total path cost to the destination. Assuming positive link costs, a unique *sink* node, symmetric links, and a connected topology, the protocol initializes the path costs $C_{sink} = 0$ and $C_i = \infty, i \neq 0$, and runs the following logic *indefinitely*:

- receives cost estimates C_j from neighbour j,
- runs at all nodes $i \neq sink$ iterations of the distance-vector local computations: $C_i = \min_{j \in \text{neigh}(i)} \left(c_{ij} + C_j \right)$, using the latest received estimates C_j and the latest local estimate of the cost c_{ij}, and
- broadcasts the result C_i to neighbours.

Distance-vector routing has known weaknesses even under these assumptions: while, under limited dynamicity of link costs, the number of algorithm iterations until convergence is finite, this number of iterations (and the amount of routing traffic ensued) become excessively large in certain topologies and locations of the link change. *Transient cycles* are then naturally formed by the distributed Bellman-Ford algorithm and subsequently solved. In the worst case, the number of broadcasted routing packets is on the order of LN^3, with N is the size of the network, and L the largest change in a link cost, and this was shown to occur over a small number of example topologies for the basic algorithm [4]. The problem is known as *counting to infinity*, and is the main performance issue in classic distance-vector protocols such as CTP. In this study, we aim to obtain and summarize extensive topological knowledge showing counting to infinity in a practical, feature-rich implementation of the basic Bellman-Ford.

2.2. *CTP design*

CTP [10, 8] does not implement specific mitigation strategies for the general topological problems of distance-vector routing; we expect similar problems when stress-testing CTP, particularly when we add the new factor of link *asymmetry*. CTP does improve a number of core features compared to the basic distance-vector algorithm described in Sec. 2.1; we list these below.

CTP link costs and neighbour table. The cost metric c_{ij} of a link in CTP is relatively sophisticated: the number of *expected transmissions* (multiplied by 10) for a packet to be received successfully – denoted *ETX*. An ETX is initialized with a very large estimate, after which it is refined: an ETX of 10 models a perfect link in which a single transmission is needed at one end for successful reception at the other end, while an ETX of 70 models a link on which a total of seven transmissions must be made for one successful reception.

Link ETXs are assessed, at each node, by a link estimator module external to the routing protocol. This estimator itself maintains a *neighbour table* of a preprogrammed size. ETX estimates are computed separately from two sources:

- For data unicasts, the ETX estimate is $10 \cdot \frac{w_u}{a}$, where a is the number of unicast packets acknowledged at the link layer, out of a window $w_u = 5$ of unicast packets sent by the node. If a equals 0, then the fraction is replaced by the number of failed data deliveries since the last successful one. This is a measure of the bidirectional quality of a link.

- For beacon broadcasts, the ETX is estimated in a similar way over a window $w_b = 3$ of received beacons; the link estimator determines whether any beacons failed to be received by searching for gaps in the sequence numbers of the beacons received from that particular neighbour. This ETX estimate only measures a directional quality of a link.

The two estimates are averaged into a single ETX value, which is reported as the ETX cost of a link. In heavy data traffic (which includes data retransmissions when acknowledgements have not been received), bidirectional ETX estimates may dominate; otherwise, directional estimates do.

Whenever a new ETX estimate is computed for the link to a neighbour, what is reported is not the new estimate, but a weighted average of the two, with the old estimate weighted at $\alpha = 90\%$; i.e., the ETX is an exponentially weighted moving average. A neighbour is evicted from the neighbour table when its ETX surpasses a threshold of 65. As standard for distance-vector protocols, the parent node is that node with the best path cost to the sink, i.e., the lowest *path ETX*.

CTP routing table. Besides including the path ETX C_j of a node j in beacon broadcasts, CTP also includes the identity of the parent node of j. Since a single parent node needs to be computed per sink at all nodes $i \neq sink$, the identity of the current parent is stored outside the routing table. Instead, this *routing table* of a preprogrammed size keeps entries for the best-quality neighbouring nodes, up to the size of the routing table. A routing table entry has the form $\langle j : l : C_j \rangle$, with node l the parent of node j; this means that the parent of each neighbour is also stored, having been obtained via beacons.

The management of the routing table is simple: a neighbour is only added after the link cost estimation is considered mature, and a neighbour is evicted only when the neighbour table kept by the link estimator also evicts that neighbour. A new parent is chosen only when the new best route is lower in path ETX than the current one by more than 1.5, and only if the link ETX to the parent is under a threshold of 50.

CTP beaconing. Beaconing is not periodic, but *adaptive* to the situation. When the topology is perceived as stable, the next beaconing interval used will be the double of the previous interval, up to a maximum period of 512 s. On the other hand, when a "problem" is perceived in setting up the collection tree, the beaconing interval is reset to its minimum of 64 ms. The problem events which reset this interval are:

- The boot of a neighbour node, when its routing table is empty; the neighbour broadcasts beacons with a "pull" bit set, requesting fresh updates.

- When the path ETX of a node drops significantly, it beacons to update its neighbours.

- Most importantly, when a *logical inconsistency* is detected, i.e., when a node receives a data packet to forward up the routing tree from another node with a smaller or equal path ETX. This method is called *datapath validation* for the routing tree.

A beacon-suppression technique is added, in place of a hold-down timer: each node counts the number of beacons received since its own last beacon transmission; if this number surpasses a threshold, then the node does not transmit a beacon. The value of the threshold was found via some testbed experimentation to not affect DDR significantly [10].

CTP data forwarding. It is the logical inconsistency of the routing tree, discovered via datapath validation, which generally signals to a node the presence of a routing loop. In a bid to increase the data delivery ratio, a data packet which is seen as looping is not dropped (as other protocols, such as MultiHopLQI, do), but is forwarded on the loop after a delay equal to the minimum beacon interval, such that the data packet is preceded by a beacon, which should trigger a topology update. In cases of highly dynamic networks, data packets may end up circling through multiple loops, hopefully repairing the routing tree, until at some point the node with an incorrect choice of parent chooses a valid next hop, and the data packet can be delivered.

Some of these features of CTP, particularly adaptive beaconing, can work against the protocol on certain topologies; we see this in Section 5.

2.3. Topological asymmetry. Protocol evaluation via simulation: stochasticity, environmental interference

We need a means to measure DDR over networks of different size, (a)symmetry of communication links, density, topology, link intermittence, duration of the experiment, and other variables related to the configuration of the protocol itself (such as the rate of issuing data packets at each node, the size of the tables, etc). In our study, we use TOSSIM [15] to set up controlled experimental networks of required size, link symmetry, density, topology, noise model, and duration, run an experiment over these settings, and subsequently quantify DDR.

Configuring a topology. TOSSIM allows an experimenter to define a network topology via its directional communication links and link gains. In simulation, the transmission power of all nodes is 0 dBm. A communication link between two nodes can theoretically have any gain above -110 dB (a threshold particular to the modelling of the network layer in TOSSIM). A viable directional link exists from node i to node j if it is assigned a gain above -110 dB; we investigate two cases: (a) idealized links modelled with a gain of 0 dB, which allows us to factor out link intermittence in favour of analyzing protocol logic, and (b) discrete gain values from across the domain of definition.

In the general case, a topology is *asymmetric*, i.e., the two directional links $i \to j$ and $j \to i$ need not have the same gain nor the same reception probability; in its extreme form, only one of the directions is a viable link, and the link is unidirectional. Link asymmetry may occur due to unequal transmission power at link ends, heterogeneity in transmission hardware, the hidden-terminal problem [19], or frequency mismatch between neighbouring nodes [16], and may be a persistent phenomenon, if certain nodes have depleted their energy supply or are located near a persistently strong interference source. Heavy asymmetry has been measured in field trials [16]; nearly 10% of links had a difference in directional packet loss of over 50% in the testbed described in [28]. Routing over unidirectional links is problematic due to the lack of efficient MAC protocols which deal natively with asymmetric links; acknowledgement-based MAC protocols are inefficient in a unidirectional environment [21]. Routing protocols need to compensate for this; many distance-vector protocols assume that links are symmetric. CTP (due to its link cost estimation) is designed to deal well with asymmetry. In most of our tests, we study the extreme form of asymmetry, i.e., unidirectionality.

The practical quality of a link in simulation may be further refined with the addition of intermittent radio-frequency noise, in which case the signal-to-noise ratio at a point in time will determine the state of the link.

In Table 1, we summarize the simulation settings used throughout the following sections. When multiple options are shown for a variable, they will be analyzed in different experimental campaigns. A network density of 1/2 is that in which 50% of all theoretically possible distinct links in the network are viable links. The two tables (the neighbour and routing table) are kept to the same size, due to the similar management of the two in CTP.

We boot all nodes at simulation time 0; there is randomness in the network stack purposely designed for concurrent boots not to cause network collisions due to synchronicity. A simulation time of 200 seconds is more than sufficient to test routing convergence, i.e., for the iterative calculation of the routing tree to reach a correct solution. Based on existing performance studies, CTP is expected to converge and deliver data within 1 second after a topology change [10].

The statistics of independent simulations. Wireless networks are affected by a number of stochastic effects, including interference on the radio frequencies used for communication, interference among the nodes themselves, packet collision, etc. To model bursts of interference, a statistical noise model is added. This noise model is generated automatically from a sampled noise trace (e.g., measured in real-world deployments), with the generation algorithm

Table 1: Configuration of simulation variables. The application was compiled with the flags: `-I$(TOSDIR)/lib/net/` `-I$(TOSDIR)/lib/net/ctp` `-I$(TOSDIR)/lib/net/4bitle`.

Network parameter	Value
Physical layer	TinyOS 2.x model of the CC2420 radio
MAC layer	standard TinyOS 2.x full-power CSMA with unicasts acknowledged
Link quality estimator	4-bit link estimator
Network size	10, 20, 30, 50 nodes
Sink node	node 0
Network symmetry	asymmetric, symmetric
Network density	1, 3/4, 1/2, 1/4, 1/8
Gain on links	ideal (0 dB), discrete ({-100, -80, -60, -40, -20, 0} dB)
RF noise model	light (strength ≤ -100 dBm), heavy (strength ≤ -40 dBm)
Experiment duration	200 seconds, 1 hour
Rate of data packets	every 1 second, every 5 seconds
Size of neighbour and routing tables	5, 10, 20

based on Closest Pattern Matching [13]. In our tests, we use a heavy-noise trace (`meyer-short`) available with TOSSIM, and another trace of constant light noise. In what follows, we refer to these as the "heavy" and "light" noise models, respectively. Transient link failures are effectively injected by the heavy RF noise model.

Due to this stochasticity in both the protocol logic and the simulation, repeated simulations of the same simulation settings do not yield identical outcomes. Because of this, in all tests, we evaluate each network under study via repetitions of the simulation. All tests presented in Section 4 simulate each network 16 times; the further analysis in Section 5 does 100 simulations. Both report the outcome, e.g., the DDR value for that network, via the **sample mean** and 95% confidence interval around the sample mean.

3. A protocol testing method based on an evolutionary algorithm

We apply the principle of *natural selection* [7] to the problem of generating and evaluating test topologies for the network protocol under study[1]. Natural selection is an evolutionary process which evolves *populations* of *individuals*, such that those features giving individuals a quantifiable advantage will be preferentially propagated in successive *generations*. To test CTP, an individual is here a *network topology* which serves as a *test case* for CTP, and the aim of a test is to search for network topologies which pose performance problems to CTP. In this section, we summarize the way our methodology answers the questions posed in Sec. 1, and describe the evolutionary algorithm; the method is supported by an implementation including the evolutionary toolkit *µGP* [22], and the WSN simulator TOSSIM.

3.1. Summary of methodology

(Low performance). In a crucial first step, we use our evolutionary algorithm to evolve a population of network topologies such that, with every subsequent generation, we obtain topologies over which CTP has increasingly low DDR. We do this for various configurations of the network parameters given in Table 1, e.g., we test CTP over more than one network size, density, noise model, data rate, and size of tables. In comparison to a test which randomly generates the same number of network topologies as test cases, the evolutionary algorithm will achieve greatly improved *coverage*: it will not only obtain test topologies for the protocol of higher interest, i.e., which show significantly lower DDR than in the random case, but will also generate large numbers of these interesting test cases. The evidence for low DDR found with our framework (and the number of generations needed to obtain it) will depend heavily on the parameters of the test; for example, we expect that, if we constrain the test to only generate connected topologies with all communication links perfectly symmetric, the low DDR obtained will be somewhat higher than in a test which allows asymmetric links into the connected topology.

Fig. 1 gives a general overview of the iterative evolutionary mechanism. An initial population of μ individuals (i.e., network topologies encoded in an abstract form) is generated randomly. The format in which a network topology is

[1]A single-objective version of this evolutionary framework for protocol testing was presented in [5]. Here, we limit the description of the framework to core concepts necessary for a standalone presentation.

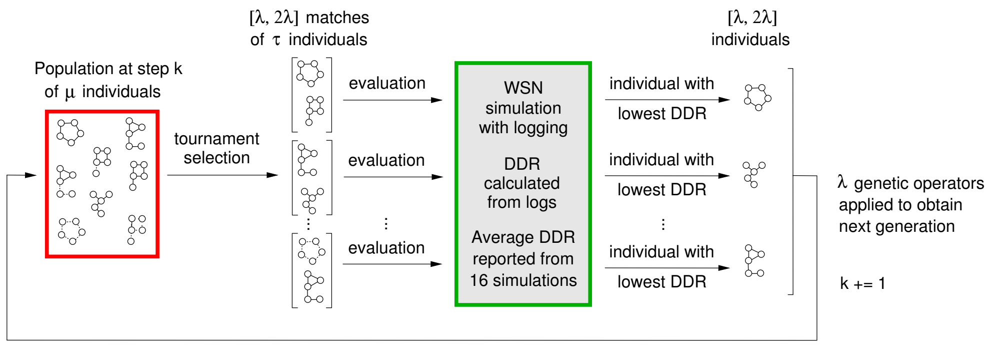


Figure 1: Overview of the evolutionary mechanism to generate network topologies as test cases for CTP

encoded as an individual in this methodology is described later in Section 3.3; in our text, we use the terms "individual" and "topology" interchangeably.

In order to then obtain an improved second generation, a genetic-inspired process of *mutation* is used: a number λ of *mutation operators* (described later in Section 3.4) will be applied to a selection of best individuals from the current population. The process of mutating an individual modifies one or more link gains in the corresponding network topology. Since a mutation operator is applied over either one or two individuals, the selection of individuals to be mutated must yield between λ and 2λ individuals, depending on the choice of mutation operators, which is itself probabilistic. The selection of individuals is done by *tournament*, which consists of *matches*. In each match, τ individuals "compete": CTP is run over each of the τ topologies in TOSSIM (as described in Section 2.3, with 16 repeated simulations), the simulations are logged, and DDR is calculated from the logs of each run; from each match, the topology with the lowest DDR is selected for mutation, the mutation process forms a new population, and the iteration step restarts.

The evolutionary algorithm from Fig. 1 is executed with the parameter settings shown in Table 2.

Table 2: EA parameters used.

Parameter	Description	Value
μ	population size	40
λ	number of genetic operators applied at every step	5
τ	size of tournament selection	2

The following *stopping conditions* can be decided for this iterative process; we use a combination of them:

- After a certain runtime dependent on the difficulty of the search, the algorithm may or may not *stagnate* for a number of generations, i.e., generate individuals which do not decrease DDR.
- Alternatively, a *runtime bound* can be placed on the runtime of an evolutionary test, either in terms of number of generations, or in terms of CPU runtime.

(Data sets). A large number of individuals will be evaluated in the sequence of populations generated by the algorithm in Fig. 1, between the first random population and the stopping condition. For example, in testing CTP over dense 20-node networks with asymmetric links and low radio noise, a stopping condition was placed at 700 generations; roughly 3000 individuals were generated in total among these generations, of which roughly 80% are *unique* topologies (i.e., topologies whose viable links form distinct graphs), and the remaining 20% are topologies which were generated identically more than once in the evolutionary process.

Furthermore, the evolutionary algorithm itself is stochastic, due to the initial population being generated randomly, and to the probabilistic choice of mutation operators at each generation. Due to this, repeated evolutionary tests of the same configuration may (and indeed, in practice do) yield different final DDR values. Thus, to gain confidence, we always repeat every evolutionary test 16 times with different random seeds. This has two advantages: (a) when one or more of the 16 repetitions did not succeed in generating interesting test cases, other repetitions did, and a lower DDR can still be located, and (b) we can provide the mean and standard deviation calculated based on this sample of 16 experiments, which helps evaluate statistically the ability of the evolutionary algorithm to converge on a certain, low DDR.

We describe the numeric low DDRs, and the sets of topologies we obtained for the various network configurations, in Section 4.

(Predictive topological metrics). On the basis of the test topologies obtained experimentally, whenever the experiments located example topologies which trigger unexpectedly low DDR, we do a manual analysis of the best topologies, and make an effort to locate the cause of low DDR in a general way, as summarized in Fig. 2. We detail the analysis of the test topologies in Sections 5 and 6.

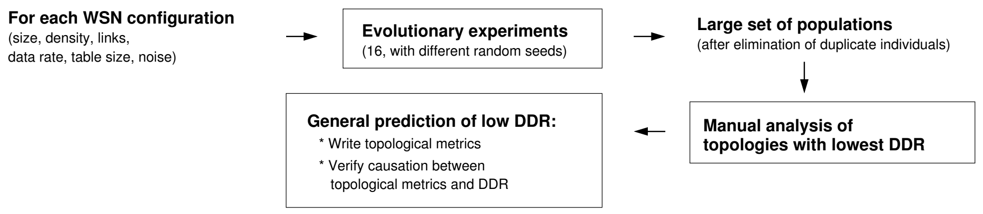

For each WSN configuration
(size, density, links,
data rate, table size, noise)
Evolutionary experiments
(16, with different random seeds)
Large set of populations
(after elimination of duplicate individuals)
Manual analysis of
topologies with lowest DDR
General prediction of low DDR:
* Write topological metrics
* Verify causation between
topological metrics and DDR

Figure 2: Overview of testing methodology for CTP

(Protocol fixes). Finally, we briefly discuss possible fixes to improve the data delivery for the protocol, also in Section 5.

3.2. Optimization of two objectives

We aim to find topological test cases which *minimize* the data delivery ratio of CTP. As such, the main objective of evolution is the DDR performance metric. However, some complications arise – we cannot allow the evolutionary algorithm to generate simply *any* topology, as *disconnected topologies* exist in the search space of possible topologies. These (1) naturally cause low DDR, as some nodes in disconnected networks will have no path to the sink, but (2) are not interesting for our analysis. To discover truly interesting topologies, we give a second objective to the evolutionary algorithm: the degree of *undirected network connectivity* should be *maximized* across the individuals generated. A graph is said to be undirectedly connected if it is connected via links which are equally good in both directions. We make this choice instead of choosing to enforce a less strict form of directed connectivity, to obtain topologies of quality also in the asymmetric case. The degree of undirected network connectivity is calculated as the fraction of nodes undirectedly connected to the sink.

Many real-life optimization problems have more than one aspect to be optimized at the same time, i.e., more than one objective to be minimized or maximized. The *lexicographical* solution consists of prioritizing the objectives according to some order of preference decided by a domain expert, so that these objectives are optimized following that predefined order. We use this approach with the connectivity objective preceding the DDR objective; this has the outcome that the evolutionary algorithm spends a number of the early generations (between 0 and roughly 100 in most of our cases) achieving maximum connectivity, after which all subsequent generations mutate mostly connected topologies and optimize the DDR objective.

3.3. Encoding topologies: genes and alleles

In our problem, an individual represents a candidate topology of the network. In Fig. 3, we show the encoding of topologies for different categories of networks.

A topology is encoded as a square matrix $N \times N$, with N the network size. Each position (i, j) in the matrix holds the gain of the link between nodes i and j, expressed in dB. The matrix is not symmetric in the general case. In Fig. 3, viable links are shown in green scale (gray scale in print, with darker cells being higher gains). Absent links are shown in white. The diagonal elements in the matrix (i.e., the self-connectivity of a node) are not part of the model.

As summarized in Section 2.3, we distinguish two cases in assigning gain values to viable links, because we are interested into investigating separately the effect of the presence and absence of radio noise on the overall protocol performance. In the absence of significant radio noise, we assign to links a fixed, idealized gain of 0 dBm, for

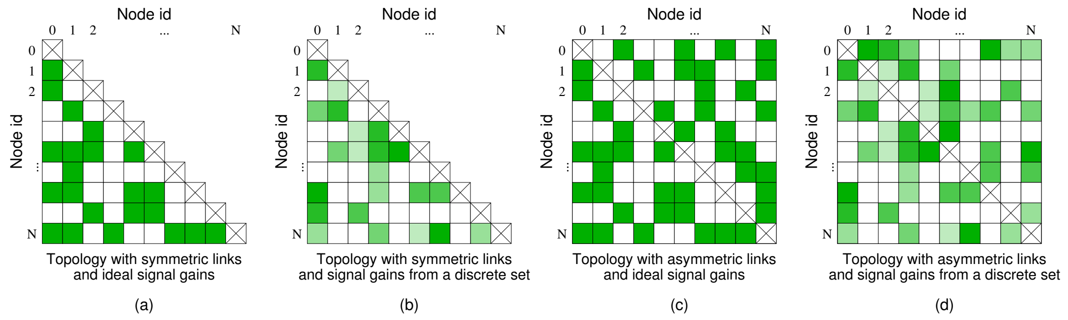

Node id
0 1 2 ... N
Node id
0 1 2 ... N
Topology with symmetric links and ideal signal gains
(a)
Topology with symmetric links and signal gains from a discrete set
(b)
Topology with asymmetric links and ideal signal gains
(c)
Topology with asymmetric links and signal gains from a discrete set
(d)

Figure 3: Encoding a topology into an evolutionary individual; (left) symmetric, and (right) asymmetric topologies. Shades of green (grey in print) model different link gains; white cells model the absence of links. For both symmetric and asymmetric topologies, we consider two cases: links with a single possible gain of 0 dB ((a) and (c)), and links with a uniform distribution of possible gains between -100 dB and 0 dB ((b) and (d)).

simplicity. In the presence of heavy noise, we discretize the set of possible gains as {-100, -80, -60, -40, -20, 0} dB: in this case, the performance of the link at one point in time will depend on both its own gain, and the strength of the noise generated by the noise model for that simulation time.

From an evolutionary perspective, this encoding translates into the following facts:

- A link is a *gene* in the *genome* of this individual.
- The configuration of a link gain represents the *allele* of a gene. For idealized links of a single gain value, a total of two different alleles exist for each gene in the individual's genome, i.e., viable and absent links, respectively. For links of differentiated gains, one allele can be further refined among the six possible gain values.

3.4. Genetic mutation and crossover operators

We use four standard mutation operators (described in Fig. 4), not designed specifically for mutating graphs, but rather having a general purpose suitable for many applications of natural-selection algorithms; we find that they are also effective in solving our problem. The operators are applied over a "linearized" individual, i.e., a matrix translated to an array in row-major order.

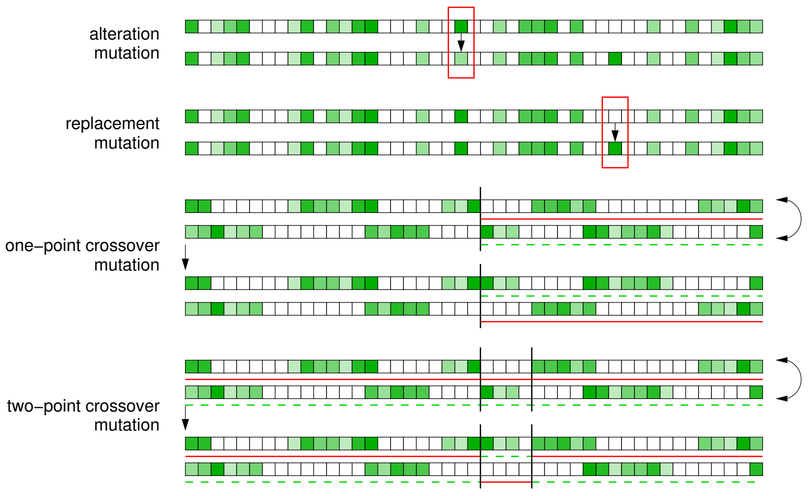


Figure 4: Four mutation operators acting over a linearized individual.

- A *replacement mutation* either creates a link in place of a previously absent link, or vice versa. When a link is created where a link also existed in a past generation, it is assigned the same gain.
- An *alteration mutation* changes the value of a link gain.
- *Crossover mutations* mix two individuals using either one or two cut points, creating two new individuals.

4. Searching for evidence of low DDR and interesting test topologies using evolutionary tests

We investigate the existence of topological problems of CTP via test campaigns, each aiming to assess any weaknesses of the protocol across network parameters. For this, the link-layer access protocol is always full power with acknowledgements for unicasts, and the values of network parameters are listed in Table 1. These test campaigns demonstrate how to obtain evidence of low DDR, and coincidently how to obtain large sets of test topologies of unusually low data-delivery performance; these data sets are publicly available[2]. Sections 4.1 to 4.4 look at the outcomes of the search for test topologies, without significant RF noise, when setting constraints upon link symmetry, network density and size, data rate, and table size, respectively; Section 4.5 compares the outcome of some of the previous tests with that of adding a pattern of heavy environmental RF noise.

4.1. (A) Symmetry versus asymmetry

To study the effect of link asymmetry upon DDR, we choose a single network size (20), network density (1/2), ideal gains (0 dB), a single data rate (one data packet issued by each node every 5 seconds), a size of 10 for *both* the neighbour table and the routing table, and light noise (for a high signal-to-noise ratio). Over these common settings, two types of topologies are generated in separate experiments:

- purely symmetric topologies, in which no asymmetric links exist
- asymmetric topologies, in which undirected connectivity (i.e., connectivity through symmetric links) is still met, but otherwise any degree of asymmetry can be added (as motivated in Sections 2.3, 3.2).

Table 3 summarizes these tests. As described in Section 3.1, each evolutionary test is repeated 16 times. The table lists the lowest DDR found among all repetitions; this DDR is the mean among a batch of 16 simulations of the same topology, so is presented as the mean and standard deviation.

Despite all topologies being connected, link asymmetry introduces problems that the CTP logic cannot deal with.

Table 3: Set of experiments (A) and lowest DDRs found (as sample mean and 95% confidence interval)

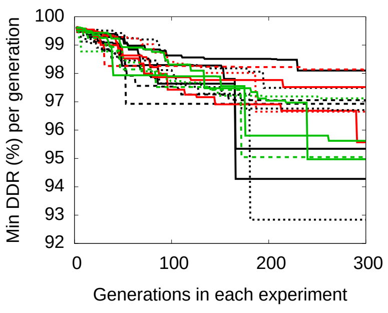
					lowest DDR found
links: symmetric	**nodes**: 20	**density**: 1/2	**table size**: 10	**noise**: low,	92.8 ± 9.9%
links: asymmetric	**nodes**: 20	**density**: 1/2	**table size**: 10	**data rate**: low	2.1 ± 1.0%

Figure 5: Set of experiments (A): symmetry. (left) The progression of generations, with each generation represented by the lowest DDR it found. (right) Mean and standard deviation among the final DDR values found by the 16 repetitions.

nodes: 20,
density: 1/2,
links: symmetric,
data rate: low,
noise: low
table size: 10

	DDR
min	92.8%
mean	96.2%
stddev	1.5%

(300 generations)



In Figs. 5 and 6, we show the progression of each test. For each battery of 16 repeated tests, we plot each repetition. For each of these repetitions, we show the minimum DDR found for connected topologies by every subsequent evolutionary generation. The evolutionary algorithm here succeeds in maximizing the connectivity of individuals (its first objective in lexicographical order) within few generations. Depending on the objectives of the test, running multiple repetitions (with different random seeds) of an evolutionary test may or may not yield similar results.

[2]Repository for test topologies: `https://github.com/doinab/evo-network-verification`.

nodes: 20,
density: 1/2,
links: asymmetric,
data rate: low,
noise: low
table size: 10

	DDR
min	2.1%
mean	4.8%
stddev	2.0%

(700 generations)

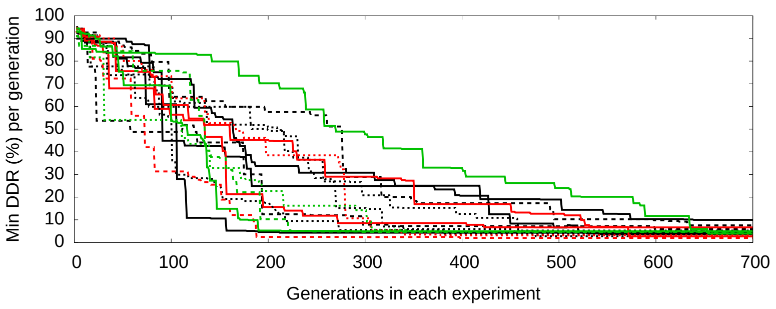


Figure 6: Set of experiments (A): asymmetry. (left) The progression of generations, with each generation represented by the lowest DDR it found. (right) Mean and standard deviation among the final DDR values found by the 16 repetitions.

In the first population, the DDR is effectively found via random generation of topologies (as per Section 3.1); it then improves through the evolutionary process, until eventually each run stagnates for a number of generations. We choose to set the same stopping condition for all repetitions of all the experiments in this section: after a runtime bound of a number *n* of generations chosen such that at least half of the 16 repetitions have stagnated on some DDR value for at least 100 generations.

Interestingly, the genetic mutation and crossover operators (described in Section 3.4), although they are general-purpose, do prove suitable for these search problems through topological space. However, in the future, improved operators could be specifically designed to manipulate individuals which encode graphs, and thus directly manipulate graph properties such as cycles, connectivity, and node degree.

A direct comparison of the evolutionary test dynamics shown in Figs. 5, 6 reveals that, while in the asymmetric case all runs converge to low DDR values, this is not true for the symmetric case, where instead the DDR values are bound to the range (92%, 100%), with most of the runs stagnated before the allotted budget of 300 generations. With asymmetry, the gradual decrease of the DDR trends (in all the runs, although with different decrease rate) indicates that there is a smooth transition of the DDR "landscape" towards a single, well defined "basin of attraction" corresponding to low DDR values in the range (0%, 10%). Although it is difficult to quantify the probability of finding misbehaving topologies in the search space, we can reasonably say that in the symmetric case such topologies are much more rare, while in asymmetric cases CTP can produce extremely low (almost-zero) values of DDR, with a broad range of topologies showing intermediate DDR values "converging" towards the worst topologies.

We derive an initial proposition from these experiments:

Proposition 4.1 *Enforcing link symmetry in connected topologies preserves good delivery. However, there exist connected topologies with link asymmetry which make CTP deliver nearly no data at the sink.*

The question arises whether the most interesting test topologies generated by the evolutionary framework could not also be generated by a simpler method such as a random generation of topologies. For this, we separately ran random tests, i.e., tests which created topologies of the same overall configuration, with Algorithm 1. The difference between the random generation of symmetric and asymmetric topologies is in the set of pairs of nodes (i, j), which is twice as large in the asymmetric case. We generate 3000 random topologies, which over-approximates the total number of unique individuals generated and evaluated by an evolutionary test, in both (symmetric, asymmetric) experimental campaigns.

Fig. 7 compares, for both the symmetric (left) and asymmetric (right) case, a random test and a representative evolutionary test from among the 16 repetitions in Figs. 5-6. In each of the four tests, a histogram is plotted showing the distribution of DDR values of the topologies found by the experiment; this illustrates the difference in the composition of the respective sets of test cases. While random sampling of the topological space yields topologies which give seemingly excellent protocol performance, the evolutionary test has far improved coverage of the "interesting" segments of the topological search space. This is particularly true for the asymmetric case, which yields numerous unique examples of topologies of DDR close to zero.

ALGORITHM 1: Random generation of topology

```
Input: Network size $N$
Input: Network density $D$ as percentage
Input: Set of viable link gains $G$
Output: Topology $t$
begin
    repeat
        $t$ = graph of $N$ nodes and no links
        forall the pairs of nodes $(i, j)$ do
            with probability $D$ generate a link $(i, j)$ with a gain $g \in G$
        end
    until $t$ is undirectedly connected;
end
```

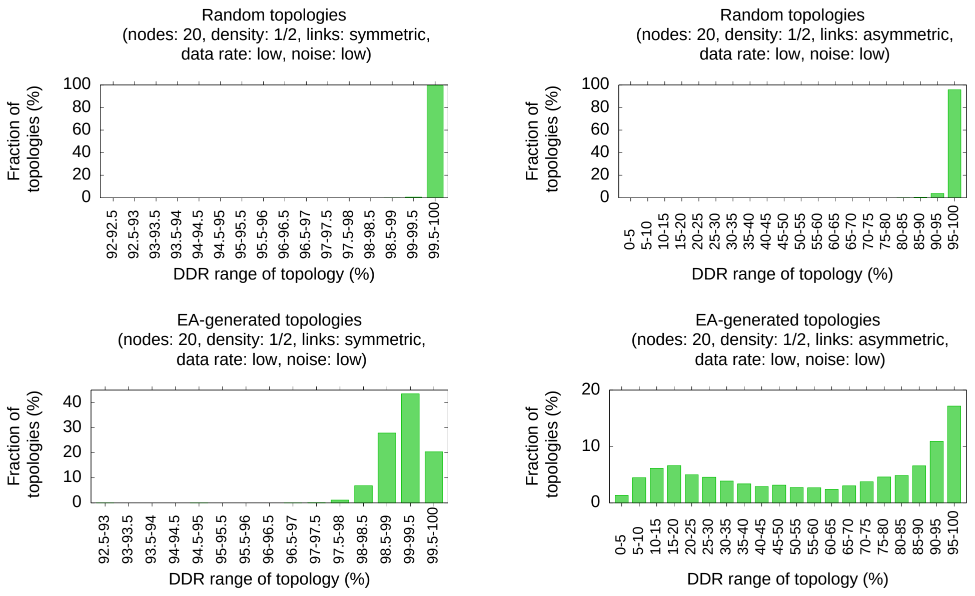


Figure 7: Set of experiments (A): comparison in the coverage of a random test versus that of an evolutionary test, for (left) symmetric topologies, and (right) asymmetric topologies.

4.2. *(B) Network size and density*

Having established in Section 4.1 that asymmetric topologies are particularly significant to explore further, we are interested in assessing whether network size and network density have an effect upon the data delivery. Table 4 summarizes the setting of the tests, and also gives the lowest DDR found for connected topologies among the repetitions of each experiment. We choose four network sizes, and a set of realistic densities.

Network density. We had expected that a comparatively lower network density, i.e., less path redundancy in the network, would find comparatively lower DDR. In fact, our tests show the contrary:

- For networks of size other than 10, the lowest data delivery was always found for the highest network density. (For small networks of size 10, the lowest data delivery is instead fairly consistent across densities.)
- Larger networks can have lower data delivery.
- Sparse networks (such as 30-node networks of 1/8 density, i.e., an average node degree of only 3.75) are less prone to near-zero data delivery. The lowest DDR found for this 30-node configuration is significantly higher,

Table 4: Set of experiments (B) and lowest DDRs found (as sample mean and 95% confidence interval)

					lowest DDR found
links: asymmetric	**nodes**: 10	**density**: 1/2	**table size**: 5	**noise**: low, **data rate**: low	9.7 ± 1.9%
		density: 1/4			8.3 ± 2.5%
		density: 3/4	**table size**: 10		9.7 ± 3.8%
	nodes: 20	**density**: 1/2			2.1 ± 1.0%
		density: 1/4			4.6 ± 0.8%
	nodes: 30	**density**: 1/2			0.1 ± 0.1%
		density: 1/4			0.3 ± 0.1%
		density: 1/8			29.0 ± 12.5%
	nodes: 50	**density**: 1/4	**table size**: 25		0.1 ± 0.0%
		density: 1/8			1.7 ± 0.7%

at 29%[3].

Furthermore, we found a significant effect of the network density on the performance of the testing algorithm. Comparing the 20-node, 1/2-density test in Fig. 6 with the 20-node, 1/4-density test in Fig. 8, we see that sparser topologies of low DDR in CTP are harder to search for than denser topologies of low DDR. I.e., the effectiveness of the evolutionary algorithm in searching for test topologies decreases with decreasing density, when fewer of the 16 test repetitions locate interesting test topologies. This signals the fact that at least one of the topological causes is more common in dense networks, and thus may be caused by a form of network overload. Later in Section 5, we verify this hypothesis.

nodes: 20,
density: 1/4,
links: asymmetric,
data rate: low,
noise: low
table size: 10

	DDR
min	4.6%
mean	18.0%
stddev	15.1%

(800 generations)

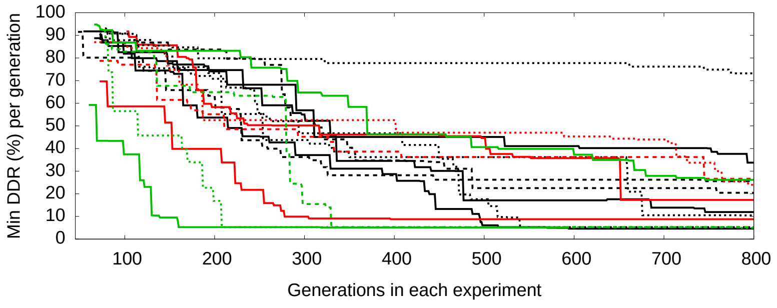


Figure 8: Set of experiments (B): lower density for 20-node networks. (right) The progression of generations, with each generation represented by the lowest DDR found. (left) Mean and standard deviation for the final DDR values found by the 16 repetitions.

We thus derive the preliminary proposition:

Proposition 4.2 *Over connected asymmetric networks of different medium-to-high densities, CTP has similar low DDR. In connected asymmetric networks of low density, data delivery is significantly more resilient; low-density, connected asymmetric topologies with low DDRs are statistically less probable than high-density topologies of the same DDR.*

Network size. The DDR values found for 10-node networks (Table 4) are such that the best test topologies have only *one node* delivering *some* data to the sink, and the DDR reaches just under the corresponding 11.1% value (i.e., 8 out of 9 non-sink nodes deliver all their data); in fact, among the best test topologies, we most often saw individual simulations with a DDR outcome of exactly 11.1%, but we also sometimes saw simulations with a resulted DDR of exactly 0%. This will be motivated later (in Section 5) in the process of analyzing the test topologies obtained. For

[3]The 30-node and 50-node, 1/8-density tests are exceptional cases: out of 16 long evolutionary tests each, 4 tests each did not succeed in maximizing connectivity (the first objective in lexicographical order) to exactly 100%, due to the relative scarcity of topologies which are, at the same time, sparse, connected, and showing a performance problem. The mean and standard deviation reported are thus calculated over 12 evolutionary tests instead of 16.

denser, larger networks, the DDR resulted is lower, such that no node had a good delivery ratio; the higher the network size, the closer to zero the DDR. We preliminarily give a general statement for DDR performance when running CTP over connected topologies: for all network sizes and middle-to-high densities, there exist topologies such that *all except one node* cannot deliver any data to the sink, and there also exist other topologies in which all nodes will deliver nearly no data.

Proposition 4.3 *Over the domain of connected asymmetric topologies of some network size N with one sink, the DDR for CTP can fall into the range* $\left[0, \frac{N-2}{N-1}\right]$.

4.3. (C) Data rate

We aim to determine the effect in data delivery after increasing the number of data packets issued in the network. Since CTP (a) detects logical inconsistencies in the routing tree using datapath validation, and also (b) estimates the ETX link cost using the acknowledgements due for unicast data packets, we have two opposing expectations:

- high data rate may overload crucial forwarding nodes, leading to low data delivery;
- high data rates should lead to better data-based link estimation in CTP, and thus better data delivery.

While all tests presented in Sections 4.1 and 4.2 used a data rate of 1 data packet per 5 seconds per node, we now add a heavy data rate of 1 packet per second per node. Table 5 summarizes the test settings and their best result.

Table 5: Set of experiments (C) and lowest DDRs found (as sample mean and 95% confidence interval)

					lowest DDR found
links: symmetric	**nodes**: 20	**density**: 1/2	**table size**: 10	**noise**: low,	92.4 ± 10.8%
links: asymmetric	**nodes**: 20	**density**: 1/2	**table size**: 10	**data rate**: high	1.1 ± 1.0%

These results show that, when the links are kept symmetric, raising the data rate makes no significant difference in the lowest DDR found. However, in tests with asymmetric links (Fig. 9), in comparison with the same configuration but lighter data traffic (Fig. 6), the DDR is found to reach similar lows. However, test topologies of very low DDR are comparatively more difficult to locate by the search algorithm: only 5 out of 16 tests have reached under the 5.3% value predicted by Proposition 4.3 using the same number (700) of generations. This is likely the effect of increased datapath validation in CTP, which removes some (but not all) possible topological problems.

We thus write the following proposition:

Proposition 4.4 *Over connected asymmetric topologies and different data rates, CTP has similar data delivery. However, low data rates are more statistically probable than high data rates to cause topological problems in CTP.*

nodes: 20,
density: 1/2,
links: asymmetric,
data rate: high,
noise: low
table size: 10

	DDR
min	1.1%
mean	28.3%
stddev	23.8%

(700 generations)

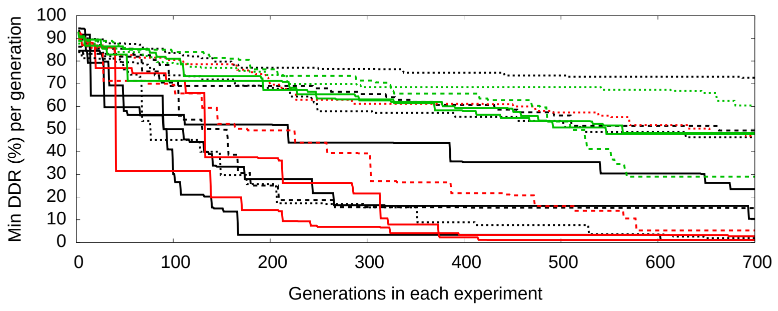


Figure 9: Set of experiments (C): high data rate over asymmetric links. (right) The progression of generations, with each generation represented by the lowest DDR found. (left) Mean and standard deviation for the final DDR values found by the 16 repetitions.

4.4. *(D) Size of tables*

We now aim to compare the delivery rates between two network configurations which are identical except for the size of the neighbour and routing tables used by CTP. Tables sizes have sometimes been found to be problematic in testbeds with ad-hoc routing [12]. For this test, we use the two network configurations from Table 6, and the table sizes 10 (i.e., half the size of the network) and 20 (i.e., equal to the network size, and the maximum necessary), respectively. Since we were aware of other related protocols having had, in deployments, problems related to the size and management of the routing table [12], we expected that, by maximizing the size of both neighbour and routing tables, data delivery will be maintained high.

Table 6: Set of experiments (D) and lowest DDRs found (as sample mean and 95% confidence interval)

					lowest DDR found
links: asymmetric	**nodes**: 20	**density**: 1/2	**table size**: 10	**noise**: low,	2.1 ± 1.0%
links: asymmetric	**nodes**: 20	**density**: 1/2	**table size**: 20	**data rate**: low	4.3 ± 3.6%

The reality of the tests was different: even with large tables, test topologies could be found so that DDR again falls under the bound of 5.3% hypothesized by Proposition 4.3. However, since the lowest DDR found with large tables has improved slightly, we foresee that different weaknesses of the protocol come into action between the different table sizes, and that the test topologies resulted may not follow the same pattern. We verify this hypothesis in Section 5. Furthermore, in comparison with the progression of generations in evolutionary tests with smaller tables, now with larger tables, the search algorithm does have a harder time locating interesting topologies. This means that, although larger table sizes do not rule out completely network problems, they make at least the problematic topologies more rare, and thus more difficult for the search algorithm to find.

4.5. *(E) Heavy RF noise for highly dynamic links*

To study the effect of link dynamicity upon data delivery, we choose a single network size (20), network density (1/2), data rate, assignment of link gains (differentiated, with any link taking any value from the set {-100, -80, -60, -40, -20, 0} dB), and noise model (heavy noise, with peaks of signal strength up to -40 dBm). We hypothesize that adding heavy RF noise to all links in the network will generally lead to overall poor data delivery. Table 7 summarizes the configuration of the tests.

Table 7: Set of experiments (E) and lowest DDRs found (as sample mean and 95% confidence interval)

					lowest DDR found
links: symmetric	**nodes**: 20	**density**: 1/2	**table size**: 10	**noise**: high, **data rate**: low	0.2 ± 0.1%
				noise: high, **data rate**: high	0.0 ± 0.0%
links: asymmetric				**noise**: high, **data rate**: low	0.0 ± 0.0%
				noise: high, **data rate**: high	0.0 ± 0.0%

The evolutionary dynamics, shown in Fig. 10, is in this case particularly interesting, and corroborates the intuitive hypothesis that high link dynamicity leads to poor network performance. In all the runs, the evolutionary algorithm first stagnates, for a variable number of generations, on one or more DDR "plateaus", i.e., topologies where the effect of noise upon DDR is not drastic. In only half of the runs, the search "escapes" these plateaus to find topologies of the true minimum DDR.

4.6. *Test parallelization, test runtimes, and test efficiency*

All the experiments have been performed on a 32-core Linux machine (Intel(R) Xeon(R) CPUs, 2.00GHz, 128GB RAM), running Ubuntu 12.04.1 LTS. To exploit the inherent parallelism of the objective evaluation (for which every simulation is repeated 16 times with different random seeds, as described in Section 2.3), we ran pairs of experiments in parallel, each one using 16 cores, so that each individual evaluation was fully parallelized. The degree of undirected connectivity and the value of DDR obtained from the 16 simulation repetitions were then averaged and passed to the evolutionary framework, as shown in Fig. 1.

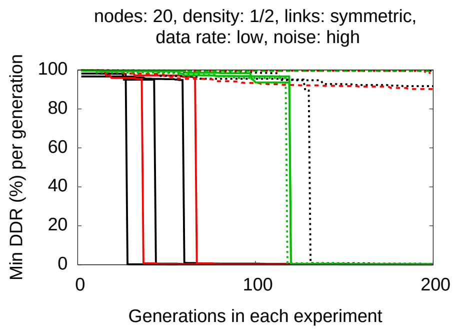
Table 8 summarizes the number of evolutionary generations, that of topologies evaluated throughout all the generations, and the runtime per test repetition. The runtime of each evolutionary test is mostly determined by the number

Figure 10: Set of experiments (E): heavy RF noise. (right) The progression of generations, with each generation represented by the lowest DDR found. (left) Mean and standard deviation for the final DDR values found by the 16 repetitions.

nodes: 20,
density: 1/2,
links: symmetric,
data rate: low,
noise: high,
table size: 10

	DDR
min	0.2%
mean	42.6%
stddev	24.6%

(200 generations)



of generations allowed in a test, the time needed for TOSSIM simulations, logging, and log processing, and, to a lesser degree, by the process of genetic mutation. The runtime is thus higher for larger network sizes, heavier noise models, and differentiated link gains. Less intuitively, topologies with low DDR require longer simulation times, due to the fact that the execution of CTP over test topologies of low DDR is more eventful (i.e., the network traffic is heavier).

Table 8: Summary of the experiments performed in this study. The number of topologies evaluated and the runtime of a single run (out of the multiple repetitions of each test) vary within a large range due to different load on our computing facility ([8 ÷ 24] available processor cores). Experiments marked with * were executed with stop condition on the number of stagnated generations (100). Experiments marked with ** were repeated 4 times (instead of 16 as in all other configurations) due to the long runtimes and after having observed small variation on the final DDR.

					generations	**topologies**	**runtime**
					(per evolutionary test)		
links: asymmetric	**nodes**: 10	**density**: 1/2	**table size**: 5	**noise**: low, **data rate**: low	400	1511-2306	3.0h-17.0h
		density: 1/4			500	2305-3122	9.3h-20.3h
		density: 3/4	**table size**: 10		125-795*	735-4468	1.0h-8.4h
	nodes: 20	**density**: 1/2			500-982*	2624-4783	7.4h-40.4h
		density: 1/4			800	2946-5212	14.4h-24.3h
	nodes: 30	**density**: 1/2			400	2112-3821	28.0h-50.2h
		density: 1/4			700	2472-5765	14.0h-54.0h
		density: 1/8			1500	5538-8698	17.4h-45.2h
	nodes: 50	**density**: 1/4	**table size**: 25		181-384**	1099-2284	22.3h-45h
		density: 1/8			538-1000**	2925-4800	25.2h-55h
links: symmetric	**nodes**: 20	**density**: 1/2	**table size**: 10	**noise**: low, **data rate**: low	300	1661-2176	6.3h-17.3h
				noise: low, **data rate**: high	300	1669-2031	8.3h-17.1h
links: asymmetric				**noise**: low, **data rate**: high	700	2731-4004	13.0h-35.4h
			table size: 20	**noise**: low, **data rate**: low	700	2917-4113	12.3h-35.1h
links: symmetric			**table size**: 10	**noise**: high, **data rate**: low	200	1009-2009	4.1h-23.1h
				noise: high, **data rate**: high	200	1163-1509	6.1h-13.1h
links: asymmetric				**noise**: high, **data rate**: low	50	293-369	2.2h-8.2h
				noise: high, **data rate**: high	50	279-374	3.0h-12.0h

It is interesting to determine which network configurations (network size, network density, link asymmetry, data rate, signal-to-noise ratio, and table sized) are comparatively more prone to problems. This can be estimated by looking at the ease (or difficulty) with which the repetitions of evolutionary tests over those configurations have found topologies of low DDR; this is also a measure of the efficiency of a test. In Figure 11, we show, for each configuration, the percentage of topologies whose DDR was evaluated (in evolutionary tests) to be *below* a threshold, chosen here to be 20%. (A note: these percentages are higher that the absolute likelihoods of problematic topologies occurring among all possible topologies of a configuration, since the evolutionary framework is biased towards finding problems. This summary is best used to compare different network configurations in terms of likelihood of topological problems.)

This summary strengthens our findings so far, i.e., that lower data delivery is more likely in larger networks, under higher densities, with link asymmetry, lower table sizes, and heavy RF noise. There are two extremes: while networks with purely symmetric links and low RF noise were never found to have low delivery rates (i.e., 0% of all topologies fall under the threshold in Figure 11), large, dense networks with link asymmetry or heavy RF noise may easily create problematic topologies.

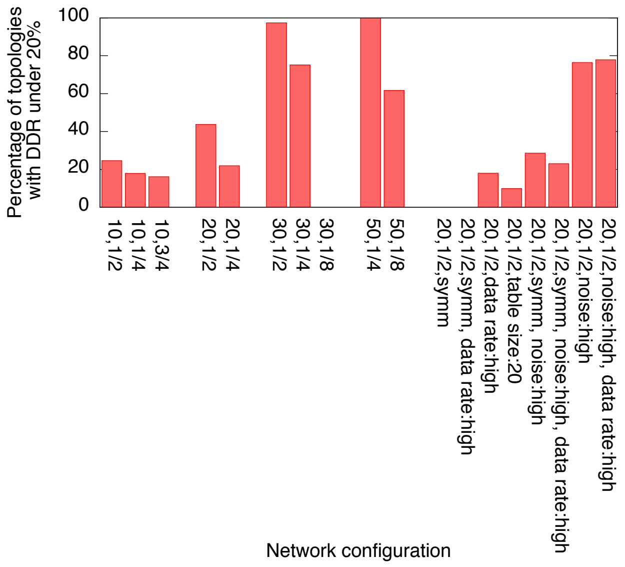


Figure 11: The percentage of all topologies with DDR *under* a threshold of 20%, out of all topologies evaluated in the evolutionary tests pertaining to one network configuration. On the x axis are the 18 network configurations tested, in the order in which they also were summarized in Table 8. The higher the bar for one network configuration, the more problematic topologies of that configuration exist.

5. Manual analysis of topologies

In this section, we zoom in on the test topologies obtained in Section 4, and make an effort to find those topological features which impede routing convergence. We focus on analyzing the execution of CTP over topologies with either link *asymmetry* or heavy RF *noise* and link dynamics, i.e., the defining topological features found through the experimentation in Section 4 to cause very low mean delivery among 16 simulations.

Since we find the behaviour of CTP over these topologies to be very highly probabilistic, for the analysis of any topology in this section, we raise the number of simulation repetitions to 100. The estimated cost of a link between two nodes i and j was denoted c_{ij} in Section 2.1 and is called *ETX* in CTP; the cost of a path to sink from node i was denoted C_i, and called *path ETX*. In this section, we use the practical terms ETX and path ETX.

We start with a summary of the section.

5.1. Summary of topological problems

We find two topological problems in asymmetric topologies:

(T_1) A pattern of *bottleneck nodes* with an insufficient *table size*;

(T_2) A pattern of *bottleneck nodes* overloaded with *beacon traffic* from neighbours in unresolved routing loops.

For topologies with heavy RF noise, we find, unsurprisingly, regardless of the (a)symmetry of the topology:

(T_3) A pattern of *bottleneck links* with intermittently low signal-to-noise ratio.

Both topological weaknesses, which confirm and give further evidence to recent results from testbed evaluations [17], have the outcome that the bottleneck nodes or links will “isolate” the rest of the network from the sink.

Overall, the causes of these weaknesses are mainly topological, e.g., a certain topological pattern is strictly necessary, and the protocol logic or configuration play a secondary role. Furthermore, we find that:

- There exists a correlation between low DDR and high network traffic, as for problem T_2.
- Problems T_1 and T_2 are often combined into the same topology.
- The execution of CTP over topologies with T_1 and T_2 is *highly probabilistic*: the order of certain network events is crucial for the outcome of a tree calculation, and those events are stochastic in nature.

Due to these being topologically caused weaknesses, short of heavily modifying the protocol, fixes for these conditions are in scarce supply: T_1 (only when *not* combined with T_2) is helped by allowing larger tables, and T_2 *may* be alleviated by a reboot of the network.

5.2. *Topological problem T_1: bottleneck nodes with overloaded routing table*

In Fig. 12 (left), we sketch the "top" asymmetric test topology of 10 nodes, density 1/2, ideal gains on links, and table size 5; this topology resulted from the experiments described in Section 4.2. We show all node identifiers (from the sink, 0, to node 9) as a means to debug the outcome of a simulation by referring to individual nodes in a given topology; in practice, nodes execute the same protocol logic, and identifiers are less relevant.

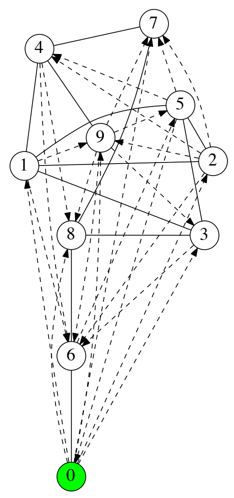
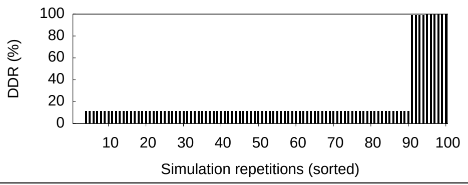
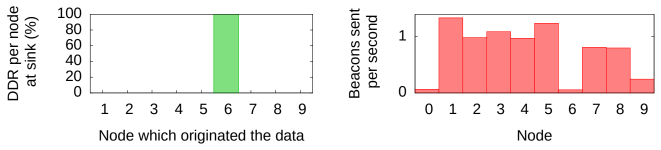


Figure 12: Asymmetric topology (size 10, density 1/2, table size 5) of low average DDR. Nodes 8 and 6 are **bottleneck nodes**. An undirected continuous line denotes a symmetric link; a directed dashed line is an asymmetric link. Two common simulation outcomes: DDR 11.1% and 0%.

100 simulations:
DDR:
mean 19.7 ± 5.3%,
min 0%,
max 100%

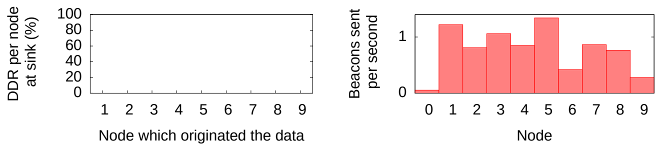


Example simulation 1: DDR 11.1%



5 parent choices for node 8 : 0 → 9 → 7 → 3 → 4 · · · (average 1 change per 10 s). All nodes except nodes 0 and 6 advertize a path ETX > 1000 after 200 s.

Example simulation 2: DDR 0%



5 parent choices for node 6 : 2 → 3 → 1 → 9 → 8 · · · (average 1 change per 5 s). All nodes except node 0 advertize a path ETX > 1000 after 200 s.

100 simulation repetitions of this topology have a mean DDR of 19.7%, and a large standard deviation, showing heavy stochasticity in the behaviour of CTP over this topology. To gain some understanding of the probability distribution of DDR over this topology, we plot the outcome of all simulations in Fig. 12 (top right); we find a non-normal step distribution with a common outcome of 11.1%, and less common 0% and 100%; this supports Proposition 4.3.

To understand the breakdown of delivery per source node in individual simulations, in Fig. 12 (right) we also show the most common delivery rate (at 11.1%) of a single simulation (in Example simulation 1), and a simulation which yields a DDR of 0%.

Example simulation 1. Node 8, two hops away from the sink, is a preferred forwarding node for all nodes (except node 6). Node 8 has 6 different neighbours which can advertize to 8 a route to the sink via incoming edges, and thus be parent nodes for 8; these are nodes 0, 3, 4, 6, 7, and 9. However, the best parent of node 8 is node 6. Based on simulation logs, we summarize the unfolding of the CTP logic which leads to node 6 never being selected as a parent for 8:

- *Beacon-based ETX estimates* "seed" the neighbour table of node 8. Thus, in the process of tree formation, node 8 receives beacons from 0 (parent:0, path ETX:0), from 9 (parent:0, path ETX:10), from 7 (parent:0, path ETX:10), from 3 (parent:0, path ETX:10), and from 4 (parent:9, path ETX:20), faster than from node 6. These neighbour nodes are added to the routing table until the table becomes full; the nodes are in yet-unresolved cycles, and have a very high beaconing rate.

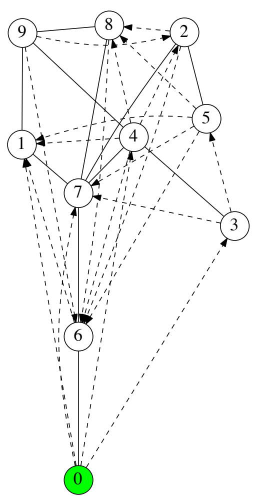
- *Data-based refining of link estimates* does not dominate beacon-based estimates. A link such as 0 → 8 is first inserted with a low link ETX of 10 in the routing table of node 8. Since the link is asymmetric, data sent from 8 to 0 is never received nor acknowledged. The link estimator slowly decays the link ETX after a time window of $w_u = 5$ data packets, but also increases it occasionally due to receiving some beacons on the link. Beacon-based ETX estimate will remain dominant, and the link 0 → 8 is never removed from the routing table of node 8.

- *Adaptive beaconing* has node 6 sending beacons very sparsely after a while (as seen in the breakdown of beacons sent by all nodes, in Fig. 12). Node 8 does initially receive beacons from node 6 (parent:0, path ETX:10), but node 6's beaconing rate decreases exponentially, due to the fact that node 6 is not in a topological position to detect a routing loop via datapath validation: its parent is always, correctly, node 0, and node 6 is not selected as a parent by its neighbours. Out of the beacons received by node 8 from node 6 (11 in total, during the 200s of evaluation), the first few are not used to populate the routing table, as the estimation of that link ETX needs a window of $w_b = 3$ to "mature" the estimation; due to the table management, the later beacons from node 6 do not evict any of the existing entries from the routing table.

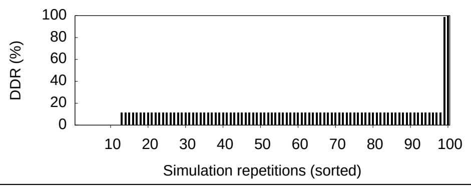
Overall, since node 8 never acquires a good parent, neither itself, nor any of the nodes it should forward data for, will deliver any data to the sink. The stochasticity of the CTP outcome over this topology is motivated by the stochasticity of the beacon reception: if the ETX to the "good" parent is estimated last, then routing convergence will never be achieved. Also, coincidently, as shown in the beacon histogram for this simulation in Fig. 12, since the beaconing rate among these nodes will not decrease, *excessive beacon traffic* (and thus, energy consumption) is a side effect.

Example simulation 2. A less probable, but possible, outcome of the same topology from Fig. 12 is a DDR of 0%, when not even node 6 delivers data, and node 6 itself is the bottleneck node with an in-degree of 7 and a table size too small. The cause is a process similar to that in the previous example: the link $6 \rightarrow 0$ never makes it into the routing table of node 6.

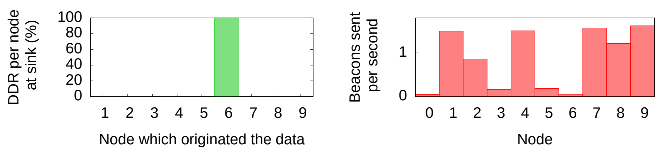
Other evolutionary tests provide test topologies of low DDR, of a similar pattern. In Fig. 13, we sketch the "top" asymmetric test topology of 10 nodes, density 1/4, ideal link gains, and table size 5; this topology also resulted from the experiments described in Section 4.2.



100 simulations:
DDR:
mean 11.5 ± 2.6%,
min 0%,
max 100%

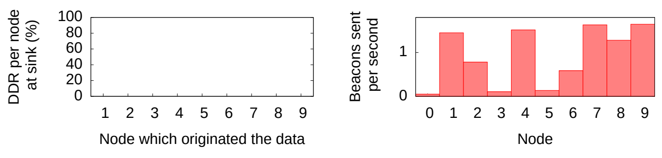


Example simulation 1: DDR 11.1%



5 parent choices for node 7 : $0 \rightarrow 3 \rightarrow 4 \rightarrow 1 \cdots \rightarrow 2 \cdots$ (average 1 change per 7s). All nodes except nodes 0 and 6 advertize a path ETX > 1000 after 200s.

Example simulation 2: DDR 0%



5 parent choices for node 6 : $4 \rightarrow 7 \rightarrow 5 \rightarrow 9 \rightarrow 2 \cdots$ (average 1 change per 4s). After 200s, nodes 3, 5 advertize a path ETX > 1000, and the remaining nodes (except the sink) advertise ETX > 2000.

Figure 13: Asymmetric topology (size 10, density 1/4, table size 5) of low average DDR. Nodes 7 and 6 are **bottleneck nodes**. An undirected continuous line denotes a symmetric link; a directed dashed line is an asymmetric link. Two common simulation outcomes: DDR 0% and 11.1%.

Despite the lower network density, the particular subgraph relevant to the problem has a similar pattern: two "bottleneck" nodes exist such that both their routing tables may be filled to capacity with incoming, but not outgoing, paths from the sink, while the table management does not "promote" into the routing table the best routing path. In all cases when routing failed, the bottleneck nodes saw a number of "bad" parents equal to the size of the routing table, compared to a single valid parent which never entered the routing table. Later in Section 6, we generalize this topological pattern as a means of predicting low DDR over any given topology.

In Fig. 13 (top right), repeated simulations of the topology shows a similar step distribution of the outcome as in Fig. 12, with identical DDR stepping points (as also shown in the two Example simulations in the figure), a similar likelihood for the most common DDR outcome of 11.1%, and somewhat different likelihoods for outcomes of 0% and 100%. We hypothesize that a DDR of 11.1% is far more likely than a more drastic one of 0% due to the fact that a bottleneck node which is connected directly to the sink node via a symmetric link is more likely to add quickly that direct link to sink into the routing table than a bottleneck node more than one hop away. In this example, increasing the size of the tables from 5 to 10 fixes the problem: the mean DDR is 99.7%.

We define the signature of this problem with Proposition 5.1:

Proposition 5.1 (Bottleneck node with overloaded routing table) *A node i satisfying the conditions below can be a* bottleneck node *for the network. In the worst case, neither node i, nor any other node which has i on its shortest symmetric path to the sink will deliver data.*

- *i is different from the sink;*
- *i has a table size smaller or equal than the number of distinct neighbour nodes j such that j has at least one incoming path from the sink, but no outgoing path to the sink.* □

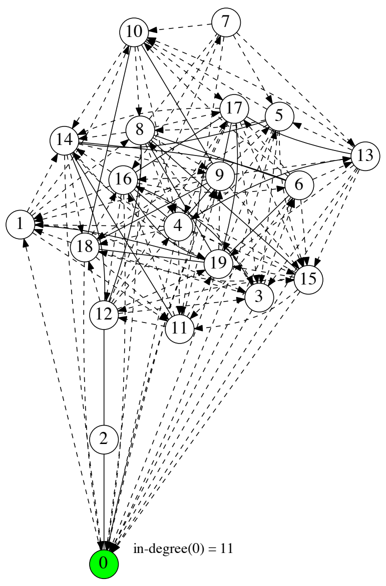
Fix for the problem. We give the following three statements with regard to effective fixes for this problem:

- This condition is fixed for small, 10-node networks, by allowing a larger size for *both* the neighbour and the routing table. With a table size of 10 (i.e., equal to the size of the network), the average DDR among simulations of the topology in Fig. 12 is 99.2%, and the sample standard deviation 0.3%, i.e., what can be called expected performance for a connected topology with ideal gains and insignificant levels of RF noise. Some parent churn occurs at boot, but the correct parent is found within 2 s after the the start of the data traffic.
- We verified that the condition is *not* fixed by a new scheme for routing table management which inserts neighbours into the table by evicting the worst existing entry.
- We verified that the condition is *not* self-repairing in time by running hour-long simulations of these test topologies. The distribution of the DDR outcome was found to be statistically similar to that for short simulations: for the topology in Fig. 12, while the mean DDR among 100 *short* simulations is 19.7%, the mean DDR among 100 *long* simulations is 14.3%, the outcomes are distributed among the same three discrete steps (0%, 11.1%, and 100%), and the average beacon rate remains at the same level of roughly 1 beacon per (non-sink) node per second throughout the duration of the simulation.
- Given the distribution of DDR among repeated simulations, with a fraction of the simulations converging of a working routing tree, network reboot *may* be a fix for the issue.

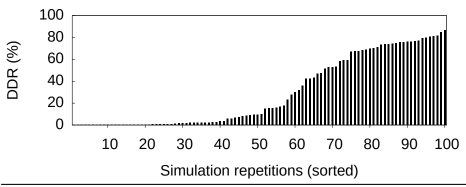
5.3. Topological problem T_2: bottleneck node overloaded with beacon traffic from neighbours in routing cycles

Tests with a larger routing table (i.e., where a fix for problem T_1 has been applied) yield a novel insight. In the example in Fig. 14, the size of the network is 20, and the size of both tables is 20 (equal to the network size; thus, the table size is sufficiently large for *any* physical topology). This topology resulted from the experiments described in Section 4.4. The figure gives the outcome of 100 simulations of the same topology; even though the standard deviation here is as large as in the previous examples of vulnerable topologies, it shows a different, smooth distribution of the DDR.

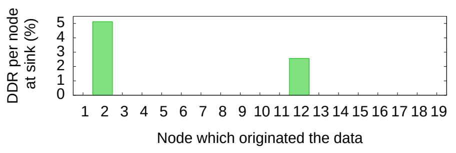
In roughly half of the simulations of this topology, global data delivery stays under 10%; a data delivery ratio over 90% is never achieved. The Example simulation in Fig. 14 shows a near-zero DDR, with two of the nodes delivering a fraction of data packets, and all nodes except the sink issuing heavy beacon traffic of around 1 beacon per node per second throughout the 200 s of simulation. While low DDR was also described in Sec. 5.2 as topological problem T_1, here, interestingly, the routing tables of nodes 2 and 12 *cannot* be overloaded. Instead, the cause is the radio-frequency overload of the sink node with beacon traffic from ten of its neighbours which maintain very high beaconing rates:

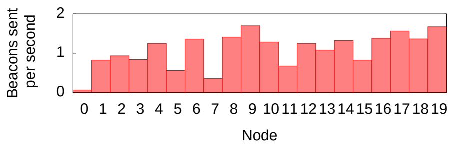


100 simulations: DDR: mean 29.2 ± 6.2%, min 0%, max 86.5%



Example simulation: DDR 0.4%





10 parent choices for node 2 : 0 → 11 → 9 → 12 → 1 → 19 → 10 → 3 → 6 → 5 ··· (average 1 change per 1.7 s).
6 parent choices for node 12 : 8 → 2 → 11 → 16 → 15 → 14 → 2 ··· (average 1 change per 2.7 s).
All nodes except node 0 advertize ETX values around 1500 after 200 s.

Figure 14: Asymmetric topology (size 20, density 1/2, table size 20) of low average DDR. Node 0 is a **bottleneck node**. An undirected continuous line denotes a symmetric link; a directed dashed line is an asymmetric link. The asymmetric links adjacent to node 2 omitted for clarity. An example simulation: DDR under 5%.

- For the first 11 s, node 2 calculates, selects and maintains the correct parent node 0, after which another neighbour is selected, and parent churn occurs until the end of the simulation. On the other hand, node 12 only occasionally and for brief intervals selects node 2 as parent.
- The reason for this parent choice is the following: the ETX of the link 2 → 0 degrades to 56 within the first 11 s, due to node 2 needing 27 total data retransmissions to receive 2 acknowledgements from node 0. This leads to the degradation of the link ETX,makes the link uncompetitive in comparison with the other links of node 2, so the protocol ignores this link in the calculation of the next best neighbour.
- At this point, a more competitive node (with ETX 10, path ETX 30) is picked as parent. The path ETXs of the neighbours of node 2 are now relatively comparable; all nodes have low link ETX, but high path ETX, which causes the parent churn at node 2. Node 0 remains in the tables, but is never chosen again as parent, since the ETX of the link 2 → 0 never improves.
- After 2 chooses an incorrect parent, it beacons the news and its increasingly high path ETX (up to 1500 after 200 s), so that node 12 doesn't see node 2 as a particularly competitive parent any longer. This means that the data packets from node 12 are also transmitted on wrong routes.

We define the signature of this problem with Proposition 5.2:

Proposition 5.2 (Bottleneck node overloaded with beacon traffic) *A node i satisfying the conditions below can be a* bottleneck node *for the network. In the worst case, nodes which have i on its shortest symmetric path to the sink will deliver no data.*

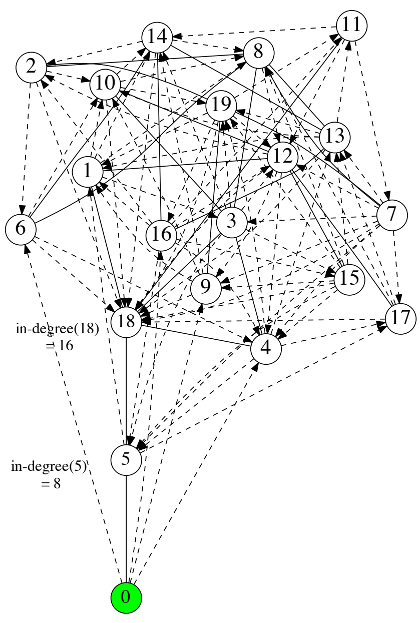
- *i has a large number of neighbour nodes j such that:*
  - *a link j → i exists, but the reverse link does not exist, and*
  - *j is part of some topological cycles which can be reached by beacons from the sink.* □

Fix for the problem. This situation of traffic overload on bottleneck nodes is primarily caused by the highest rate for adaptive beaconing in CTP, together with a topological load on the bottleneck node. Possible fixes include topology control to limit the link asymmetry, and a less aggressive beaconing.

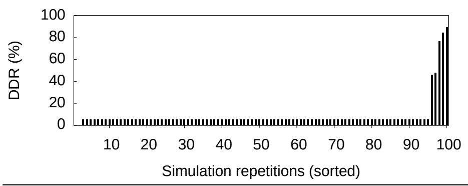
5.4. Combination of topological problems T_1 and T_2

Among the best test topologies, more frequent than isolated problems are composed problems. Test topologies of intermediate DDR (e.g., 50%) effectively are topologies in which only a portion of the nodes are affected by one or more topological problems. Such topologies may frequently occur in practical testbeds; they have been found in [17].

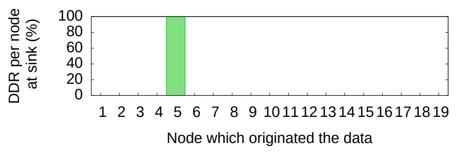
As an example, Fig. 15 shows the best test topology of 20 nodes, density 1/4, and table size 10. The most common simulation output for this topology is a 5.3% DDR (**Example simulation 1** in Fig. 15 (right)), i.e., the situation caused by problem T_1, in which node 18 is a bottleneck, node 5 delivers all data to the sink, but no other nodes deliver any packets.



100 simulations: DDR: mean 8.3 ± 2.8%, min 0.3%, max 89.5%

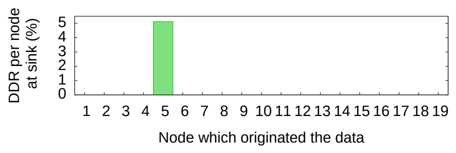


Example simulation 1: DDR 5.3%



15 parent choices for node 18 : 4 → 9 → 14 → 11 → 17 → 2 → 15 → 1 → 10 ··· (average 1 change per 1.6 s).

Example simulation 2: DDR 0.3%



7 parent choices for node 5 : 0 → 15 → 16 ··· (average 1 change per 3 s).

Figure 15: Asymmetric topology (size 20, density 1/4, table size 10) of low average DDR. Nodes 18 and 0 are **bottleneck nodes**. An undirected continuous line denotes a symmetric link; a directed dashed line is an asymmetric link. Two common simulation outcomes: DDRs 5.3% and nearly zero.

A second bottleneck of this topology may be the sink node 0 itself, even if with a lower likelihood than node 18: in a small percentage of the simulations (such as **Example simulation 1** in Fig. 15 (right)), in addition to the problem created at node 18, node 5 itself delivered few or no data packets to the sink. Node 5 does choose the sink as its first parent, and delivers its first data packets. However, by simulation second 16, due to the same phenomenon of the degradation of ETX estimation due to data retransmissions, the ETX of the link 5 → 0 is degraded to a value of 125, and node 0 is replaced as parent, followed by heavy parent churn. Node 0 is never again chosen as parent due to its link ETX remaining above the threshold of 50.

The fact that node 0 is also a bottleneck, makes us hypothesize that increasing the size of CTP tables from 10 to 20 will not be a partial fix. This is indeed the case: in Fig. 16, 100 simulations with large table size improve the average outcome of the topology.

5.5. Topological problems T_1 and T_2 are preserved under high data rate

Testing the protocol with high data dates confirmed the same topological issues T_1 and T_2. We omit giving an example here; our public repository contains the test sets.

100 simulations with table size 20: DDR: mean 70.2 ± 5.9%, min 5.3%, max 95.4%

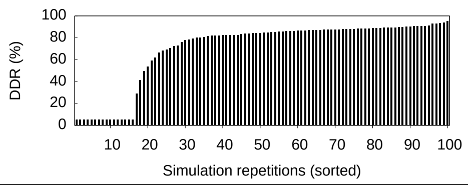
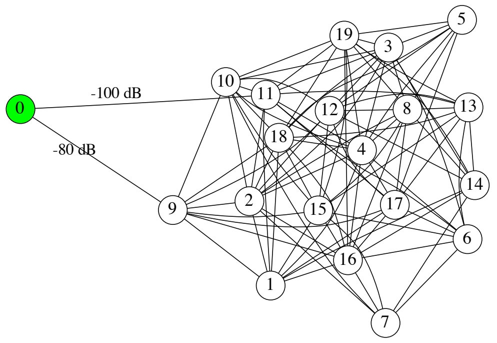


Figure 16: The 20-node, 1/4-density, asymmetric topology of low average DDR from Fig. 15 shows good improvement when increasing the size of the CTP neighbour and routing tables from 10 to 20.

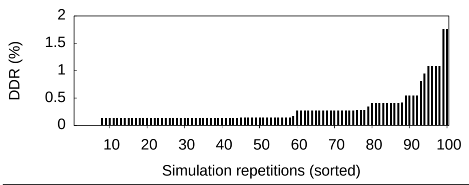
5.6. *Topological problem T_3: bottleneck links of low gain*

In Section 4.5, all tests with a heavy noise model yielded similar, near-zero DDR. Under link asymmetry in particular, this result is expected: even under ideal conditions of light noise and high link gains, we could previously find topological problems T_1 and T_2. However, with *symmetric* links and light noise, the tests in Section 4.1 did not previously find test topologies to show a DDR under 92%. Thus, here we show an example of a symmetric test topology of near-zero DDR under heavy noise; intuitively, these will show a pattern of bottleneck links of low signal-to-noise ratio.

In Fig. 17, we show the "top" test topology of 20 nodes, density 1/2, and symmetric links. Unlike for the other topological problems, the statistical outcome of this example topology has a low standard deviation, i.e., the test runs are very consistent.



100 simulations: DDR: mean 0.29 ± 0.14%, min 0.00%, max 1.76%



Example simulation: DDR 0.27%

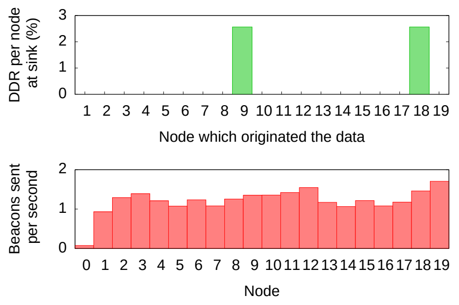


Figure 17: 20-node, 1/2-density, symmetric topology of low average DDR under heavy noise, and a simulation outcome with a DDR close to zero.

8 parent choices for node 9 : $0 \rightarrow 16 \rightarrow 18 \rightarrow 17 \rightarrow 12 \rightarrow 10 \cdots$ (average 1 change per 4.54s). All nodes except node 0 end up advertizing ETX values around 800 after 200 s.

The link from node 9 to the sink (of gain -80 dB) is initially added into node 9's routing table and the sink is selected as parent. However, the low signal-to-noise ratio of that link effectively makes the link a failure: neither beacons, nor acknowledgements to data packets are received consistently by node 9, e.g., only two acknowledgements are received during the 200 s of the example simulation. Consequently, the link ETX estimation is degraded, and a point is reached when the estimation falls below the threshold of 50, and another parent is selected. The link from node 11 to the sink (of gain -100 dB) is overpowered by noise and never viable.

Other test topologies with RF noise obtained by the evolutionary experiments have a similar structure: a different number of links to and from the sink node exist, yet all have low gains, and thus low signal-to-noise ratios. No effective fix exists for this situation other than topology control, as the problem is purely topologically caused, and not a failure of the routing protocol.

We define the signature of topological problem T_3 with Proposition 5.3:

Proposition 5.3 (Bottleneck link of intermittently low signal-to-noise ratio) *A link $i \rightarrow j$ satisfying the conditions below can be a* bottleneck link *for the network. In the worst case, neither node j, nor any other node which has j on its shortest path to the sink will deliver data.*

- *$i \rightarrow j$ is part of the shortest path to the sink of many nodes in the network;*
- *the strength of the RF noise in the network is probabilistically higher than the signal strength on the $i \rightarrow j$ link.* $\square$

6. Using topological metrics to predict protocol performance

In this section, we attempt to learn a general lesson from the test topologies generated for topological problems T_1 and T_2, and answer the (Predictive topological metrics) question from Section 1: Can we quantify those particular topological features which can provably be shown to *cause* low performance in the protocol?

In what follows, Section 6.1 gives basic notation and definitions. Sections 6.2 and 6.3 describe relevant topological metrics, and means by which causality between topological metric and protocol performance was verified.

6.1. Basic definitions and notation

Directed and undirected graphs. A WSN topology is a *directed* graph $G = (V, E)$ without self-loops, where the node denoted *sink* $\in V$ represents the root of the collection tree. We denote by $U(G) = (V, U(E))$ that *undirected* graph having the same node set as G, but where only a subset of edges from E are preserved in $U(E)$: $\forall u, v \in E, u \leftrightarrow v \in U(E)$ iff $u \rightarrow v \in E$ and $v \rightarrow u \in E$.

Directed and undirected paths. A *path* in a graph is an *acyclic* sequence of nodes. If a *directed path* $\langle u \rightarrow \ldots \rightarrow v\rangle$ exists in G from node u to v, then v is said to be *reachable* from u. A node v reachable from the sink node is said to be *sink-reachable*. For a node v, a directed path $\langle sink \rightarrow \ldots \rightarrow v\rangle$ is called an *in-path* to v, and a directed path $\langle v \rightarrow \ldots \rightarrow sink\rangle$ is called an *out-path* from v. A path between the nodes u and v, such that the path also exists in $U(G)$, is called an *undirected path*.

Directed and undirected connectivity. A *connected component* is a subgraph in which all nodes are mutually reachable (i.e., via directed paths). A graph is said to be *connected* if it has exactly one connected component. A graph is said to be *undirectedly connected* if it is also connected via undirected paths. Note that when the concept of connectivity is used, directed connectivity is implied; undirected connectivity is always stated explicitly.

Undirected shortest path. In a connected graph there exists a tree of shortest paths to the sink. If the graph is undirectedly connected, there exists a tree of undirected shortest paths to the sink.

We denote the table size of a node v by TableSize(v).

6.2. Topological metrics which predict low DDR

Definition 6.1 ***(Instream and outstream node)*** *For a node v, a neighbour node u of v is called an* instream *node for v if there exists an in-path into v of the form $\langle sink \rightarrow \ldots \rightarrow u \rightarrow v\rangle$, and an* outstream *node if there exists an out-path of the form $\langle v \rightarrow u \rightarrow \ldots \rightarrow sink\rangle$. For a node v, the set of instream nodes is denoted $V_{in(v)}$, and the set of outstream nodes is denoted $V_{out(v)}$.*

Intuitively, for a node v, an instream node u is a node which will be a candidate parent for v, for a protocol which does at least a partial estimation of link costs using directed beacon packets. Unless the instream node u is also an outstream node, the choice of u for parent will be unfortunate for v.

Definition 6.2 ***(Topological in-overload)*** *A node v for which the metric $O(v) := \left|V_{in(v)} \setminus V_{out(v)}\right|$ is large is said to have* topological in-overload.

We experimentally found this type of topological overload to be dangerous when $O(v) \geq \text{TableSize}(v)$ (for T_1), or when many instream nodes send heavy CTP routing traffic towards v, overloading the node (for T_2). Such heavy traffic is caused in CTP primarily when the instream nodes are situated in unresolved topological loops. We define supporting definitions for this situation below.

Definition 6.3 ***(S-cycle)*** *Given a connected directed graph G, a cycle $\langle v_1 \to v_2 \to \ldots \to v_k\rangle$ is called a* sinkless cycle *(S-cycle) if $\forall i = 1..k, v_i \neq sink$ (i.e., the sink node is not part of the cycle).*

Definition 6.4 ***(Traffic in-overload)*** *A node v with many of the nodes which are counted in $O(v)$ also situated in a number of S-cycles is said to have* traffic in-overload*. Traffic in-overload implies topological in-overload.*

The last piece of the puzzle is in defining bottleneck nodes. In a graph G which is undirectedly connected (i.e., the topology has a good-quality skeleton of symmetric links), topological in-overload brings a non-zero probability for problem T_1 to be caused by node v (depending on $\text{TableSize}(v)$), while traffic in-overload raises the probability for problem T_2 to be caused by node v. Then, if such an overloaded node v lies on the shortest paths of other nodes, then these other nodes likely deliver little to no data to the sink.

Definition 6.5 ***(Bottleneck nodes and obstructed nodes)*** *In an undirectedly connected graph G, a node v such that:*

- *either the topological in-overload of v exceeds the table size of v, $O(v) \geq TableSize(v)$*
- or *v has large traffic in-overload*
- *and v is situated on the shortest undirected paths to the sink of a set of other nodes in G*

is called a bottleneck node*. Any node which has v on its shortest undirected path to the sink (including v itself) is called an* obstructed node*; the set of obstructed nodes in G is denoted $V_O \subseteq V$.*

A "problem topology" is one which has a large fraction of obstructed nodes. While a problem topology will have a heavily probabilistic DDR outcome, the fraction of obstructed nodes can predict low DDR outcomes for any given topology.

6.3. Verification of topological metrics for DDR

We aim to verify quantitatively that a large fraction of obstructed nodes in the topology will cause CTP to have low DDR for that topology; the experimental observation of the fact that this topological metric qualitatively correlates with a low DDR is not sufficient to establish causation.

For this, we write a simple, random topology-generating algorithm (Alg. 2), designed such that it maximizes the number of obstructed nodes in any topology t generated. To do this is a simple way, it always creates, as seen in all the top topologies given as examples in Section 5, a lasso-like topology, i.e., a skeleton of undirected links such that a small number of these links (in our examples from Section 5, two adjacent links), starting in the sink node, are common to the undirected paths to the sink of *all* other nodes. Creating bottleneck nodes among the nodes of these few adjacent links effectively obstructs all the nodes in the network.

In Alg. 2, some features of the topologies generated are configurable in the input. Notably, this includes the number of adjacent bottleneck nodes B, and the topological in-overload O. At step 1 and step 2, a skeleton tree of undirected nodes is formed to connect all nodes; the tree has a linear "trunk" of B nodes and $B-1$ links. Traffic in-overload is ensured by creating (at step 3 and step 4) sinkless, sink-reachable cycles around all the nodes which are not bottleneck nodes. At step 5, O of these nodes in S-cycles are turned into instream nodes for the bottleneck nodes. Finally, to control the density of the topology, step 6 adds purely asymmetric links to the topology among non-bottleneck nodes.

By running this algorithm, test topologies are generated in an efficient manner; the evaluation of these topologies does show the expected low DDR and large standard deviation among repetitions of the simulation of a single topology. Fig. 18 shows 100 topologies randomly generated by Alg. 2 with settings $N = 20, D = 1/2, B = 2, O = 10, C = 4$; the topologies are evaluated by running 16 simulations of CTP with a table size of 20, to showcase the more drastic topological problem T_2.

ALGORITHM 2: *Reverse generation of test topologies* based on knowledge of topological metrics which cause low DDR in CTP. All links are assigned an ideal gain of 0 dB.

```
Input: Network size $N$, with the node set $\{0, \ldots, N-1\}$; Network density $D$; Number of bottleneck nodes $B$; Topological
       in-overload for bottleneck nodes $O$; Cycle upper bound $C$
Output: Topology $t$
begin
        $t$ = graph of node set $\{0, \ldots, N-1\}$ and no links
step 1  add a linear path of symmetric links $0 \leftrightarrow 1 \leftrightarrow \cdots \leftrightarrow B$
step 2  foreach node $n$ in $\{B, \ldots N-1\}$ do
            choose randomly a node $m$ from $\{B-1, \ldots n-1\}$
            add symmetric link $n \leftrightarrow m$
        end
step 3  repeat
            add a set $S$ of cycles of random size between 3 and $C$, of asymmetric links, among nodes from $\{B-1, \ldots N-1\}$
        until all nodes in $\{B-1, \ldots N-1\}$ are included in cycles;
step 4  foreach cycle in $S$ do
            choose a node $n$ on the cycle and add an asymmetric link $0 \rightarrow n$
        end
step 5  foreach node $b$ in some subset of $\{0, \ldots, B-1\}$ do
            repeat
                choose randomly a node $n$ from $\{B, \ldots N-1\}$
                if the link $b \rightarrow n$ does not exist in $t$ then
                    add an asymmetric link $n \rightarrow b$
                end
            until $O$ links have been added;
        end
step 6  repeat
            choose randomly two different nodes $n_1, n_2$ from $\{B-1, \ldots N-1\}$
            if the link $n_2 \rightarrow n_1$ does not exist in $t$ then
                add an asymmetric link $n_1 \rightarrow n_2$
            end
        until density $D$ is met;
end
```

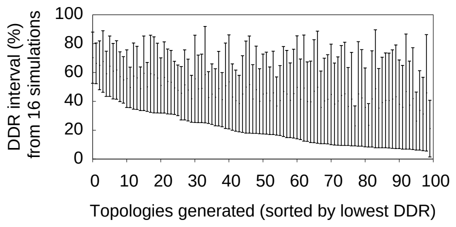
7. Related work

Testbed evaluation. Realistic experimentation with protocols is crucial. However, it is difficult to generalize from protocol evaluation on well-connected testbeds built out of homogeneous node platforms. In [10], the selection of 14 testbeds was such that "the hope is that different testbed environments we examine sufficiently capture a reasonable degree of variation in node density, radio technology, wireless environment, and deployment topology." The data rate varied across testbeds from one packet sent by each node every 8 seconds to one every 5 minutes, with experiments of several hours. The highest average parent churn rate found was 31.37/node·hour, with all except three of the testbeds showing an average parent churn under 5/node·hour. Two phenomena were found to be the dominant causes of data loss: link dynamics and transient loops.

High DDRs are reported in [10]: the lowest DDR (90.5%) was found on a testbed which did not use a full-power MAC protocol, was only intermittently connected, and of a rather low density (a minimum in-degree of 9 for a network size of 131). Quick convergence is also generally reported: in one experiment, after node failures, some nodes dropped up to 2 packets each immediately after the disruption, and most nodes found new routes in under 1 second. In another experiment, although the beacon rate in the network had decreased to one beacon every 8 minutes, CTP took only 325 ms to select a new parent for a node whose previous parent was removed. However, occasional difficulties with routing convergence and parent churn also occur in a subset of the nodes in the network.

In general, failure in uncontrolled deployments is rarely visible in fine detail to protocol developers: "We frequently failed to understand [..] performance results and could not determine who was to blame (i.e., the testbed characteristics, or the routing layer?)" [11]. The performance of certain deployments of precursors of CTP has been

Figure 18: 100 topologies generated by Alg. 2 with settings $N = 20, D = 1/2, B = 2, O = 10, C = 4$. The topologies are evaluated by running 16 simulations of CTP with table size 20, and the minimum and maximum DDR among the 16 simulations are shown.



found to be puzzling and counter to intuition, and different deployments were found not to paint a unified picture of the protocol performance: "comparing [evaluation] results from different testbeds is much like comparing apples and oranges" [11]. A problem was found in a deployment related to the management of the routing table in unexpectedly dense networks, similar to our topological problem T_1: "On the field about 70 nodes form a single cell around the gateway, which forces MintRoute to make a selection since it has room for only 16 nodes in its neighbor list. When adding neighbors, the replacement policy is as follows: MintRoute picks the neighbor with the lowest statistics value and evicts it from the table. As a consequence, the gateway is not part of the neighbor list for the majority of nodes" [12].

The existence of bottleneck nodes. Recent long-term experimentation on the large-scale GreenOrbs testbed found that, at all times, some fragments of the network routed no packets to the sink, and these fragments were as large as 23.5% of the network. Of course, since diagnosis packets from those areas suffered the same fate as the regular data packets, the type of fault remained unclear at the time; the network functionality was recovered after rebooting a number of crucial bridging node [18]. Subsequently, also over GreenOrbs, knowledge of critical bottleneck nodes and links was obtained: "Those critical nodes receive excessive incoming traffic, with fluctuating link loss rates and account for a large portion of packet drops. Current data collection protocols, like CTP with ETX as the routing metric, however, do not seem to successfully handle those cases" [17]. In the large-scale CitySee testbed, periodical packet loss was seen in 20% of the network; functionality was again recovered by rebooting certain critical nodes [18].

The BOND bottleneck node detector was developed in [18]. The tool depends on being able to collect periodical diagnostic data from the nodes in the network. It then learns in real-time, using a Hidden Markov Model, the strength of the dependence each node has on its (possible) parent nodes; it is then possible to predict, for example, that the effect of removing a parent node with strongly dependent children (i.e., a bottleneck node) will be the partial disconnection of the network. With such knowledge, when an error situation does occur in the network, the network manager knows where to manually add or remove nodes to recover functionality. An automated, rather than manual, option for maintaining the functionality of a network is to update the protocol itself with automated measures to reduce that traffic congestion which would create bottlenecks; e.g., [23] updates CTP with congestion-reduction intelligence.

Other diagnoses of topological fault modes. [20] also finds a protocol-independent (topological) metric that correlates with the ratio of data delivery to the sink in CTP. The metric is the sum of packet reception rates (obtained experimentally) over all shortest paths from every node to the sink; for this purpose, the shortest paths are calculated post-experiment using Dijkstra's algorithm. The analysis is limited to a small sample of large-scale, well-connected, but also asymmetric physical topologies in realistic testbeds, and also 50 network topologies in simulation. The work does not focus on worst-case scenarios, but does gather experimental evidence about the correlation between performance metrics and topological metrics.

[1] used an evolutionary algorithm to generate a critical test case for a simplified TCP/IP network. [3] proposed a semi-automatic method to generate worst-case data-throughput scenarios for the IEEE 802.11 MAC protocol. Their approach is based on a search algorithm through the state-space of an abstract model in state-machine syntax; this has the disadvantage that the modeling makes simplifying assumptions for some system features, instead of analyzing a protocol implementation. [2] does a search of the state space of a particular WSN application for fire detection, using a fitness function designed such that it guides the search algorithm towards WSN situations in which not only the system fails to detect a fire, but does so because of a minimal number of node failures caused by the environment. [6] uses a multiobjective evolutionary algorithm to obtain Pareto fronts (i.e., two-dimensional tradeoffs) for two CTP performance factors (delivery and energy cost); this result effectively shows the extent to which one performance factor can be improved without worsening the other. This study is small scale, but gathers some evidence for a negative correlation between DDR and energy cost. [27] models mobile mesh networks using queuing theory, and captures

the dynamics of network performance (including network throughput) via repeated simulation, using a predetermined model or measured trace of topology change; it does not focus on finding worst-case scenarios. Formal verification (sound but not necessarily complete) may be applied to obtain topological evidence of protocol faults [25]; since it is more computationally intensive, this requires abstract protocol remodelling, and obtained smaller topological patterns of 5 nodes as evidence of protocol faults.

8. Conclusions

We give topological insights on the delivery performance of a Collection Tree Protocol implementation. These are obtained with a testing methodology based on the concept of generational natural selection combined in a tool chain with a network simulator. We obtain example topologies over which the protocol has very low delivery rates close to 0%. Using these example topologies and detailed logs of the execution of the protocol, we convert this knowledge into topological metrics, which essentially measure two types of overload on a bottleneck node.

This testing methodology is general, protocol-independent, and has better coverage of crucial test cases compared with simulation-based testing using random test topologies: given an existing protocol implementation and a suitable simulator for this implementation, the method helps locate interesting topologies from the point of view of protocol performance. The topological problems discovered here are intuitive, and instances of similar topological issues were previously found in a large-scale testbed evaluation of CTP [17]. The advantage of our method is that it is able to find such concrete evidence of poor protocol performance, purely via computational means, fully automatically, and at protocol design time. Here, we complement existing results on the topic by showing that bottlenecks can also occur in small networks, and further show that this protocol performance outcome is probabilistic, and that link asymmetry is also a crucial factor in creating bottlenecks.

Future work will include testing the performance of other network protocols for low-power, self-organizing wireless networks, and designing and evaluating implementations for fixes of the performance problems found.

[1] Baldi, M., Corno, F., Rebaudengo, M., and Squillero, G. (1997). GA-based performance analysis of network protocols. In *Proc. Ninth IEEE International Conference on Tools with Artificial Intelligence, 1997*, pages 118–124.

[2] Bate, I. and Fairbairn, M. (2013). Searching for the minimum failures that can cause a hazard in a wireless sensor network. In *Proceedings of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, GECCO '13, pages 1213–1220, New York, NY, USA. ACM.

[3] Begum, S., Helmy, A., and Gupta, S. (2009). Modeling and test generation for worst-case performance evaluation of MAC protocols for wireless ad hoc networks. In *IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 1–10.

[4] Bertsekas, D. and Gallager, R. (1992). *Data Networks (Second Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[5] Bucur, D., Iacca, G., Squillero, G., and Tonda, A. (2014a). The impact of topology on energy consumption for collection tree protocols: An experimental assessment through evolutionary computation. *Applied Soft Computing*, 16(0):210–222.

[6] Bucur, D., Iacca, G., Squillero, G., and Tonda, A. (2014b). The tradeoffs between data delivery ratio and energy costs in wireless sensor networks: A multi-objective evolutionary framework for protocol analysis. In *Proceedings of the Sixtienth Annual Conference on Genetic and Evolutionary Computation Conference*, GECCO '14, New York, NY, USA. ACM.

[7] Darwin, C. (1859). *On the Origin of the Species by Means of Natural Selection: Or, The Preservation of Favoured Races in the Struggle for Life*. John Murray.

[8] Fonseca, R., Gnawali, O., Jamieson, K., Kim, S., Levis, P., and Woo, A. (2007). TinyOS Extension Proposal (TEP) 123: The collection tree protocol (ctp).

[9] Fonseca, R., Gnawali, O., Jamieson, K., and Levis, P. (2006). TinyOS Extension Proposal (TEP) 119: Collection.

[10] Gnawali, O., Fonseca, R., Jamieson, K., Kazandjieva, M., Moss, D., and Levis, P. (2014). CTP: An Efficient, Robust, and Reliable Collection Tree Protocol for Wireless Sensor Networks. *ACM Transactions on Sensor Networks (TOSN)*, 10(3).

[11] Langendoen, K. (2006). Apples, oranges, and testbeds. In *In Proc. IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS '06)*, pages 387–396.

[12] Langendoen, K., Baggio, A., and Visser, O. (2006). Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture. In *Proc. Int. Conf. on Parallel and distributed processing*, pages 174–181. IEEE Computer Society.

[13] Lee, H. J., Cerpa, A., and Levis, P. (2007). Improving wireless simulation through noise modeling. In *Proceedings of the 6th international conference on Information processing in sensor networks*, IPSN '07, pages 21–30, New York, NY, USA. ACM.

[14] Levis, P., Gay, D., Handziski, V., Hauer, J.-H., Greenstein, B., Turon, M., Hui, J., Klues, K., Sharp, C., Szewczyk, R., Polastre, J., Buonadonna, P., Nachman, L., Tolle, G., Culler, D., and Wolisz, A. (2005). T2: A second generation OS for embedded sensor networks. Technical Report TKN-05-007, Technische Universität Berlin.

[15] Levis, P., Lee, N., Welsh, M., and Culler, D. E. (2003). TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *Proceedings of the ACM conference on Embedded Networked Sensor Systems (SenSys)*, pages 126–137.

[16] Liu, R., Rosberg, Z., Collings, I., Wilson, C., Dong, A., and Jha, S. (2008). Overcoming radio link asymmetry in wireless sensor networks. In *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*, pages 1–5.

[17] Liu, Y., He, Y., Li, M., Wang, J., Liu, K., and Li, X.-Y. (2013). Does Wireless Sensor Network Scale? A Measurement Study on GreenOrbs. *IEEE Trans. Parallel Distrib. Syst.*, 24(10):1983–1993.

[18] Ma, Q., Liu, K., Zhu, T., Gong, W., and Liu, Y. (2014). BOND: Exploring hidden bottleneck nodes in large-scale wireless sensor networks. In *ICDCS*, pages 399–408.

[19] Prakash, R. (1999). Unidirectional links prove costly in wireless ad hoc networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, DIALM '99, pages 15–22, New York, NY, USA. ACM.

[20] Puccinelli, D., Gnawali, O., Yoon, S., Santini, S., Colesanti, U., Giordano, S., and Guibas, L. (2011). The impact of network topology on collection performance. In *Proc. 8th European Conference on Wireless Sensor Networks (EWSN)*, pages 17–32.

[21] Ramasubramanian, V. and Mossé, D. (2008). BRA: A bidirectional routing abstraction for asymmetric mobile ad hoc networks. *IEEE/ACM Trans. Netw.*, 16(1):116–129.

[22] Sanchez, E., Schillaci, M., and Squillero, G. (2011). *Evolutionary Optimization: the μGP toolkit*. Springer Publishing Company, Incorporated, 1st edition.

[23] Sun, X., Qian, H., Wang, B., and Liu, Q. (2013). ECTP: An enhanced data collection protocol based on CTP. *International Journal of Grid and Distributed Computing*, 6(5):83–92.

[24] Werner-Allen, G., Lorincz, K., Johnson, J., Lees, J., and Welsh, M. (2006). Fidelity and yield in a volcano monitoring sensor network. In *Proceedings of the 7th symposium on Operating systems design and implementation*, OSDI '06, pages 381–396. USENIX Association.

[25] Woehrle, M., Bakhshi, R., and Mousavi, M. (2012). Mechanized extraction of topology anti-patterns in wireless networks. In Derrick, J., Gnesi, S., Latella, D., and Treharne, H., editors, *Integrated Formal Methods*, volume 7321 of *Lecture Notes in Computer Science*, pages 158–173. Springer Berlin Heidelberg.

[26] Woo, A., Tong, T., and Culler, D. (2003). Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, pages 14–27, New York, NY, USA. ACM.

[27] Xu, K., Tipper, D., Qian, Y., Krishnamurthy, P., and Tipmongkonsilp, S. (2014). Time varying performance analysis of multihop wireless networks with cbr traffic. *IEEE Transaction on Vehicular Technology*.

[28] Zhao, J. and Govindan, R. (2003). Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, pages 1–13, New York, NY, USA. ACM.