Demos: Robust Orchestration for Autonomous Networking

Andreas Biri
ETH Zürich, Switzerland
abiri@ethz.ch

Marco Zimmerling
TU Darmstadt, Germany
marco.zimmerling@tu-darmstadt.de

Lothar Thiele
ETH Zürich, Switzerland
thiele@ethz.ch

Abstract

Research in wireless sensor networks has resulted in a remarkable breadth of highly capable systems. However, while specialized protocols perform well in the setting they were designed for, they often lack the ability to quickly adapt once operating conditions change drastically. Of particular importance is resilience to node and link failures, as clusters of nodes that lost their leader or split apart need to re-organize and find each other again. With Demos, we present a low-power wireless protocol that ensures robust network orchestration despite such failures. Demos rapidly finds consensus on leadership with its cluster coordination mechanism even if the set of nodes fluctuates by introducing new election quorums. In addition, a novel cluster discovery scheme enables autonomous clusters to merge on the fly and maximize network coverage. Experiments with controlled mobility on a multi-hop network of 24 nodes demonstrate that Demos maintains a reliable data exchange despite severe disruptions and adapts to changes within seconds. We further find that Demos' ability to continuously coordinate and discover achieves highly robust orchestration of fully autonomous clusters.

Categories and Subject Descriptors

C.2.1 [**Computer-communication Networks**]: Network Architecture and Design—*Distributed networks, wireless communication*; C.2.2 [**Computer-communication Networks**]: Network Protocols; C.3 [**Special-purpose and Application-based Systems**]: Real-time and embedded systems

General Terms

Design, Experimentation, Reliability

Keywords

Fault tolerance, Consensus, Network orchestration, Concurrent transmissions, Wireless sensor networks

1 Introduction

While many wireless sensor networks (WSNs) rely on static nodes and infrastructure such as data sinks, a growing class of applications features mobile nodes and autonomous operation. Examples of particular societal and industrial importance range from human sensing [25, 44] to multi-agent systems [22, 27]. Re-configurable hardware [4], flexible physical layers [7], and data-driven media access [40] provide the necessary robustness and adaptivity for such applications on the link layer. However, current network-layer solutions still fall short of fulfilling their requirements.

Problem. Imagine a swarm of drones in a search-and-rescue scenario. It requires continuous coordination and data exchange among drones over a wireless multi-hop network to jointly localize human victims [50, 51] or to safely navigate complex spaces [27, 35]. During a mission, the network may split into multiple clusters voluntarily (e.g., when multiple regions of interest are identified) or unexpectedly (e.g., due to sudden non-line-of-sight conditions). Using current network-layer protocols, such a network split causes a loss of coordination for a subset of drones [17, 24, 39, 54] or renders the drones unable to reconnect and merge into one complete network [32, 33, 34], thus endangering mission success.

The root of this problem is that nearly all existing network-layer solutions centrally orchestrate the nodes. Within a *cluster* of connected nodes, which may comprise all or only a subset of the nodes in a network, a *coordinator* distributes information to synchronize the nodes and instructs them on when each one may communicate. If the coordinator becomes unavailable, orchestration is halted and the data exchange breaks down. While node mobility [25, 44] can cause such disruptions, static clusters may also be affected due to node or link failures [1, 53]. Despite its relevance, current designs [24, 32, 33, 39, 54] cannot quickly and safely appoint a new coordinator after arbitrary node and link failures. While some protocols permit nodes to re-establish coordination after a network splits into clusters, they lack the means for these clusters to discover each other and merge [20, 34].

For robust orchestration despite node and link failures, a protocol needs to (*i*) accurately identify nodes in a cluster and their traffic demands, (*ii*) maintain reliable communication in spite of frequent network topology changes, and (*iii*) discover and merge clusters. These features enable autonomous networking as required by emerging applications.

Contribution. We present *Demos*, the first low-power wireless protocol that provides robust network orchestration for autonomous multi-hop networking. Demos features a novel consensus mechanism to reliably determine a cluster coordinator and to ensure that cluster information remains up-to-date. By minimizing network state, Demos swiftly adapts its data exchange and instantly reacts to changes in the network. Key to this robustness is Demos' ability to preserve orchestration in case the current coordinator becomes unavailable based on newly introduced election quorums. Combined with a unique cluster discovery scheme, nodes always remain in exchange with connected nodes and quickly merge clusters to maximize coverage. By building on concurrent transmissions [19, 29], Demos reliably propagates information while being agnostic to network topology changes. This flexibility enables the protocol to inherently support frequent network fluctuations and high node mobility.

This paper makes the following major contributions:

- We introduce Demos, the first low-power wireless protocol that achieves robust network orchestration despite arbitrary node and link failures.
- We propose novel methods for cluster coordination and discovery, enabling autonomous clusters that can dynamically split and merge to increase network coverage.
- We provide an open-source implementation of Demos and demonstrate its ability to robustly orchestrate nodes under challenging conditions in testbed experiments.

Based on the application and protocol requirements specified in Section 2, we give an overview of Demos in Section 3. We describe the protocol in more detail in Section 4 and provide a formal analysis in Section 5. In Section 6, we test Demos in various mobility and failure scenarios and demonstrate its ability to adapt on the spot and ensure maximal connectivity.

2 Motivation and Challenges

We first discuss the application requirements, deduce prerequisites for robust network orchestration, and then identify the challenges we need to overcome in the design of Demos.

2.1 Application and Protocol Requirements

Application requirements. Autonomous systems typically feature different traffic categories (e.g., control, coordination, and application data [22]). As each category differs in bandwidth and temporal resolution, the application must be able to specify and adapt its traffic demands. Especially in multi-agent systems, reliable real-time traffic is indispensable, with requirements changing over time depending on the developing scenario (e.g., switching from searching for a victim to streaming video feeds after localization [22]).

To support mobile networks such as drone swarms, consistently maintaining connectivity such that no node remains isolated is paramount [50, 51] and should be temporally and spatially robust. However, due to large coverage areas and precarious environments, multi-hop communication is often a must [22] and needs to cope with a dynamic topology [31].

Most important, however, is the ability of nodes to self-organize into an autonomous network by detecting nearby nodes and establishing communication. Nodes may frequently encounter network fluctuations [51] and seek to dynamically exchange information whenever possible [50].

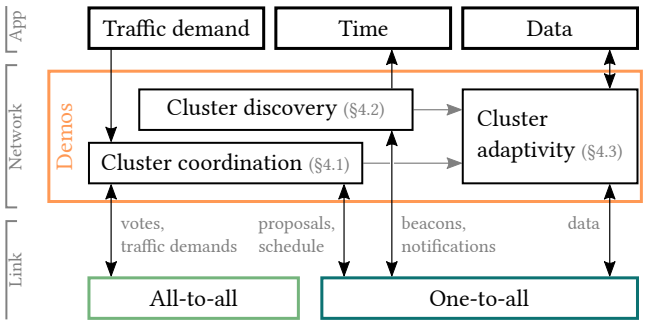

Figure 1. Demos provides network orchestration for the application. Based on two link-layer primitives, it establishes coordination, discovery, and adaptivity on the network layer.

Protocol requirements. From the preceding application requirements, we deduce the following requirements for a suitable network-layer protocol. To cater to strict and challenging traffic demands, communication should be scheduled to prevent contention and ensure efficient use of the limited bandwidth [17]. Reliable scheduling requires the collection of traffic demands from the currently participating nodes and uniquely assigning time slots to avoid packet collisions.

To cope with time-varying communication links, network orchestration should have minimal dependence on the current topology. As a consequence, the protocol should be sufficiently robust to ensure that the exchange of data is preserved despite unexpected node and link failures.

Lastly, to autonomously and rapidly coordinate communication without the potential for contradicting decisions, each cluster should choose a unique cluster coordinator. The election of a coordinator must fulfill the conditions of agreement (i.e., all connected nodes elect the same node), termination (i.e., if there are valid candidates, one must eventually be elected), validity (i.e., the elected node must be part of the cluster), and stability (i.e., a new coordinator is only elected if the previous one has become unavailable) [5, 15].

2.2 Challenges

For a protocol design that autonomously schedules traffic, supports reliable multi-hop networking, and robustly orchestrates a network as outlined above, Demos must be able to

(i) elect a coordinator in a cluster of unknown size,
(ii) preserve communication when a cluster splits away,
(iii) continuously discover other nearby clusters, and
(iv) merge clusters to maximize network coverage.

Next, we investigate each challenge in more detail and identify existing shortcomings that motivate Demos' design.

Electing a cluster coordinator. Most leader election algorithms rely on knowing the exact network size [45] and only consider election a start-up problem [2] or after long periods of inactivity (e.g., 2 min [20]). However, our application may require elections at any moment because node and link failures can cause a leader to suddenly disappear. Elections should only occur if the current coordinator is unavailable to preserve stability and must result in a unique decision for agreement based on votes from an unknown set of nodes.

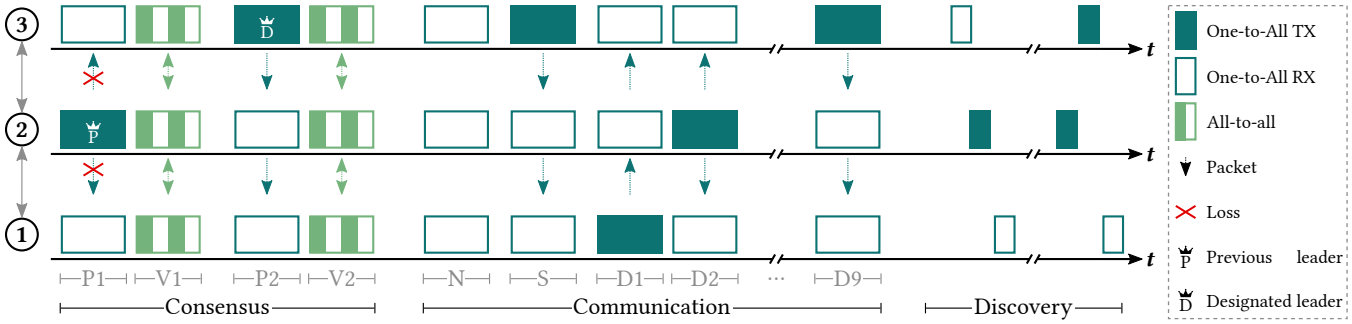

Figure 2. A Demos round consists of three phases: *Consensus*, *communication*, and *discovery*. During consensus, E pairs of proposal (P) and voting (V) slots are used for cluster coordination ($E = 2$ in this illustration). While the first pair is assigned to the leader of the previous round, later ones are designated so other nodes may get newly elected. The communication phase consists of a notification (N), a scheduling (S), and multiple data (D) slots. For discovery, nodes interact with other clusters.

Network split. Two clusters that separated could continue to use the same schedule. For at least one cluster, however, the coordinator is no longer available. So far, protocols then usually fall back to bootstrapping after a prolonged timeout [20]. Instead, the leaderless nodes may swiftly initiate an election and autonomously establish the new traffic demands.

Cluster discovery. After re-establishing cluster coordination, clusters may operate independently. However, such splits lead to a gradual decay of the network into patches with limited individual coverage. For recombination, a secondary radio could be used in parallel [10, 25] to look for other clusters in the vicinity without affecting the data exchange on the primary radio. To avoid such an increase in complexity and costs, a holistic protocol design may instead combine data exchange and discovery. However, most neighbor discovery protocols depend on a fixed structure [18, 42] as they are not co-designed with a scheme to exchange application data. Furthermore, they focus exclusively on the pairwise discovery of nodes and do not cater to multi-hop networks.

Cluster merging. Once another cluster has been discovered, a node could independently leave its previous cluster and synchronize to the new one. While this straightforward approach based on pairwise discovery and individual decision-making is predominant [13, 52], such a design only gradually increases network coverage. Even worse, as discovery can be bi-directional, merging could be simultaneously attempted in both directions and data exchange may collapse as nodes attempt to synchronize to disbanded clusters. Alternatively, nodes may leverage cluster-internal communication and exchange their discoveries to take a collective decision, but long delays for merging clusters could remain an issue.

3 Designing Demos

Demos aims to achieve one of the early visions of WSNs: To facilitate the distribution of sensor nodes in a deployment area, which would then automatically coordinate and communicate by themselves without detailed preconfiguration [36]. To this end, Demos provides robust network orchestration which enables nodes to constantly exchange data and maximize their coverage despite a fluctuating network topology. By introducing dynamic cluster coordination, the protocol tackles the first two challenges presented in Section 2.2 and ensures that nodes maintain coordination even after a network splits into multiple clusters. With its cluster-wide discovery scheme, Demos overcomes the remaining challenges and offers an autonomous, recuperating communication service among all connected nodes. In the following, we first present the overall design principle before giving an overview of Demos' structure and protocol phases.

3.1 Demos in a Nutshell

Demos is a network orchestration protocol that continuously re-elects a cluster coordinator and aggregates information on all connected nodes. During this election, the coordinator periodically collects the votes and traffic demands of nodes in the cluster and then assigns exclusive time slots through scheduling. For this communication, the cluster uses a frequency channel that directly depends on its coordinator to prevent interference with others. However, if no coordinator is available, the nodes follow a shared randomized sequence of node IDs to deterministically elect a new leader. After the exchange of application data, a distributed cluster discovery scheme detects neighboring clusters and notifies the cluster coordinator. The entire cluster may then decide to merge simultaneously with its neighboring cluster.

To support such abrupt changes in the network topology and provide a robust data exchange despite high node mobility, Demos builds on a link layer utilizing concurrent transmissions [55], as displayed in Figure 1. Such network flooding is topology-agnostic and permits reliable communication due to its inherent redundancy. Based on these primitives, Demos provides novel mechanisms for the coordination (Section 4.1) and discovery (Section 4.2) of clusters as well as for adaptivity to changing conditions (Section 4.3).

3.2 Protocol Structure

Demos is structured into rounds of period T which all nodes of a cluster $C \subseteq \mathcal{N}$ execute synchronously, where $\mathcal{N} = \{i \in \mathbb{N} \mid 1 \leq i \leq N\}$ is the total set of nodes in a network. Each round consists of three phases, as illustrated in Figure 2. To elect a coordinator, the cluster votes on the proposing nodes during the *consensus* phase and exchanges the current traffic demands. During the *communication* phase, a contention slot can be used to notify the cluster coordinator

of discovered clusters. If the coordinator decides to merge with another cluster, it then instructs its cluster to disband, as further described in Section 4.2. Otherwise, it broadcasts a schedule which is subsequently followed to sequentially disseminate application data. In the final *discovery* phase, nodes use the rest of the round to listen for and advertise to other clusters in the vicinity. If another cluster is discovered, the coordinator is notified in the next round.

Communication primitives. Demos builds on two complementing types of concurrent transmissions, a *one-to-all* and an *all-to-all* primitive. One-to-all floods, as introduced by Glossy [19], are employed for most slots of the protocol and consist of a packet that is initially broadcasted by a single node and then concurrently forwarded by each receiver. An all-to-all primitive like Chaos [29], on the other hand, is used during the consensus phase to efficiently propagate the vote and traffic demand of each node through the network by continuously merging information during consecutive sub-slots. Combining these primitives enables Demos to aggregate information in parallel from an unknown set of nodes with an all-to-all exchange and then schedule one-to-all floods that reliably reach their target with minimal latency.

Deterministic randomization. Demos repeatedly seeds a number generator to share randomized state without explicit distribution. Sequences produced by this deterministic generator are used to (*i*) create a non-repeating sequence of designated leaders for the consensus phase, (*ii*) map the node ID of a cluster coordinator to a frequency channel on which the cluster communicates, and (*iii*) assign non-overlapping reception (RX) and transmission (TX) slots for discovery.

3.3 Consensus Phase

The consensus phase consists of E pairs of *proposal* (`P`) and *voting* (`V`) slots. The first pair is reserved for the previous leader that coordinated the cluster in the last round to provide stability. If the election is unsuccessful, as explained in Section 4.1, the remaining pairs are designated to potential successors. A designated leader that is part of the cluster proposes if it has not yet voted in the current round and can be elected as the new cluster coordinator upon success.

Proposal. Apart from the first `P` slot in which the previous leader can propose, the remaining $E-1$ slots are designated according to a randomized sequence containing the total set of nodes $\mathcal{N}$. The offset inside this sequence is based on the *round counter* r and the *election counter* $e \in [1, E)$ and is synchronized across the cluster. The designated node then proposes to ask others for their vote in the current round.

Voting. A node that received a proposal casts its vote if it has not already done so in an earlier `V` slot of the same round. Together with its current traffic demand, votes from all nodes are then exchanged within the cluster and continuously merged. At the end, the designated leader locally determines the result of the election, as detailed in Section 4.1.

3.4 Communication Phase

After having established consensus on the cluster coordinator and aggregated the current demand, the communication phase is executed. A *notification* (`N`) slot permits nodes to inform the coordinator about other discovered clusters. The cluster coordinator then decides whether a merge should be conducted, as explained in Section 4.2. The following *scheduling* (`S`) slot is used to either disseminate information on the cluster to be merged with or to distribute a schedule based on the current demands of the nodes. To avoid that missing demand information occasionally causes nodes to not be scheduled, a received demand is valid for up to P rounds. Lastly, application data is exchanged during multiple consecutive *data* (`D`) slots if a schedule has been received.

Notification. To extend pairwise to cluster-wide discovery, a node floods a received beacon containing information on a discovered cluster so it reaches the coordinator. As access is contention-based, only one of potentially multiple discovered clusters is successfully reported per round.

Scheduling. The `S` slot enables the coordinator to instruct the cluster on how to continue. If it has decided to merge, it propagates the necessary information so its cluster can align itself to the communication and frequency channel of the new cluster. Otherwise, the coordinator distributes a schedule based on the previously gathered traffic demands, including a bitmap containing the current members of the cluster.

Data. Each `D` slot is assigned to a node that may initiate a one-to-all flood of application data. All others assist in propagating a received packet by retransmitting it concurrently.

3.5 Discovery Phase

The discovery of other clusters occurs during the remaining round time on a dedicated frequency channel that is shared among all clusters. To guarantee overlapping discovery phases between clusters, we presume that the phase duration T_d exceeds $T/2$. To permit the adaptation of T_d depending on the currently demanded number of `D` slots and round period T, Demos employs probabilistic discovery [36]. The discovery phase is split into slots of fixed length, during which nodes either transmit a beacon (`B`), listen, or remain idle based on given probabilities. Building on this scheme, Demos leverages cluster coordination to increase its energy efficiency. As the communication ranges of nodes in a cluster overlap and they would hence operate redundantly if they simultaneously perform discovery, we deterministically divide discovery across the cluster by assigning non-idle slots to nodes. If another cluster is discovered, the cluster coordinator is informed in the next `N` slot, as shown in Figure 3.

4 Demos in Detail

Next, we focus on the two key concepts that enable Demos to form autonomous clusters, cluster coordination and cluster discovery. Thereafter, we examine how the protocol leverages its flexibility and adapts to changing conditions.

4.1 Cluster Coordination

Without packet loss, the election of a coordinator is undisputed as all nodes of a cluster receive the first proposal and cast their votes accordingly. However, the minimum required number of votes in favor to safely call elections, called *quorum*, is non-trivial when considering imperfect communication and variable cluster sizes. Based on a subset of votes, the cluster must determine a unique leader. We develop two novel methods to define quorums that enable nodes to still find agreement on a coordinator under such circumstances.

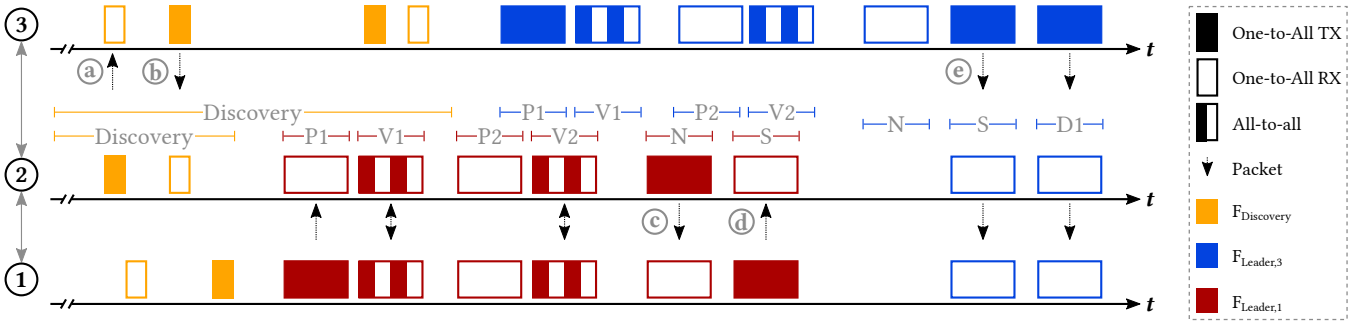

Figure 3. During discovery, a beacon from a leader with a lower ID will be ignored (*a*). To merge with a dominating cluster, the received beacon (*b*) will be forwarded to notify the current cluster coordinator (*c*). The leader then instructs nodes to disband and join the discovered cluster on the corresponding frequency channel (*d*), where they synchronize to the new coordinator (*e*).

Absolute quorum. The classical solution to guarantee a unique decision is to require agreement from a majority of nodes [3, 28, 38, 39]. Therefore, the *absolute quorum* Q_a can be defined as $Q_a := F > N_c/2$, where F is the number of votes in favor of the proposal and N_c is the cluster size.

However, N_c varies over time. To accurately keep track of the cluster size, each node locally estimates N_c and updates this value each round as follows. Nodes include their current estimate in the exchange during the `V` slot and store the maximum of all received estimates as N_{net}. After the consensus phase, they further count the number of votes they have encountered at least once as N_{votes} (independent of whether they are in favor or against). Updating the estimate of N_c then occurs in two steps. First, a network filter sets $N_c \leftarrow \lfloor \alpha \cdot N_{net} + (1-\alpha) \cdot N_c \rfloor$, with $\alpha \in [0,1]$. Thereafter, a temporal filter is applied to get $N_c \leftarrow \lfloor max(\beta \cdot N_{votes} + (1-\beta) \cdot N_c, N_{votes}) \rfloor$, with $\beta \in [0,1]$. If a node has been elected as cluster coordinator, it distributes its own estimate of N_c with the schedule during the `S` slot so the entire cluster adopts the value of the leader.

This mechanism leverages centralized updates of N_c to equalize estimates when a coordinator is available and otherwise enables nodes to autonomously adapt until a new leader is elected. First, the network filter applies α to warrant that all nodes of a cluster use similar values and do not underestimate the cluster size to avoid multiple simultaneous leaders. A high α prevents a strong decrease when packets were locally missed but have been received by others. Lowering α limits the influence of a single well-connected node if most only receive a subset of votes so that chances for successful elections remain high. The temporal filter integrates β to ensure that the estimate converges to the true size over time. If more nodes than expected have been heard, the empirically validated number of nodes is adopted to prevent an underestimation. Otherwise, the new estimate is weighted with β for a gradual transition that smooths temporal fluctuations.

Relative quorum. The absolute quorum directly depends on an accurate estimate of N_c to safely call a vote. As the quorum is particularly relevant when a new leader needs to be elected after nodes split apart and N_c changed drastically, waiting until the estimates converge to the true value may prevent the exchange of data for several rounds. To expedite elections in highly mobile scenarios, we introduce a *relative quorum* Q_r that only depends on the fraction of votes in favor of the proposal as gathered during the ongoing election.

To accomplish independence from N_c, we leverage the known high reliability of concurrent transmissions [55]. Suppose that for a connected cluster of N_c nodes, a one-to-all proposal reaches at least $N_o = \gamma_o \cdot N_c$ nodes, with $\gamma_o \in [0,1]$. The designated leader then collects votes using the all-to-all exchange in the `V` slot, with each of the N_o nodes that have received the proposal submitting a vote. At least $\gamma_a \cdot N_o$ submitted votes reach the designated leader, with $\gamma_a \in [0,1]$. In other words, the leader receives $V \in [\gamma_a \cdot \gamma_o \cdot N_c, N_c]$ votes and we can therefore bound $N_c \in [V, \frac{V}{\gamma_a \cdot \gamma_o}]$. As for the absolute quorum, a result is unique if $F > N_c/2$ is satisfied. Using the derived upper bound, a sufficient (but not necessary) condition is $F > \frac{V}{2 \cdot \gamma_a \cdot \gamma_o}$. We can hence find a quorum that is entirely based on the votes V of the current `V` slot as

$$Q_r := \frac{F}{V} > \frac{1}{2 \cdot \gamma_a \cdot \gamma_o}$$

Extended absolute quorum. Knowledge of γ_o can also be exploited to improve Q_a. As at least N_o nodes receive the first proposal and vote in its favor, at most $N_c - N_o = (1-\gamma_o) \cdot N_c$ votes remain uncast if a proposal was sent. It hence suffices to receive $F > (1-\gamma o) \cdot N_c$ votes in favor to assert that no other leader was previously elected, enabling us to define

$$Q_{a+} := F > min(\frac{1}{2}, 1-\gamma_o) \cdot N_c$$

Both absolute quorums permit nodes to skip consensus after having voted but rely on N_c and may result in multiple leaders per cluster if the estimate does not reflect the actual cluster size. While the relative quorum guarantees safe elections if γ_o and γ_a are valid assumptions, it requires participation in all E elections per round and thus needs more energy. Demos supports the use of all three quorums, whose performance we compare in real-world experiments in Section 6.4.

4.2 Cluster Discovery

With its cluster coordination, Demos enables dynamic cluster management that tolerates faults and permits network splits. However, to maximize its connectivity, the protocol

also requires a mechanism to detect and merge clusters. Demos builds on an established neighbor discovery scheme and extends it to *cluster-wide discovery* to increase coverage. We thereby leverage knowledge of other nodes in the cluster to cooperate and distribute discovery across all of them.

Continuous discovery. In contrast to current WSN protocols that treat discovery as an isolated task during bootstrapping, Demos must retain the ability to find and merge clusters while continuously exchanging data in parallel. As traffic demand is dynamic and constantly influences the number of scheduled `D` slots as well as the round period, the remaining time for discovery is restricted and requires a highly adaptive mechanism. The Birthday protocol [36] provides flexible pairwise discovery and achieves low detection latencies despite these constraints, as it does not require a fixed structure and permits clusters to independently adjust parameters. By splitting the discovery phase into slots during which nodes transmit beacons with probability P_t and listen with P_l, discovery is likely despite a low duty cycle of $P_t + P_l$.

Cluster-wide discovery. Demos extends this concept with novel mechanisms to reduce energy consumption and discovery latency. First, we leverage that Demos coordinates an entire cluster and divide discovery equally across nodes instead of performing it redundantly in parallel. By prohibiting overlapping discovery slots within a cluster, we avoid the scalability issues of other discovery protocols due to colliding transmissions [26] and preclude inefficient simultaneous listening periods. Assigning discovery slots to nodes merely requires including the cluster members as a bitmap in the distributed schedule and randomizing their order locally using a deterministic pseudorandom generator. Second, we exploit that Demos ultimately aims to discover clusters and not individual nodes, for which a single node receiving a beacon from a node in another cluster suffices. By notifying the entire cluster via the coordinator, all nodes can then switch simultaneously without having to pause their data exchange.

Discovery interface. To merge with another cluster, a coordinator must be able to align the communication of its own cluster accordingly. The beacon distributed by a node during discovery contains the relative time since the round started, the round period T, the round counter r, the ID of the cluster coordinator, and N_c to enable this synchronization. Knowing the start of the last round and T, another cluster may directly reorient itself to the upcoming round and can infer the current offset in the sequence of designated leaders based on r. To remain valid, the time passed since the reception of the beacon is added to the received time since the start of the round when the beacon is forwarded during the `N` and `S` slots.

Merge criteria. When the coordinator receives information on a discovered cluster, it must determine whether to merge. To avoid that two clusters attempt to simultaneously merge with each other, Demos relies on the ID of the cluster coordinator and merges a cluster led by a node with a lower ID into the cluster of a leader with a higher ID, as visualized in Figure 3. To prevent a single node with a high ID from temporarily disrupting the communication of a much larger cluster, N_c can also be included as a merge criterion, with a larger size dominating and the ID merely used to break ties.

4.3 Cluster Adaptivity

Building on top of robust cluster coordination and a network structure that continuously adjusts to the underlying topology, Demos can adapt essential protocol parameters to cater to the current application requirements. To maximize its flexibility and reduce complexity, the protocol does so by deliberately maintaining minimal persistent network state.

Traffic demand. To support high node mobility, Demos must expect frequent changes in the coordinator and the set of nodes belonging to a cluster. Therefore, classical solutions such as explicit registration when joining [10, 43] and timeouts to remove unavailable nodes [20] are not applicable. To continuously adapt the communication schedule according to current demand, Demos includes requests directly in the all-to-all communication of the `V` slot. As this exchange ensures that the demands of all nodes in the cluster are always up-to-date at the elected leader despite network splits and merges, it can immediately react to changes.

Round period. Many round-based protocols adopt a fixed round period T to circumvent scheduling [23, 29] or temporally improve liveness by maintaining communication despite the unavailability of the coordinator [32, 33]. However, such a limitation severely restricts the flexibility of a protocol, as it fixes parameters such as the frequency of cluster coordination, the number of data slots, and the likelihood of discovery. It further constrains the protocol in reacting to changing real-time scheduling requirements. Demos permits the cluster coordinator to adapt T and communicate its current value in the `S` slot. This flexibility is made possible by its robust cluster coordination, as Demos tolerates missing such an update by quickly re-electing a leader after desynchronization. Combined with cluster discovery, the protocol rejoins nodes that are out-of-sync and thereby ensures continued connectivity while offering high adaptivity.

Frequency channel. Demos employs separate frequency channels per cluster based on the node ID of the cluster coordinator to prevent packet collisions when clusters encounter each other. This mechanism further inherently avoids heavily contested channels due to external interference, as it causes nodes to synchronize with leaders on reliable frequencies. By voluntarily yielding as coordinator if the packet reception rate drops below a given threshold, Demos supports blacklisting of frequencies to boost its robustness.

4.4 Practical Example

To illustrate Demos' concepts, Figure 4 depicts how a network splits and merges again. The previous leader proposes itself for re-election in the first `P` slot of each round to confirm the availability of the coordinator. However, a change in the network topology causes the network to temporarily split into two clusters in round 2. While leader 1 can still coordinate one cluster, the other consisting of nodes 3 and 4 must wait for a designated `P` slot to elect a new leader. The stable re-election of leader 1 is made possible by our novel relative quorum, as the cluster size N_c could not be re-estimated yet and would have prevented a successful election based on an absolute quorum. With two autonomous clusters led by nodes 1 and 4, the cluster discovery mechanism enables them to merge once the communication link between nodes

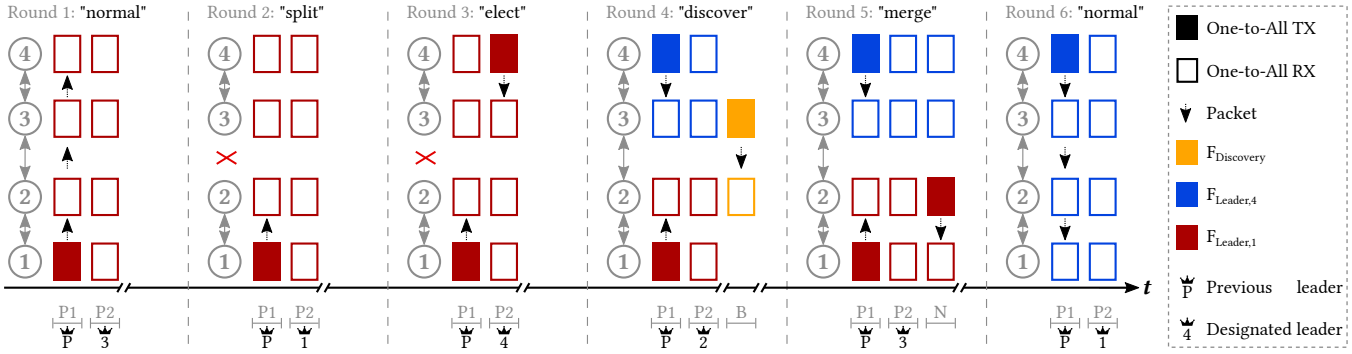

Figure 4. In round 1, a network of four nodes coordinated by node 1 successfully re-elects its previous leader. After a topology change breaks the communication link between nodes 2 and 3 in round 2, the network splits into two clusters. As soon as a P slot is designated to a node from the leaderless cluster in round 3, nodes 3 and 4 independently resume the data exchange. After the link is re-established in round 4, node 2 notifies node 1 about a received beacon (B) and the clusters merge in round 5.

2 and 3 recuperates in round 4. As the clusters operate on a separate frequency band, packet collisions are prevented during the discovery and merge process. The beacon containing the cluster information is received by node 2 and forwarded to the leader using the N slot, whereafter node 1 decides to merge due to its lower ID and the network is reunited again.

5 Formal Analysis of Election Latency

We now formally investigate the expected number of elections if a new coordinator is required. After a network splits, at least one new cluster coordinator must be elected. As Demos cycles through a non-repeating randomized sequence of the total set of nodes $\mathcal{N}$ to designate leaders in the consensus phase, knowing the expected latency until a new cluster coordinator can be established is essential to estimate when communication may resume. Supposing a cluster size N_c and $N = |\mathcal{N}|$, the probability that a designated leader is not part of the cluster is $(N - N_c)/N$. We find $p_L = 1 - \frac{(N-N_c)\cdot(N-N_c-1)\cdot\ldots\cdot(N-N_c-L+1)}{N\cdot(N-1)\cdot\ldots\cdot(N-L+1)} = 1 - \frac{(N-N_c)!\,(N-L)!}{N!\,(N-N_c-L)!}$ that a leader is successfully elected within L tries. Similarly to p_L, we derive the expected number of tries until success

$$\mathrm{E}[L] = \sum_{L=1}^{N-N_c+1} L \cdot \frac{(N-N_c)!\,(N-L)!\,N_c}{N!\,(N-N_c-L+1)!} = \frac{N+1}{N_c+1}$$

For example, if a network of $N = 24$ nodes splits in half, Demos only requires 1.92 elections on average to establish a new leader. As shown in Figure 5, even if only a small cluster of $N_c = 5$ nodes separates, using $E = 3$ it likely already elects a new leader in the second round. These analytical results closely match our experiments in Section 6.4.

6 Evaluation

Demos is designed to provide high robustness and flexibility even under challenging conditions. To examine the protocol in action, we test (*i*) its ability to maintain cluster coordination despite node and link failures, (*ii*) the adaptivity to a fluctuating network topology that separates and combines clusters, (*iii*) the performance of the quorums introduced in Section 4.1, and (*iv*) Demos' energy efficiency with its novel cluster coordination and discovery mechanisms.

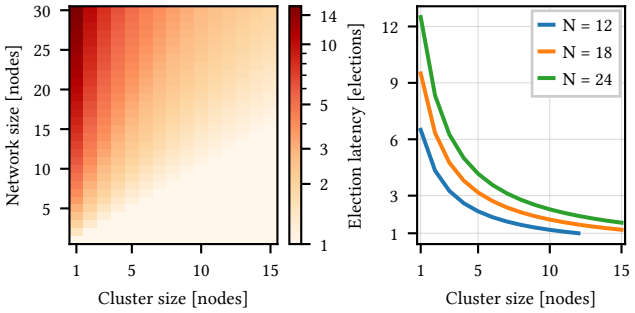

Figure 5. For all network sizes, Demos only requires few elections on average until a new designated leader proposes for clusters containing more than a small fraction of nodes.

6.1 Implementation

To support a wide coverage area using LoRa and also provide more efficient GFSK communication if a shorter range suffices, we utilize a Semtech SX1262 RF transceiver driven by an STMicroelectronics STM32L433CC microcontroller. Operating in the 868 MHz band, its broad TX power range from −9 to +22 dBm enables further adaptation to the targeted communication range. For our indoor tests, we use the GFSK modulation at 250 kbps and a TX power of +7 dBm.

Slot primitives. All-to-all communication during the V slot is based upon Chaos [29], with all nodes of a cluster probabilistically initiating the exchange of votes and demands. One-to-all floods, as employed in the P, N, S, and D slots, use Glossy [19] with consecutive transmissions for reliability [30]. For the discovery phase, one-to-all beacons are sent in Glossy slots with an increased RX duration. Multiple consecutive transmissions boost the discovery reliability and ensure overlaps with listening slots of other clusters in the discovery phase. Furthermore, beacon flooding quickly spreads information when multiple clusters converge simultaneously.

Randomization. Demos integrates two different sources for randomization, a built-in hardware unit that produces truly

random output as well as a deterministic generator yielding pseudorandom numbers based on a given seed value. The former ensures that the usage of the Chaos sub-slots [29] differs between nodes and data can be exchanged, while the latter initially generates a fixed randomized sequence for designating leaders and the mapping of leader IDs to frequency channels. The designated leader is then selected in each election e using a sequence offset $(E-1)\cdot r+e$, where E denotes the number of elections per round and r is the round counter. Before each discovery phase, the deterministic number generator is re-seeded based on the ID of the cluster coordinator and r to locally compute the slot assignments for discovery.

Research artifacts. We provide the source code of Demos as open source [9]. Additional test scripts grant extensive control of experiments on FlockLab [48] by dynamically activating nodes at run time, offer the ability to easily sweep parameters, and visualize results to study their impact.

6.2 Experimental Setup

Test scenario. To test Demos under realistic conditions in confined spaces where non-line-of-sight conditions partition the network into clusters, we utilize the FlockLab testbed [48]. 24 nodes are spread across one floor of an office building and provide fixed node locations. In addition, we employ a mobile node that is moved to dynamically change the network topology. We use four configurations to examine Demos' effectiveness in a variety of scenarios. 12 nodes with a mean hop distance of 1.31 form the smallest network, while an extended set of 18 nodes has an average of 1.65 hops, and all 24 nodes communicate via 1.84 hops on average. Figure 6 illustrates the fourth setup where a mobile node connects two clusters of static nodes.

Protocol parameters. For our tests, we use a round period of $T = 3$ s. By default, we conduct $E = 2$ elections during the consensus phase, with each `V` slot split into 36 all-to-all sub-slots to exchange votes and demands. A received demand is valid for up to $P = 10$ rounds. Each node requests 1 data slot per round and sends 20 bytes using 3 transmissions per flood in one of 30 `D` slots. The remaining time of $T_d = 0.56 \cdot T$ is used for discovery. Due to Demos' fast discovery and merging mechanism, we do not expect multiple clusters to converge simultaneously and therefore we set $P_t = P_l = 0.5$ to minimize the discovery latency [36]. The clusters share 14 non-overlapping frequency channels for communication, with an additional one reserved for discovery. We set $\alpha = 0.5$ and $\beta = 0.33$ and find safe lower bounds $\gamma_o = \gamma_a = 0.9$ based on an empirical packet reception rate (PRR) exceeding 97.6 % for one-to-all and 93.1 % for all-to-all primitives in the three static configurations.

Baseline. To investigate the impact of Demos' novel cluster coordination and discovery mechanisms, we compare it to LWB [20], a well-known protocol that builds on the same one-to-all floods to exchange data. While LWB relies on a preconfigured *host node* that coordinates the network, it is one of the only WSN protocols that include a failover policy if the coordinator is unavailable. In case of a host failure, nodes independently hop across frequency channels and try to reach the fixed coordinator on this channel. As in Demos' `S` slot, the host distributes a schedule for a subsequent succession of data slots that correspond to Demos' `D` slots. Demand is registered using a contention slot and assumed to be constant for multiple rounds. Exploiting this limitation, the host sends the next schedule already at the end of a round and retransmits it at the start of the next one to boost the reception probability of schedules.

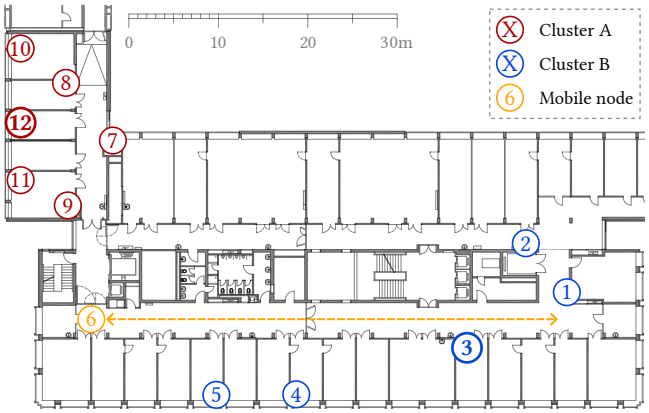

Figure 6. A mobile node traverses a corridor from left to right and then returns to its start position, thereby changing the network topology between two clusters of nodes.

6.3 Robustness to Network Topology Changes

To investigate how Demos reacts to changes in the network topology, we start with a static experiment and examine the behavior of the protocols when the coordinator fails. In a second experiment, we instead include a mobile node and cause the network to first separate into two clusters and recombine again afterward. Throughout the experiments, we monitor the PRR, i.e., the percentage of received data packets compared to the maximal number of data packets that could have been received by all nodes in the network (i.e., sent packets times the number of receivers).

Scenario: Node failure. We employ 12 nodes distributed across the testbed, as depicted in Figure 6. 11 of these nodes are static, whereas node 6 is mobile. While Demos uses the relative quorum Q_r, presented in Section 4.1, to elect its leader dynamically, LWB is configured to use node 12 as the host ("H1"), with node 3 serving as a failover ("H2") [20]. The timeout until nodes switch to the failover is set to 1 min.

In a first experiment, we keep node 6 static and let Demos elect a leader. After 60 s, we trigger a node failure by turning node 12 off and observe the behavior of Demos and LWB.

Results: Node failure. Figure 7 shows the measured PRR of Demos and LWB as well as the optimal PRR over time. We find that all Demos nodes quickly discover each other and form a network, as Demos gathers and updates the current traffic demands instantly by aggregating them in the `V` slot directly preceding scheduling. At 60 s, we observe that the PRR dips as no more packets can be received by the faulty node 12. Even though node 12 was the leader as others merged with its cluster due to its high node ID, the remaining nodes directly elect a replacement when the previous leader is unavailable and seamlessly continue communication with-

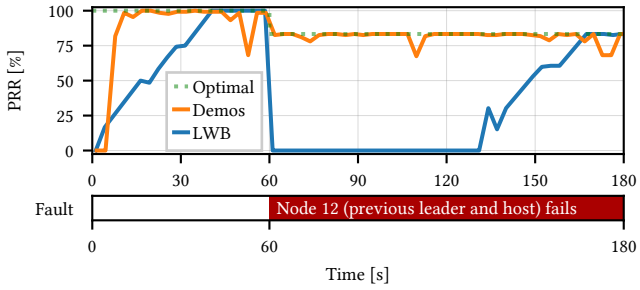

Figure 7. Failure of the host node halts LWB's communication until a failover takes over, while Demos instantly elects a new leader and approaches the optimal PRR at all times. Missed schedules cause minor PRR drops for both protocols.

out a single missed round. LWB on the other hand adapts much slower to changing traffic demands, with at most a single node sending its demand per round. The protocol further suffers from a catastrophic drop in PRR once the host fails as no schedule is distributed anymore. After the nodes time out while listening for the host, the failover host slowly accumulates the demands again until LWB finally approaches the optimal PRR as well, 107 s slower than Demos.

Scenario: Link failure. In a second experiment, we investigate how the failure of an essential link between two clusters influences their data exchange. We let the nodes establish a network and start moving node 6 from left to right after 60 s. Once arrived at 105 s, node 6 remains stationary for a minute before returning to its start position which it reaches at 210 s. This recombination is particularly demanding as the clusters are uncoordinated and may interfere with each other.

Results: Link failure. In Figure 8, we observe that Demos quickly establishes a single network consisting of all 12 nodes as in the first experiment. Once node 6 starts to move down the corridor at 60 s, its connectivity with cluster A (nodes 7–12) diminishes. As cluster B (nodes 1–6) relies on this link to communicate with cluster A, the reduced link quality results in a gradually decreasing PRR. At 88 s, we find that the link between the two clusters has completely failed and they can only communicate internally at maximally $\frac{2 \cdot 6 \cdot (6-1)}{12 \cdot (12-1)} = 45\,\%$ PRR. By directly electing a leader for cluster B, Demos matches this new optimal performance immediately after the complete split. This success is only made possible by our novel quorums, as nodes in both clusters could not win a classical majority vote that requires more than 6 nodes. The plots in Figure 8 tracing the states of the protocol illustrate that Demos maintains its data exchange without interruption in contrast to LWB, as can be seen in the nodes of both Demos clusters continuously being scheduled. Once node 6 returns to its former position, the two clusters quickly discover each other. Leveraging cluster discovery, all nodes merge simultaneously, visible by the PRR surging back to the maximum at 206 s. LWB takes much longer to activate its failover policy after the clusters split apart. After a minute during which cluster B attempts to bootstrap and only cluster A can communicate at 22 % PRR, node 3 starts

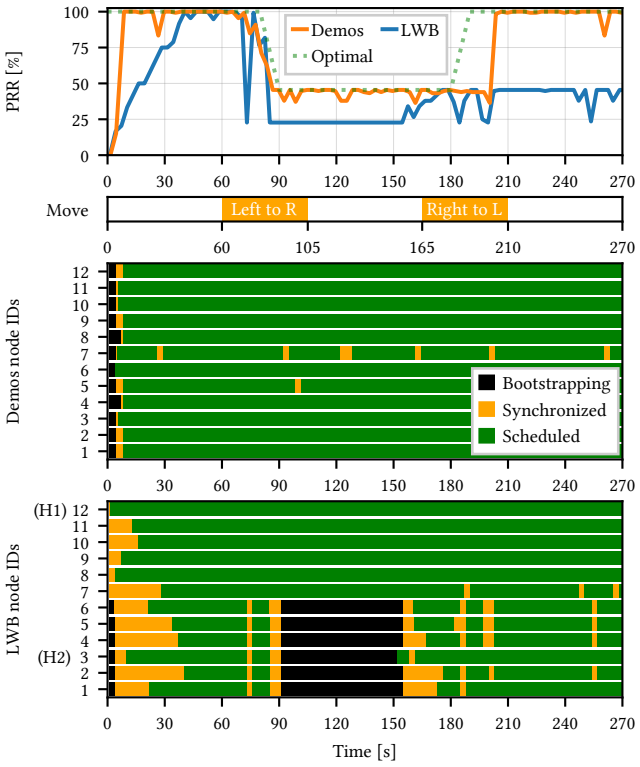

Figure 8. Moving node 6 splits the initial network topology into two clusters. Demos (middle) elects a cluster coordinator for each of them after separation and maintains optimal PRR by recombining the clusters when node 6 returns to its start position. LWB (bottom) only forms a second cluster after prolonged bootstrapping and cannot utilize the restored network topology to maximize the PRR as it lacks discovery.

as a failover host at 157 s. Subsequently, cluster B gradually ramps up its data exchange on a separate frequency channel. As LWB lacks continuous cluster discovery like all WSN protocols apart from Demos, it is unable to merge clusters even after the original network topology has been restored and remains limited to less than half of the optimal PRR.

6.4 Dynamic Cluster Coordination

Next, we compare the three quorums Q_a, Q_{a+}, and Q_r introduced by Demos in terms of election latency, stability, and energy costs. In Section 5, we determined an expected election latency of $\mathrm{E}[L] = \frac{N+1}{N_c+1}$ elections until a designated leader may propose. Based on the subsequently cast votes, the quorum then stipulates whether the election was successful. Hence, $\mathrm{E}[L]$ is a lower bound for finding a new leader.

Scenario. To examine the differences between the quorums, we use $E = 3$ elections per round and test networks of 12, 18, and 24 nodes. After 30 s without disturbance, we cause two-thirds of the network, including its coordinator, to fail. During the next 30 s, we observe when elections occur and how stably the network performs under the new conditions. We run 60 tests per combination of quorum and network size and always mutate the sequence of designated leaders.

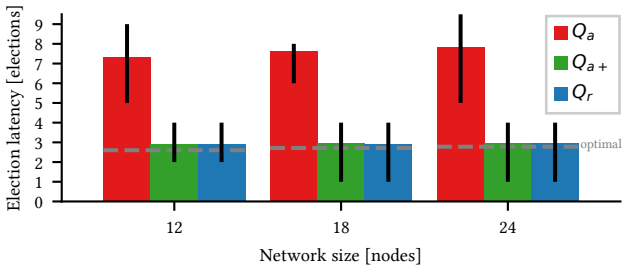

Figure 9. After a major disruption, Q_a takes multiple rounds until the estimated N_c is low enough for a successful election. Leveraging the high reliability of concurrent transmissions, Q_{a+} and Q_r match the optimal election latency (gray). Black bars show the 25^{th} and 75^{th} percentile around the mean.

Table 1. Instabilities in the first 30 s ("pre-fault") occur less often than fluctuations thereafter ("post-fault"). Q_r demonstrates excellent robustness without a single flawed election.

Quorum	Q_a			Q_{a+}			Q_r		
Network size	12	18	24	12	18	24	12	18	24
Pre-fault er.	0	0	1	0	1	0	0	0	0
Post-fault er.	13	6	4	0	1	1	0	0	0
Total errors	**24 (13.3 %)**			**3 (1.7 %)**			**0 (0.0 %)**		

Results: Latency. As the classical, fixed quorum $N/2$ [28, 39] can never be satisfied with $N_c = N/3$, we do not further consider it for this evaluation. Figure 9 depicts the average number of elections until a new leader could be established for the absolute and relative quorums. As expected, we find that Q_a requires multiple rounds until the estimated cluster size N_c is low enough for a node to gather a majority of votes. This latency slightly increases with the total network size, from 7.33 elections for 12 nodes to 7.83 for 24 nodes. On the other hand, Q_{a+} and Q_r can immediately find a new cluster coordinator once the previous leader becomes unavailable. We discover that both need only 2.90 elections on average for 12 nodes and 2.95 elections for a network of 24 nodes. These numbers further validate our formal analysis from Section 5 experimentally, based on which we expect at least 2.6 elections and 2.77 elections respectively.

Results: Stability. We encounter only a single run using Q_a (0.56 %) in which a previously elected leader is not successfully re-elected during the steady first 30 s as it could not gather enough votes to pass the quorum. However, due to the delay in estimating the new cluster size after the node failures, the first elected node cannot preserve its leadership and loses it again within a few rounds in 12.78 % of runs using Q_a. As seen in Table 1, this fluctuation predominantly affects the smallest network of 12 nodes where a single missing vote is often decisive and is quickly succeeded by stable elections thereafter once N_c has converged. Q_{a+} is much more robust, with only 1.67 % of runs encountering two simultaneous leaders that both temporarily fulfill the quorum and immediately merge. Lastly, Q_r did not experience a single instability or unexpected loss of leadership in 180 runs.

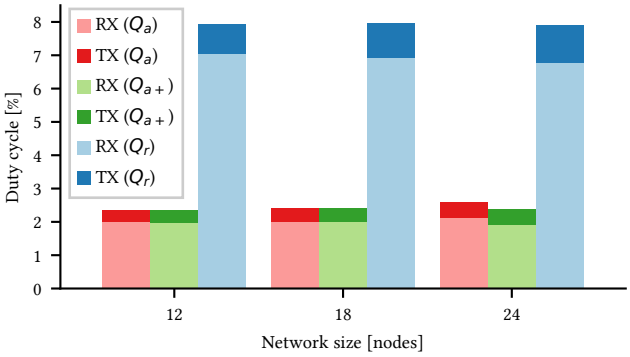

Figure 10. When using the absolute quorums Q_a and Q_{a+}, the rest of the consensus phase may be skipped after voting. In contrast, nodes employing Q_r need to participate in each election to ensure that the fraction of votes remains valid.

Results: Costs. To analyze the energy costs of the cluster coordination mechanism, we combine the reception (RX) and transmission (TX) costs of the `P` and `V` slots in Figure 10. As expected, nodes using the absolute quorums Q_a and Q_{a+} can leverage the fact that they only have to participate in a single election per round until they have cast their votes, after which they may skip the remaining two elections. In contrast, Q_r depends on the participation of all nodes, as receiving information on previously cast votes is necessary to prevent multiple leaders based on the fraction of votes in favor. This requirement results in an overhead that scales linearly with E, as seen in Q_r's costs which are three times higher compared to Q_a and Q_{a+}. Due to the excellent scalability of the all-to-all primitive, doubling the number of nodes from 12 to 24 merely increases TX costs by 25.8 %, and Demos' total overhead remains almost identical.

Verdict. We observe a trade-off between robust elections and energy efficiency. Q_a does not require assumptions on the communication reliability and reduces its energy consumption by only taking part in one election per round in most cases, but takes longer to succeed after network changes and suffers from occasional instabilities while N_c is re-estimated. On the other hand, Q_r is the most stable quorum and does not rely on an accurate estimation of the cluster size, but requires nodes to participate in each election even if they already cast their vote. We find that Q_{a+} strikes a good balance by matching Q_r in its low election latency while only necessitating participation in one election per round.

6.5 Comparing the Costs of Robustness

While we have seen in Section 6.3 that Demos excels in reacting to a node failure through its cluster coordination mechanism and facilitates cluster merges through its cluster discovery scheme, these features introduce additional costs. Therefore, we investigate the energy overhead of Demos and its components and compare it to LWB [20] as a baseline.

Scenario. We test both protocols for networks of 12, 18, and 24 nodes. We run each configuration for 15 min and skip the first 30 s to only observe steady-state behavior. Demos employs Q_{a+} as a quorum based on the verdict in Section 6.4.

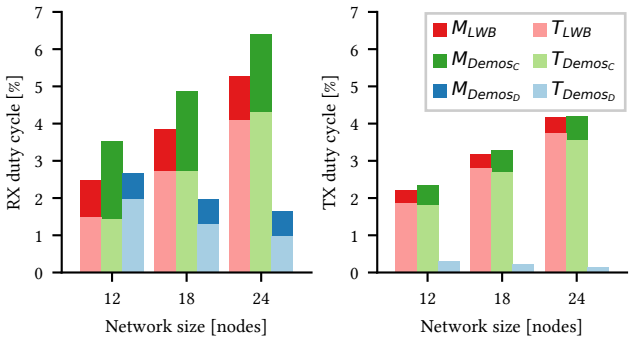

Figure 11. Comparing the management (M) and transport (T) duty cycles of LWB, the consensus and communication part of Demos ($Demos_C$), and the discovery scheme of Demos ($Demos_D$), we find that the energy overhead quickly diminishes for larger networks even for low data packet rates.

Results. In Figure 11, we separately look at the consensus and communication part of Demos ($Demos_C$) and its discovery scheme ($Demos_D$). We further differentiate between management traffic M (such as the `P`, `V`, and `S` slots for $Demos_C$ and the `N` slot for $Demos_D$) and the transport of data T (`D` slots for $Demos_C$ and the discovery phase for $Demos_D$).

As both LWB and Demos schedule equivalent traffic demands and use the same underlying one-to-all floods, the transport costs match. The management overhead of both protocols slightly rises with the network size as more information has to be shared and the average hop distance increases, prolonging listening until a packet reaches a node. However, we find that even though Demos updates the complete cluster information each round, this merely increases RX time by 92.3 % and TX time by 51.1 % on average compared to LWB. Note that the current data rate at 1 packet per round is minimal; in a real scenario, the demand to transport data is usually significantly higher [22]. As the discovery slots are distributed across nodes using Demos' concept of cluster discovery, even the current maximal rate of continuously transmitting ($P_t = 0.5$) or listening ($P_l = 0.5$) is dominated by consensus and communication. If slower discovery is permitted or if discovery can be temporally discontinued in a static scenario, these costs diminish even further.

7 Related Work

Robust networking. While Demos offers an exceptional degree of resilience, many protocols can also adjust to changing conditions to varying extents. LWB [20] centrally schedules a series of Glossy floods [19], but adapts slowly and suffers from collapsing data exchange and network splits when the coordinator is unavailable. Chaos [29] and Mixer [23] only require a coordinator for synchronization but restrict nodes to sharing a fixed amount of information. Hybrid [46] and Harmony [33] leverage a combination of Glossy and Chaos slots to improve robustness. However, Hybrid only collects data centrally at the coordinator, while Harmony can only enable fixed assigned slots and cannot cope with prolonged network splits. Instead, Demos supports the delivery of an arbitrary number of packets to any node and can merge clusters without restrictions. Furthermore, Demos dynamically elects a cluster coordinator and handles a node failure instantly without affecting the rest of the network.

To boost the reliability of floods, Dimmer [40] and OSF [7] flexibly adjust link-layer parameters depending on the current conditions and SmarTiSCH [54] alters the timing and frequencies of transmissions to mitigate interference. However, these protocols handle orthogonal issues to Demos and do not address a failure of the coordinator or the discovery of other clusters to maximize data exchange.

Coordination in WSNs. Consensus protocols such as OTR [8], A^2 [3], and WirelessPaxos [39] agree on a value such as a leader ID without leveraging it to bootstrap further communication. Other protocols such as STARC [43] require an explicit handover from a previous leader, which is infeasible after leader failures. Therefore, protocols based on concurrent transmissions either hard-code the coordinator [23, 24, 29] or exclusively elect during bootstrapping [2, 20]. Timeout-based detection of leader absence is prone to failure and hence is usually configured to multiple minutes [20], and thereafter takes dozens of seconds to elect a substitute [2]. At the other extreme, STeC [11] elects an ad-hoc leader each round but prerequires external synchronization, and BUTLER [34] constantly synchronizes but cannot coordinate a decentralized network. Classical leader election algorithms for ad-hoc networks are also inapplicable, as they require the availability of most nodes of a known network [6], guaranteed message delivery [45], at least dozens of network floods [5], or unrealistic restrictions on message propagation [15]. Demos' cluster coordination avoids timeouts through continuous re-elections, handles failures with minimal delay, and supports arbitrary network splits.

Neighbor discovery. Both randomized [36, 49] and deterministic [18, 42, 47] discovery protocols focus primarily on pairwise discovery [14]. As packet collisions increase with network size and impede discovery [26], self-adapting cycles [12] and the probabilistic skipping of transmissions [21] preserve a low discovery latency even for dense networks. Demos solves this issue by integrating nodes into clusters and dividing transmissions across its members to mitigate collisions. Group-based discovery shares discovery schedules to assist neighboring nodes [13, 52] or deliberately de-synchronizes them to boost the group's discovery success [37]. Sharing information on discovered nodes [16] and a symbiosis of group management and discovery [41] are concepts that are also found in Demos. Demos extends these concepts so that entire clusters of nodes can efficiently run cluster discovery while continuing to exchange data, and adds synchronized merging in combination with cluster coordination to consistently maximize network connectivity.

8 Conclusion

This paper introduces Demos, a highly robust protocol to orchestrate autonomous clusters in low-power WSNs. By combining a novel cluster coordination mechanism based on constant elections with a continuous cluster discovery scheme that leverages cluster information to reduce discovery latency and energy costs, Demos supports highly mobile

networks and persistently maximizes connectivity between all nodes. We introduce three new quorums to determine a cluster coordinator and show that this mechanism effortlessly handles node failures and network splits. By instantly merging clusters, we demonstrate that Demos achieves the optimal PRR and gracefully reacts to changes in the network topology. With its high robustness, Demos serves the vision of dynamic and mobile clusters that operate autonomously and automatically recover from any node or link failure.

9 Acknowledgments

This research was supported by the Swiss National Science Foundation under NCCR Automation (grant agreement `51NF40_180545`) and by the German Research Foundation (grants `ZI 1635/1-1` and `ZI 1635/2-1`).

10 References

[1] M. Afanasov et al. Battery-Less Zero-Maintenance Embedded Sensing at the Mithræum of Circus Maximus. In *SenSys*. ACM, 2020.

[2] B. Al Nahas et al. Network Bootstrapping and Leader Election Utilizing the Capture Effect in Low-Power Wireless Networks. In *SenSys*. ACM, 2017.

[3] B. Al Nahas et al. Network-Wide Consensus Utilizing the Capture Effect in Low-Power Wireless Networks. In *SenSys*. ACM, 2017.

[4] E. Aras et al. A Low-Power Hardware Platform for Smart Environment as a Call for More Flexibility and Re-Usability. In *EWSN*. Junction Publishing, 2019.

[5] J. Augustine et al. Robust Leader Election in a Fast-Changing World. In *FOMC*. Open Publishing Association, 2013.

[6] J. Augustine et al. Leader Election in Sparse Dynamic Networks with Churn. In *IPDPS*. IEEE, 2015.

[7] M. Baddeley et al. OSF: An Open-Source Framework for Synchronous Flooding over Multiple Physical Layers. In *EWSN*. ACM, 2022.

[8] A. Benchi et al. Solving Consensus in Opportunistic Networks. In *ICDCN*. ACM, 2015.

[9] A. Biri, R. Da Forno, and T. Kuonen. Demos: Source code. gitlab.ethz.ch/tec/public/demos, 2023. Accessed: 2023-06-16.

[10] A. Biri et al. SociTrack: Infrastructure-Free Interaction Tracking through Mobile Sensor Networks. In *MobiCom*. ACM, 2020.

[11] A. Biri et al. STeC: Exploiting Spatial and Temporal Correlation for Event-Based Communication in WSNs. In *SenSys*. ACM, 2021.

[12] H. Cai et al. Self-Adapting Quorum-Based Neighbor Discovery in Wireless Sensor Networks. In *INFOCOM*. IEEE, 2018.

[13] L. Chen et al. Group-Based Neighbor Discovery in Low-Duty-Cycle Mobile Sensor Networks. *IEEE Trans. Mob. Comput.*, 15(8), 2016.

[14] L. Chen et al. Neighbor discovery in mobile sensing applications: A comprehensive survey. *Ad Hoc Networks*, 48, 2016.

[15] H. Chung et al. Optimal Regional Consecutive Leader Election in Mobile Ad-Hoc Networks. In *FOMC*. ACM, 2011.

[16] R. Cohen et al. Continuous Neighbor Discovery in Asynchronous Sensor Networks. *IEEE/ACM Trans. Networking*, 19(1), 2011.

[17] D. Dujovne et al. 6TiSCH: deterministic IP-enabled industrial internet (of things). *IEEE Commun. Mag.*, 52(12), 2014.

[18] P. Dutta et al. Practical Asynchronous Neighbor Discovery and Rendezvous for Mobile Sensing Applications. In *SenSys*. ACM, 2008.

[19] F. Ferrari et al. Efficient network flooding and time synchronization with Glossy. In *IPSN*. ACM, 2011.

[20] F. Ferrari et al. Low-Power Wireless Bus. In *SenSys*. ACM, 2012.

[21] Z. Gu et al. A Practical Neighbor Discovery Framework for Wireless Sensor Networks. *Sensors*, 19(8), 2019.

[22] S. Hayat et al. Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint. *IEEE Commun. Surv. Tutorials*, 18(4), 2016.

[23] C. Herrmann et al. Mixer: Efficient Many-to-All Broadcast in Dynamic Wireless Mesh Networks. In *SenSys*. ACM, 2018.

[24] T. Istomin et al. Interference-Resilient Ultra-Low Power Aperiodic Data Collection. In *IPSN*. IEEE, 2018.

[25] T. Istomin et al. Janus: Dual-Radio Accurate and Energy-Efficient Proximity Detection. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 5(4), 2021.

[26] J. Khan et al. On the Energy Efficiency and Performance of Neighbor Discovery Schemes for Low Duty Cycle IoT Devices. In *PE-WASUN*. ACM, 2017.

[27] D. Lakew et al. Routing in Flying Ad Hoc Networks: A Comprehensive Survey. *IEEE Commun. Surv. Tutorials*, 22(2), 2020.

[28] L. Lamport. Paxos Made Simple. *ACM SIGACT News*, 32(4), 2001.

[29] O. Landsiedel et al. Chaos: Versatile and Efficient All-to-All Data Sharing and in-Network Processing at Scale. In *SenSys*. ACM, 2013.

[30] R. Lim et al. Competition: Robust Flooding using Back-to-Back Synchronous Transmissions with Channel-Hopping. In *EWSN*. Junction Publishing, 2017.

[31] M. Lopez et al. Towards Secure Wireless Mesh Networks for UAV Swarm Connectivity: Current Threats, Research, and Opportunities. In *DCOSS*. IEEE, 2021.

[32] X. Ma et al. DeCoT: A Dependable Concurrent Transmission-Based Protocol for Wireless Sensor Networks. *IEEE Access*, 6, 2018.

[33] X. Ma et al. Harmony: Saving Concurrent Transmissions from Harsh RF Interference. In *INFOCOM*. IEEE, 2020.

[34] F. Mager et al. BUTLER: Increasing the Availability of Low-Power Wireless Communication Protocols. In *EWSN*. ACM, 2022.

[35] F. Mager et al. Scaling beyond Bandwidth Limitations: Wireless Control with Stability Guarantees under Overload. *ACM Trans. Cyber-Phys. Syst.*, 6(3), 2022.

[36] M. McGlynn et al. Birthday Protocols for Low Energy Deployment and Flexible Neighbor Discovery in Ad Hoc Wireless Networks. In *MobiHoc*. ACM, 2001.

[37] R. Morillo et al. More the Merrier: Neighbor Discovery on Duty-Cycled Mobile Devices in Group Settings. *IEEE Trans. Wireless Commun.*, 21(7), 2022.

[38] D. Ongaro et al. In search of an understandable consensus algorithm. In *USENIX ATC*. USENIX Association, 2014.

[39] V. Poirot et al. Paxos Made Wireless: Consensus in the Air. In *EWSN*. Junction Publishing, 2019.

[40] V. Poirot et al. Dimmer: Self-Adaptive Network-Wide Flooding with Reinforcement Learning. In *ICDCS*. IEEE, 2021.

[41] A. Purohit et al. WiFlock: Collaborative group discovery and maintenance in mobile sensor networks. In *IPSN*. IEEE, 2011.

[42] Y. Qiu et al. Talk more listen less: Energy-efficient neighbor discovery in wireless sensor networks. In *INFOCOM*. IEEE, 2016.

[43] P. Rathje et al. STARC: Low-power Decentralized Coordination Primitive for Vehicular Ad-hoc Networks. In *NOMS*. IEEE, 2020.

[44] P. Rathje et al. TraceBand: Privacy-Preserving Contact Tracing on Low-Power Wristbands. In *EWSN*. ACM, 2022.

[45] B. Sidik et al. PALE: Time Bounded Practical Agile Leader Election. *IEEE Trans. Parallel Distrib. Syst.*, 31(2), 2020.

[46] A. Spina et al. XPC: Fast and Reliable Synchronous Transmission Protocols for 2-Phase Commit and 3-Phase Commit. In *EWSN*. Junction Publishing, 2020.

[47] W. Sun et al. Hello: A generic flexible protocol for neighbor discovery. In *INFOCOM*. IEEE, 2014.

[48] R. Trüb et al. FlockLab 2: Multi-Modal Testing and Validation for Wireless IoT. In *CPS-IoTBench*. OpenReview.net, 2020.

[49] S. Vasudevan et al. Efficient Algorithms for Neighbor Discovery in Wireless Networks. *IEEE/ACM Trans. Networking*, 21(1), 2013.

[50] S. Waharte et al. Coordinated Search with a Swarm of UAVs. In *SAHCNW*. IEEE, 2009.

[51] Y. Wang et al. Multiple vehicle Bayesian-based domain search with intermittent information sharing. In *ACC*. IEEE, 2011.

[52] L. Wei et al. A Fast Neighbor Discovery Algorithm in WSNs. *Sensors*, 18(10), 2018.

[53] D. Winkler et al. Plug-and-Play Irrigation Control at Scale. In *IPSN*. ACM, 2018.

[54] Z. Yu et al. SmarTiSCH: An interference-aware engine for IEEE 802.15. 4e-based networks. In *IPSN*. IEEE, 2022.

[55] M. Zimmerling et al. Synchronous Transmissions in Low-Power Wireless: A Survey of Communication Protocols and Network Services. *ACM Comput. Surv.*, 53(6), 2020.