Parameterized Verification of Track Topology Aggregation Protocols

Sergio Feo-Arenis, Bernd Westphal

Parameterized Verification of Track Topology Aggregation Protocols

Sergio Feo-Arenis and Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Abstract. We present an approach for the verification aggregation protocols, which may be used to perform critical tasks and thus should be verified. We formalize the class of track topology aggregation protocols and provide a parameterized proof of correctness where the problem is reduced to checking a property of the node's aggregation algorithm. We provide a verification rule based on our property and illustrate the approach by verifying a non-trivial aggregation protocol.

1 Introduction

We study the verification of *aggregation protocols*, which are often used in sensor networks [7,12,13]. Such protocols compute an aggregation function in a distributed fashion. An aggregation function is a function of the data at sensor nodes where the messages sent by a node are determined by messages received from other nodes and the own data. Data is propagated from sensor nodes towards specially designated sink nodes, where the result of the computation is made available.

A simple example of an aggregation protocol is one where the maximum value among the sensors' data is calculated at a sink node. Here, if intermediate nodes transmit only the maximum value among the received data as it is forwarded to the sink, the utilization of resources can be improved. The reduced amount of data packets can, e.g., reduce energy usage, reduce the use of available bandwidth and increase the maximum data collection rate.

Sensor networks may use unreliable communication channels. In order to mitigate the problem of unreliable communications, sensor networks may take advantage of the fact that broadcast transmissions may be received by multiple nodes. In this way, they effectively provide multiple *aggregation paths* over which a node's data can be collected.

Some aggregation functions are, however, *duplicate-sensitive*. That is, obtaining a correct result requires that sensor values are aggregated only once. This is the case when, e.g., calculating the sum or the average of the sensor values.

Data aggregation protocols are correct in the presence of unreliable communications if, whenever an aggregation path is available from a node to the sink, the sensor's data is correctly aggregated at the sink. I.e., it is aggregated, but exactly once if the function being calculated is duplicate-sensitive. This concept of correctness is the strongest possible when unreliable communications are employed. The failure of all aggregation paths cannot be completely ruled out.

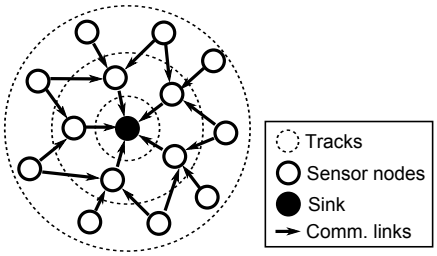

Fig. 1. Track Topology

Our goal is to provide a method to (semi-)automatically verify correctness of aggregation protocols given the implementation of the algorithm executed by network nodes. As a first approach, we concentrate on *track topologies*, where nodes are arranged in *tracks* according to their distance (in hops) to the sink, and aggregation paths traverse consecutive track boundaries as shown in Figure 1.

We study aggregation protocols for topologies that have a reliable *correction infrastructure*. I.e., nodes within tracks have reliable communication links that can be used to correct communication errors. This is the case for, e.g., the use of short-range directional antennas or aggregation systems where communication units have multiple transceivers.

We additionally assume a *schedule*, i.e., the sequence in which nodes perform their computations and transmit their data, such that the data from input nodes is always available at the time of executing a node's algorithm.

Even under those assumptions, verifying correctness of an aggregation protocol requires considering all possible topologies, with a possibly unbounded number of nodes. Aggregation protocols are parameterized systems and can thus not be verified by explicitly verifying all instances. The problem of parameterized verification is undecidable in general [1].

We approach the problem of verifying a specific class of aggregation protocols in this work. We make the following contributions.

1. We identify the class of aggregation protocols for track topologies with reliable correction infrastructure and provide a formalization as a parameterized verification problem, including the presence of link failures.
2. We provide a proof rule that reduces the parameterized verification problem to checking a verification condition on the nodes' algorithm and show its soundness. We thus provide an inductive invariant for the verification of track topology aggregation protocols.
3. We illustrate the utility of our results by applying the verification rule to a non-trivial aggregation protocol.

This work illustrates the possibility of performing parameterized verification for distributed systems with topologies that are not regular and with dynamic aspects such as node or link failures. We are able to concentrate the effort of verification on single nodes, without the need to consider topology or scheduling

issues. Our formalization could be used together with approaches such as [14] to obtain parametric proofs for such systems. Formalizing link and node reliability can provide the basis for deriving inductive invariants.

2 Track Topologies and Aggregation Protocols

We define the class of systems that are the subject of our verification effort. Inspired by instances seen in wireless sensor networks [7], we develop the concept of track topologies by establishing the properties that characterize them.

Definition 1. *A* track topology

$$T = (N, \longrightarrow, \rightsquigarrow, V)$$

with domain $\mathcal{D}$ is a finite, node-labeled, acyclic graph with nodes $N = \{n_1, \ldots, n_k\}$, edges $\longrightarrow \cup \rightsquigarrow \subseteq N \times N$, and labeling function

$$V : N \to \mathcal{D}$$

which satisfies the following conditions:

- *The set of nodes is partitioned into $d \in \mathbb{N}_0$ non-empty sets $N_0, \ldots, N_d$ called* tracks. *We call N_0 the* sink track *and use $track(n)$ to denote the number of the track to which node $n \in N$ belongs.*
  The number d is called the depth *of the topology, denoted by $depth(T)$.*
- *The relation $\longrightarrow$ is called* aggregation relation *and connects nodes in a track with nodes in the track immediately before, i.e.,*

$$\forall\, (n, n') \in \longrightarrow \bullet\ track(n) = track(n') + 1 \tag{1}$$

  For $(n, n') \in \longrightarrow$, n' is called aggregator *for n. A path from a node n to a node $n' \in N_0$ is called* aggregation path.
- *The relation $\longrightarrow$ consists of two disjoint relations, the* primary aggregation relation *PA and the* secondary aggregation relation *SA, such that (N, PA) constitutes a forest and (N, SA) is a directed, acyclic graph.*
  For $(n, n') \in PA$ (SA), n' is called primary parent *(*secondary parent*) of n. We introduce the abbreviation $SA(n) = \{n' \mid (n', n) \in SA\}$ to denote the set of* secondary children *of n.*
- *The relation $\rightsquigarrow$ is disjoint from $\longrightarrow$ and called* correction relation. *It connects the primary parent of a node with all of its backup parents, i.e.*

$$\begin{aligned} &\forall\, (n, n') \in PA\ \exists\, n_0, n_1, \ldots, n_m \bullet \\ &\quad n_0 = n' \wedge \forall\, 0 \leq i < m \bullet n_i \rightsquigarrow n_{i+1} \wedge SA(n) \subseteq \{n_0, \ldots, n_m\}. \end{aligned} \tag{2}$$

  Note that due to the restriction that the aggregation relation connects only adjacent tracks, $\rightsquigarrow$ always connects nodes in the same track.

k is called the size *of T, denoted by $|T|$.*

An example of a track topology with primary, secondary and correction relations can be seen in Figure 2.

For our inductive proof, we require track topologies to be closed under removal of single nodes in the sink track. By allowing to remove only nodes in the sink track without outgoing correction edges, it is ensured that no correction path between a primary aggregator and the backup aggregators of some node is broken, thus preserving the property (2).

Lemma 1. *Let $n_0 \in N_0$ be a node from the sink track of a track topology $T = (N_0, \longrightarrow_0, \rightsquigarrow_0, V_0)$ with domain $\mathcal{D}$ which does not have any outgoing edges in the correction relation to any other node.*

Then $T \setminus \{n_0\} := (N, \longrightarrow, \rightsquigarrow, V)$ with

- $N = N_0 \setminus \{n_0\}$, $V = V_0|_N$,
- $\longrightarrow = \{(n, n') \in \longrightarrow_0 |\ n' \neq n_0\}$, $\rightsquigarrow = \{(n, n') \in \rightsquigarrow_0 |\ n' \neq n_0\}$,
- $track(n) = track_0(n) - 1$ *if* $N_{0_0} = \{n_0\}$, *and* $track(n) = track_0(n)$ *otherwise,*

is a track topology with domain $\mathcal{D}$.

Whenever aggregation is performed, nodes initiate a transmission round, each transmitting their computed messages over the communication medium. In the following, we formalize unreliable communication links. We introduce an edge labeling function for the aggregation relation of a topology that indicates whether communication is successful or not during a transmission round.

Definition 2. *A* communication function *of a track topology $T = (N, \longrightarrow, \rightsquigarrow, V)$ is an edge labeling*

$$f : (\longrightarrow) \to \mathbb{B}$$

of the aggregation relation $\longrightarrow$ of T where $f(n, n') = 1$ if and only if the communication was successful between the connected nodes n and n'. We use F_T to denote the set of communication functions of T, and $n \longrightarrow_f n'$ if and only if $f(n, n') = 1$.

The value of node n is available at the sink track *if and only if there exists a $\longrightarrow$-path from n to a node in the sink track such that the communication along the path was successful according to f. We use $V_f(n)$ to denote the value of n if the value is available at the sink track according to f and "$\circ$" otherwise, i.e. V_f is pointwise defined as*

$$V_f(n) = \begin{cases} V(n) & , \text{ if } \exists\, n_0 \in N_0 \bullet n \longrightarrow_f^* n_0, \\ \circ & , \text{ otherwise.} \end{cases} \tag{3}$$

In the following, we present the formalization of aggregation protocols and their correctness, which provides the framework necessary to develop a formal verification approach.

We observe that an aggregation protocol can be formalized as a function that maps a set of sensor values to a final aggregation value. For simplicity, and without loss of generality, we assume that the domain of the sensor values and

that of the aggregation result are the same. In practice, complex functions may utilize different domains.

We continue by defining correctness of an aggregation protocol in the presence of unreliable communication links. For that, we formalize correctness relative to the communication function f: whenever there is a working path from a node to a sink, the data of the node should arrive (in aggregated form) at the sink.

The assumption that the correction infrastructure is reliable plays an important role. If the correction links were not reliable, then one would have to define correctness not only for working aggregation paths but also for working correction paths. We restrict our analysis to a reliable correction infrastructure. Extending it to unreliable correction links should work in a similar fashion and is the subject of future work.

Definition 3. *Let $\mathcal{T}$ be a set of track topologies with domain $\mathcal{D}$ and*

$$\mathcal{P} : \mathcal{T} \times F_{\mathcal{T}} \to \mathcal{D},$$

a track topology aggregation protocol *on $\mathcal{T}$, where F_T denotes the set of communication functions of the track topologies in $\mathcal{T}$.*

Let $\mathcal{A}$ be a family of aggregation functions

$$\mathcal{A}^k : \mathcal{D}_\circ^k \to \mathcal{D},$$

over domain $\mathcal{D}$ where $k \in \mathbb{N}_0$ and $\mathcal{D}_\circ = \mathcal{D} \sqcup \{\circ\}$.

The aggregation protocol $\mathcal{P}$ is called correct *wrt. $\mathcal{A}$ if and only if, whenever the value of a node n is available to the sink track, then the data transmitted by n is aggregated exactly once by $\mathcal{P}$, i.e. if*

$$\begin{aligned} \forall\, T = (N, \longrightarrow, \rightsquigarrow, V) \in \mathcal{T}, k = |N| \;\forall\, f \in F_T \bullet \\ \mathcal{P}(T, f) = \mathcal{A}^k \left(V_f(n_1), \ldots, V_f(n_k)\right). \end{aligned} \tag{4}$$

Now we formally characterize the aggregation protocols that are the subject of this work. We formalize the distributed calculation of the aggregation function as an equation system over local aggregation functions that map incoming messages and additional parameters to outgoing messages. The most prominent requirement in our formalization is that correction requests are directly observable in the network transmissions, thus allowing reasoning about the consistency of the messages computed by the nodes.

Definition 4. *Let $\mathcal{T}$ be a set of track topologies with a domain $\mathcal{D}$ such that $(\mathcal{D}_\circ, \oplus, \circ)$ is a monoid. A track topology aggregation protocol $\mathcal{P}$ on $\mathcal{T}$ is called* distributed homogeneous *if and only if*

- *there exists a family of sets Msg_k of* messages *together with two families*

$$\mathscr{A}^k : Msg_k^k \to \mathcal{D}$$
$$\mathscr{E}^k : Msg_k^k \to 2^{ID_k}$$

of decoder functions, $k \in \mathbb{N}_0$, $ID_k = \{1, \ldots, k\}$.
Decoder function $\mathscr{A}^k$ *obtains the aggregation value encoded in a message and* $\mathscr{E}^k : Msg_k^k \to 2^{ID_k}$ *extracts from a message the* correction requests *contained in that message, i.e., the set of nodes encoded in the message for which a corrective action should be taken if possible,*

– *there exists a family of* local aggregation functions

$$aggr^k : ID_k \times \mathcal{D} \times (ID_k \nrightarrow Msg_k) \to Msg_k, k \in \mathbb{N}_0,$$

such that, for track topology $T = (N, \longrightarrow, \rightsquigarrow, V) \in \mathcal{T}$ *and communication function* $f \in F_T$*, the aggregation result of the protocol* $\mathcal{P}$ *is the* $\oplus$*-sum of the messages emitted by the sink track, i.e.*

$$\mathcal{P}(T, f) = \bigoplus_{n_i \in N_0} \mathscr{A}(\tau_i) \tag{5}$$

where message τ_i *of node* $n_i \in N$ *in the track topology is defined by applying the local aggregation function to the identity* i *of the node, its value, and the set of messages relevant for* n_i *which have successfully been communicated according to* f*.*
That is, the messages $\tau_1, \ldots, \tau_k$ *are the solution of the equation system*

$$\begin{aligned}
\tau_1 &= aggr^k(1, Vn_1, M_1) & M_1 &= \{i \mapsto \tau_i \mid (n_i, n_1) \in (\longrightarrow_f \cup \rightsquigarrow)\} \\
\tau_2 &= aggr^k(2, Vn_2, M_2) & M_2 &= \{i \mapsto \tau_i \mid (n_i, n_2) \in (\longrightarrow_f \cup \rightsquigarrow)\} \\
&\vdots & &\vdots \\
\tau_k &= aggr^k(k, Vn_k, M_k) & M_k &= \{i \mapsto \tau_i \mid (n_i, n_k) \in (\longrightarrow_f \cup \rightsquigarrow)\}.
\end{aligned}$$

Note 1. A distributed homogeneous track topology aggregation protocol $\mathcal{P}$ with domain $\mathcal{D}$ is correct for track topologies with a single node in the sink track if and only if

$$\mathscr{A}(\tau) = \mathcal{A}^k\left(V_f(n_1), \ldots, V_f(n_k)\right) \tag{6}$$

where τ is the message emitted by the sink node.

Also note that we implicitly assume a computation sequence (or *schedule*) that ensures data availability for all nodes. That is, whenever a node starts its computations, all the input data is available. Thus, we assume that all nodes connected to a node n are scheduled to transmit their data before n.

3 Verification of Track Topology Aggregation Protocols

To be able to provide a parameterized proof of correctness as defined in the previous section, we require the aggregation function to be an associative operator over the aggregation domain, so that there is a correspondence between the final aggregated values and the partial values transmitted by intermediate nodes. We also require the operator to provide a neutral element to be able to aggregate sensors whose communication fails or do not have data available in a uniform fashion.

Definition 5. *A family of aggregation functions $\mathcal{A}^k : \mathcal{D}_\circ^k \to \mathcal{D}$, $k \in \mathbb{N}_0$, over domain $\mathcal{D}$ is called* monoidal *with respect to $(\mathcal{D}_\circ, \oplus, \circ)$, if and only if $(\mathcal{D}_\circ, \oplus, \circ)$ is a monoid and*

$$\forall\, v_1, \ldots, v_k \in \mathcal{D}_\circ \bullet \mathcal{A}^k(v_1, \ldots, v_k) = v_1 \oplus \cdots \oplus v_k. \tag{7}$$

Bringing together the use of correction links and the transmission of correction requests, we introduce the concept of responsibility among nodes, defined as the confluence of both the availability of another node's data and of a correction request for that node.

Definition 6. *Let $T = (N, \longrightarrow, \rightsquigarrow, V)$ be a track topology and $f \in F_T$ a communication function.*

A node $n' \in N$ is called responsible *for node $n \in N$, denoted by $n \longrightarrow_g n'$, if and only if n' is an aggregator of n, communication from n to n' was successful according to f, and communication from n to the predecessors of n' in the correction relation was not successful according to f, i.e. if*

$$n \longrightarrow_g n' :\Longleftrightarrow n \longrightarrow_f n' \wedge \forall\, (n'', n) \in \rightsquigarrow^+ \bullet f(n, n'') = 0 \tag{8}$$

We say a node n has a working aggregation path *to node $n' \in N$ if and only if there is a sequence of responsible nodes between them, i.e., if $n \longrightarrow_g^* n''$.*

Note that working aggregation paths are unique (if existent). Protocol correctness can then be reformulated as the property that every node correctly aggregates the data of those nodes with a working aggregation path to it.

Lemma 2. *Let $\mathcal{A}$ be a monoidal aggregation function family with respect to $(\mathcal{D}_\circ, \oplus, \circ)$ and $\mathcal{P}$ a distributed homogeneous track topology aggregation protocol on track topologies $\mathcal{T}$ with domain $\mathcal{D}$.*

If the message of a node $n \in N$ encodes the $\oplus$-aggregation of the values of all nodes n' for which n is responsible, i.e. if

$$\forall\, T \in \mathcal{T}, f \in F_T\ \forall\, i \in ID_k \bullet \mathscr{A}(\tau_i) = \bigoplus_{n \longrightarrow_g^* n_i} V(n), \tag{9}$$

then $\mathcal{P}$ is correct wrt. $\mathcal{A}$.

Proof. Let $T \in \mathcal{T}$ be a topology of size k and $f \in F_T$ a communication function.

Let $n \in N$ be a node with a working aggregation path to a node $n' \in N_0$ in the sink layer, i.e. if $n \longrightarrow_g^* n'$, By Definition 6, there is a path from n to n' with successful communications, i.e. $n \longrightarrow_f^* n'$. Thus, by Definition 2, $V_f(n) = V(n)$.

Let $n \in N$ be a node with no working aggregation path to any node $n' \in N_0$ in the sink layer. Because such a path consists of primary and backup parents which are, by Definition 1, connected by the correction relation $\rightsquigarrow$. Thus there would exist a unique *first* parent for each track, and therefore n would have a working aggregation path, thus the value of n is not available at the sink track. Hence, by Definition 2, $V_f(n) = \circ$.

By Definition 4, the premise (9), and the premise that $\mathcal{A}$ is monoidal we obtain

$$\mathcal{P}(T,f) = \bigoplus_{n_i \in N_0} \bigoplus_{n \longrightarrow_g^* n_i} V(n) = \bigoplus_{n \in N} V_f(n) = \mathcal{A}^k(V_f(n_1), \ldots, V_f(n_k)) \qquad (10)$$

thus $\mathcal{P}$ is correct. □

Now that we stated a property local to every node in a network, we introduce an abbreviation in the style of Hoare logic to denote the satisfaction of a postcondition given some precondition. We can in this form state the verification condition that must be satisfied in order to show protocol correctness. We choose this notation given that we intend to allow the verification of the implementation of the local aggregation functions against this property using software model checking.

Definition 7. *Let $\mathcal{P}$ be a distributed homogeneous track topology aggregation protocol and let P^k, Q^k be families of formulae over id, v, M_{id}, and τ_{id}.*

We write

$$\{P\}\ aggr\ \{Q\}$$

if and only if for for each $k \in \mathbb{N}_0$ and each valuation of id, v, M_{id} which satisfies P^k, that valuation extended by $\tau_{id} = aggr^k(id, v, M_{id})$ satisfies Q^k.

Having provided all the prerequisites, we can now state the main theorem of this work, where we reduce protocol correctness to the verification of a local node property. Assuming that all nodes in the network execute the same implementation of the local aggregation function *aggr*, we claim that if the local computation at a node ensures consistent aggregation up to that node, including the transmission of consistent correction requests, then the protocol is correct. We prove the soundness of this claim inductively.

Theorem 1. *Let $\mathcal{P}$ be a distributed homogeneous track topology aggregation protocol for the set $\mathcal{T}$ of track topologies with domain $\mathcal{D}$.*

If $\{P\}\ aggr\ \{Q\}$ with

$$P^k \equiv id \in ID_k \wedge v = V(n_{id})$$

and

$$\begin{aligned} Q^k \equiv \mathscr{A}(\tau_{id}) = V(n_{id}) \oplus & \bigoplus_{\substack{\tau_i \in M_{id} \\ \wedge((n_i,n_{id}) \in PA \vee (i \in \mathscr{E}(M_{id}) \wedge (n_i,n_{id}) \in SA))}} \mathscr{A}(\tau_i) \\ \wedge\, \mathscr{E}(\tau_{id}) = & \{i \mid \tau_i \notin M_{id} \wedge (n_i, n_{id}) \in PA\} \\ & \cup \{i \mid i \in \mathscr{E}(M_{id}) \wedge (\tau_i \notin M_{id} \vee (n_i, n_{id}) \notin \longrightarrow)\} \end{aligned}$$

then $\mathcal{P}$ is correct.

Proof (sketch). We show by double induction on the depth d of topologies and the size ℓ of their sink track that the set of messages emitted by the nodes according to the protocol satisfy the following properties:

- (■): For each node n, the emitted message comprises the aggregation of all nodes which n is (transitively) responsible for.
- (⧫): Correction for node n_j is initially requested by a node n_i if and only if n_i is the primary parent and there was no successful communication between the two.
- (★): A correction request for node n_j emitted by n_i implies that no predecessor of n_i in the correction relation was able to provide a correction.

Formally, we show

$$\begin{aligned}
&\forall\, d \in \mathbb{N}_0, \ell \in \mathbb{N}_0\ \forall\, T = (N, \longrightarrow, \rightsquigarrow, V) \in \mathcal{T} \bullet \\
&\quad \mathit{depth}(T) = d \wedge |N_0| = \ell \implies \\
&\qquad \forall\, n_i \in N \bullet \mathscr{A}(\tau_i) = \bigoplus_{n \longrightarrow_g^* n_i} V(n) && (■) \\
&\qquad\quad \wedge \forall\, j \bullet j \in \mathscr{E}(\tau_i) \setminus \mathscr{E}(M_i) \iff (n_j, n_i) \in PA \wedge f(n_j, n_i) = 0 && (⧫) \\
&\qquad\quad \wedge \forall\, j \in \mathscr{E}(\tau_i) \implies \forall\, (n', n_i) \in \rightsquigarrow^* \bullet\, n_j \longrightarrow n' \implies f(n_j, n') = 0. && (★)
\end{aligned}$$

Then (■) in particular implies (9), thus $\mathcal{P}$ is correct by Lemma 2. □

Discussion The theorem presented allows us to eliminate most sources of infiniteness from the parameterized correctness proof for the studied protocols. Taking advantage of the fact that nodes have only local knowledge of the state of the system and the topology, we are able to implicitly profit from the resulting local symmetry of irregular networks.

The resulting verification problem is one where the system parameter k remains as a variable, and can thus be treated symbolically by verification tools. Alternatively, even a bounded correctness proof (up to some maximum size) can be provided by explicit model checking, provided that the data types used in the implementation are finite.

Developers of aggregation protocols can profit from the inductive proof provided. They can concentrate on verifying their particular implementations against our verification condition; a process that is less involved than providing a complete proof.

Some implicit assumptions were taken into account. We considered a single aggregation round, where nodes transmit only once. We thus assume that nodes are *memoryless* between aggregation rounds. This has, nonetheless, an advantage: systems are allowed to dynamically change their topology between aggregation rounds.

4 Case Study

We analyze the ridesharing protocol [9,10], proposed for use in DARPA's satellite cluster system F6 [3]. Nodes are arranged in tracks and reliability in the

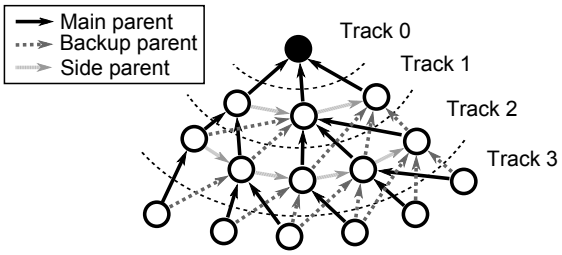

Fig. 2. Ridesharing topology

presence of failing communication links is improved by coordinating corrective actions between nodes. We concentrate on a variant of the ridesharing protocol with reliable correction links. An initial attempt of the verification of the ridesharing protocol without considering the aggregation function, but the participation vector is presented in [8], for self-containment, we recall the protocol description here.

The goal of the protocol is to aggregate the readings of sensor arranged in a track graph towards a single, specially designated node called *sink*. Except from the sink, every node has one assigned parent (the *main* parent), which is responsible for aggregating its data. Additionally, other nodes in communication range are designated as fail-over aggregators (the *backup* parents) in case the communication between a node and its main parent fails.

The possible data paths from sensors to the sink form a directed, acyclic graph (DAG) with multiple paths through the track graph (See Fig. 2). Primary and backup edges are between adjacent tracks. There are also side edges within the same track. A side edge points to a node from its *side parent*. The main edges form a spanning tree with the sink as root.

Main parents transmit correction requests if the communication to their children fails. Those requests are relayed by the track nodes towards the backup parents connected to them using the reliable side links. If a backup parent receives a correction request for a backup child, and the data is available, it aggregates the data and stops propagating the error correction request.

We show the aggregation algorithm as proposed in [10] (see Algorithm 1). Each node in a given ridesharing topology has a unique identity. Input id gives the identity of the node and PC, BC, SP are the sets of primary children, backup children, and side parents of the node respectively. Input v gives the current sensor reading and rcv the messages received by id. The algorithm computes the message to be sent by id, given v and rcv.

Nodes transmit triples $\langle A, P, E\rangle$ with the accumulated sensor value A and two control boolean vectors of length $|N|$. The *participation vector* P indicates for each node whether its value is included in A and the *error vector* E indicates at each position, whether correction is required for the node at that position.

Aggregation starts by initializing A with zero and P and E with all zeroes. If node id has sensor data, it is aggregated to A and P is updated accordingly.

Algorithm 1: The *ridesharing* aggregation algorithm. [10]

```
input: id, PC, BC, SP, v, rcv
A := 0; P := 0̄; E := 0̄;
if v ≠ NULL then { A := A + v; P[id] := 1 } ;          // Aggregate local sensor reading
E := rcv[SP];
foreach c ∈ PC ∪ BC do
    if rcv[c] ≠ undefined then
        if c ∈ PA ∨ (c ∈ SA ∧ E[c] = 1) then                  // Aggregate received values
            (A_c, P_c) := rcv[c]; A := A + A_c; P := P | P_c; E[c] := 0;
        end
    else if c ∈ PC then                                        // Request error correction
        E[c] := 1;
    end
end
return (A, P, E);
```

We use $rcv[SP]$ to denote the bit-wise disjunction of the error vectors received from id's side parents. Then, E comprises all requests for corrections. In the loop, the received messages from id's children are processed: if id received the message from c and if c is a primary child or a backup child with a pending request for correction in E then c's data is aggregated, i.e. A is updated and the P vector becomes the disjunction of the incoming P vectors. If id did not receive the message from primary child c, it flags a request for the correction of c and leaves A and P unchanged.

4.1 Verification of the Ridesharing Protocol

We show how the ridesharing protocol fits into the framework presented in Section 2. First, we show that the topologies used are track topologies.

- The set of nodes $N = \{n_1, \ldots, n_k\}$ is the set of nodes present in a ridesharing topology, there is a single sink node in the sink track $N_0 = \{n_k\}$,
- The main parent relation is the primary aggregation relation PA. It connects nodes between adjacent tracks and forms a tree with n_k as root.
- The backup parent relation is the secondary aggregation relation SA. It forms an acyclic directed graph (N, SA).
- The side parent relation is the correction relation $\rightsquigarrow$. It connects nodes sequentially.
- The *track* function can be defined with respect to PA since it forms a spanning tree:

$$track(n) = \begin{cases} 0 & \text{iff } \nexists\, (n, n') \in PA \\ track(n') + 1 & \text{iff } \exists\, (n, n') \in PA \end{cases}$$

- each node n_i has access to its sensor value v_i from the domain $\mathcal{D}$ of bounded integers, if the sensor does not have a value, $v_i = 0$ is used. Thus the labeling function is defined as: $V = \{n_i \mapsto v_i \mid 1 \leq i \leq k\}$.

Second, we show that the protocol is distributed homogeneous.

- The domain is the type of the sensor readings, we assume it to be bounded integer numbers: $\mathcal{D} = \{-2^n, \ldots, 2^n - 1\}$ for some positive integer n.

- The aggregation operator is the sum operator. I.e., $\oplus = +$.
- The domain, aggregation operator and neutral element $(\mathcal{D}, +, 0)$ form a monoid. Thus the aggregation function $\mathcal{A}^k(v_1, \ldots, v_k) = v_1 + \cdots + v_k$ is monoidal.
- The set of messages is the set of transmissions of the nodes $\tau_i = \langle A_i, P_i, E_i \rangle$ where A_i, P_i, E_i are the values calculated by *ridesharing* at node i. Thus, $Msg_k = \mathcal{D} \times \mathbb{B}^k \times \mathbb{B}^k$.
- The value decoder function reads the aggregation value directly from a message, i.e., $\mathscr{A}(\tau_i) = A_i$. The correction requests can also be read directly from the transmitted messages: $\mathscr{E}(\tau_i) = \{j \in ID_k \mid E_i[j] = 1\}$.
- The local aggregation function is Algorithm 1. I.e.,

  $$aggr^k(i, v, M) = ridesharing(i, PA(i), SA(i), SP(i), v, M)$$

  where $PA(i) = \{j \in ID_k \mid (j, i) \in PA\}$. $SA(i)$ and $SP(i)$ are defined analogously w.r.t. SA and $\rightsquigarrow$. M is defined with respect to a communication function as in Def. 4.

We can thus apply the verification theorem to the ridesharing protocol. It is left to verify whether the formula

$$\{P\}\, ridesharing(i, PA(i), SA(i), \rightsquigarrow(i), v, M)\, \{Q\}$$

is valid for P and Q as defined in Theorem 1.

To that end, we developed a model including our formalization from Sec. 2 using Boogie [2]. The topology and communication functions are represented by axioms. Additional axioms represent the properties of the system analyzed. The aggregation algorithm is input directly in imperative form by representing boolean arrays as functions of type $array : N \to \mathbb{B}$. Instead of bounded integers, we used unbounded integers to simplify the model, thus introducing the assumption that the actual data types used in an implementation are enough to accommodate the aggregation values without overflowing. Simple invariants for the loop in the algorithm were necessary: framing conditions for the loop variables and the fact that consistency is preserved across iterations of the loop was sufficient. P and Q were expressed in terms of the protocol variables and used as pre- and postconditions of the algorithm respectively[1].

The verification took 1.05 seconds to verify 35 partial verification conditions (including sanity checks), while using approximately 13 MB of memory. Boogie was able to show that the verifications conditions are all valid. We thus have shown correctness of the ridesharing protocol with reliable side links.

Discussion. Due to the symbolic representation of the parametric set of nodes, the theorem prover used by Boogie requires only a finite number of cases to show the validity of the generated verification conditions fully automatically.

recall all assumptions

[1] Model available at `http://www.informatik.uni-freiburg.de/~arenis/forte13/`

Thanks to the result that the ridesharing protocol is correct, in the sense of Definition 3, a reliability measure can be simply computed from the failure probability of the inter-track radio links as follows:

Let $p(n_i, n_j)$ be the probability of node n_j successfully receiving the transmission of a connected node n_i. The probability of the data of some node n being successfully aggregated is then equal to the probability of all links in each possible aggregation path failing:

$$1 - \prod_{\pi \in paths(n)} \left(1 - \prod_{1 \leq j < |\pi|} p\left(\pi[j], \pi[j+1]\right) \right)$$

where $paths(n)$ are the sequences $\pi = n_1, \ldots, n_m$ of nodes starting at node n, connected by the aggregation relation and ending in the sink node. The length $|\pi| = m$. The expression $\pi[i]$ refers to the i-th node on the path π.

If all links have the same success probability p, then the probability of a node n in track t successfully being aggregated at the sink simplifies to

$$1 - \left(1 - p^{track(n)}\right)^{|paths(n)|}$$

assuming that link failures are independent.

These results thus reduce the need to use simulation tools to determine the overall reliability of the ridesharing protocol.

5 Related Work

The parameterized analysis of safety properties is undecidable in the general case [1]. The problem of verifying properties for parameterized systems has been approached by several authors. We discuss a few that we feel are the closest to our line of work.

We can best situate our work in the line of work of Namjoshi et al. [14] for the verification of irregular networks. Although we concentrate on asynchronous systems, the verification condition presented in this work can be viewed as a particular case of an inductive, compositional invariant, though used in the context of irregular networks with dynamic links. We ensure the local symmetry of aggregation networks with redundancy by explicitly considering three types of edges and restricting the analysis to systems where inter-node communication occurs after computation, without interleaving. We enrich the computation model by explicitly including the network dynamic as part of the system state.

The particular problem of network dynamic in the form of link and node failures has been the subject of recent efforts [5]. We present an instance of verification of broadcast networks with link unreliability, which can simulate node failures, for the special case of aggregation protocols.

The method of "network grammars" [4], where regular families of networks are described using context-free grammars is used to verify parameterized systems by using abstraction. Several model checking problems are derived, which,

if solved, deliver a proof of correctness. We do not explicitly perform any kind of abstraction on the states of the local aggregation programs, leaving that possibility open for the method used to establish the satisfaction of our verification condition.

Similar methods rely on finding so called network invariants [11,15], overapproximations of the sets of reachable states of the systems, and proving that they imply the property being verified. The focus lies on automatically finding those invariants through model checking. We distance ourselves from the automatic finding of invariants and perform explicit induction using a fixed invariant, delivering a proof rule for the class of aggregation protocols with irregular topology. Explicit induction has been also used for the verification of regular networks [6].

The systems we consider in this work are asynchronous systems where computation steps of network components are considered atomic. The systems could be, however, considered as an instance of bounded-data parameterized systems in the sense of [15].

This work is a follow-up of the initial analysis of the protocol from our case study [8]. We improve the analysis by extending it to the use of arbitrary duplicate-sensitive aggregation functions, eliminating the need for the explicit transmission of a vector of aggregated nodes. Here, we provide a complete, formal inductive proof and, as a result, simplify the resulting verification conditions. The software model checking step of the verification is thus simplified, improving the chances of obtaining a verification problem within the capabilities of current software verification tools.

6 Conclusions

We have presented an inductive, parameterized proof of correctness for track topology aggregation protocols. We reduce the problem to the verification of a single verification consistency condition on the nodes' aggregation algorithm, which can be automatically checked using state of the art software verification tools.

We identify a class of aggregation protocols that allows us to illustrate the feasibility of reducing the verification of parameterized, asynchronous systems with irregular topologies and dynamic link (and thus node) reliability to establishing local consistency properties.

Possible directions for future work are the relaxation of the constraints for the systems verified –to enlarge the class of systems covered–, such as the reliability of the correction infrastructure and the acyclicity of some of the node connection relations.

Additionally, the use of our verification conditions as a template for the automatic generation of inductive node invariants can be explored, such that given decoding functions for a protocol messages, the verification task is further automated.

Finally, criteria for the decidability of the resulting verification problem can be studied. This would lead to the identification of a decidable class of parameterized systems.

References

1. Apt, K.R., Kozen, D.: Limits for automatic verification of finite-state concurrent systems. Inf. Process. Lett. 22(6), 307–309 (1986)
2. Barnett, M., Chang, B.Y.E., DeLine, R., Jacobs, B., Leino, K.R.M.: Boogie: A modular reusable verifier for object-oriented programs. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.P. (eds.) FMCO. LNCS, vol. 4111, pp. 364–387. Springer (2005)
3. Brown, O., Eremenko, P.: The value proposition for Fractionated space architectures. In: AIAA Space 2006. No. 7506, AIAA (2006)
4. Clarke, E.M., Grumberg, O., Jha, S.: Verifying parameterized networks. ACM Trans. Program. Lang. Syst. 19(5), 726–750 (1997)
5. Delzanno, G., Sangnier, A., Zavattaro, G.: Verification of ad hoc networks with node and communication failures. In: Giese, H., Rosu, G. (eds.) FMOODS/FORTE. LNCS, vol. 7273, pp. 235–250. Springer (2012)
6. Emerson, E.A., Namjoshi, K.S.: On reasoning about rings. Int. J. Found. Comput. Sci. 14(4), 527–550 (2003)
7. Feng, J., Eager, D.L., Makaroff, D.J.: Aggregation protocols for high rate, low delay data collection in sensor networks. In: Fratta, L., Schulzrinne, H., Takahashi, Y., Spaniol, O. (eds.) Networking. LNCS, vol. 5550, pp. 26–39. Springer (2009)
8. Feo-Arenis, S., Westphal, B.: Formal verification of a parameterized data aggregation protocol. In: Brat, G., Rungta, N., Venet, A. (eds.) NASA Formal Methods. LNCS, vol. 7871, pp. 428–434. Springer (2013)
9. Gobriel, S., Khattab, S., Mossé, D., Brustoloni, J., Melhem, R.: Ridesharing: Fault tolerant aggregation in sensor networks using corrective actions. In: IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks. pp. 595–604 (2006)
10. Iskander, M.K., Lee, A.J., et al.: Privacy and robustness for data aggregation in wireless sensor networks. In: 17th ACM conference on Computer and communications security. pp. 699–701. ACM (2010)
11. Kesten, Y., Pnueli, A., Shahar, E., Zuck, L.D.: Network invariants in action. In: Brim, L., Jancar, P., Kretínský, M., Kucera, A. (eds.) CONCUR. LNCS, vol. 2421, pp. 101–115. Springer (2002)
12. Liu, M., Gong, H.G., Mao, Y.C., Chen, L.J., Xie, L.: A distributed energy-efficient data gathering and aggregation protocol for wireless sensor networks. Journal of Software 16(12), 2106–2116 (2005)
13. Montresor, A., Jelasity, M., Babaoglu, O.: Robust aggregation protocols for large-scale overlay networks. In: Dependable Systems and Networks, 2004 International Conference on. pp. 19–28. IEEE (2004)
14. Namjoshi, K.S., Trefler, R.J.: Uncovering symmetries in irregular process networks. In: Giacobazzi, R., Berdine, J., Mastroeni, I. (eds.) VMCAI. LNCS, vol. 7737, pp. 496–514. Springer (2013)
15. Pnueli, A., Ruah, S., Zuck, L.D.: Automatic deductive verification with invisible invariants. In: Margaria, T., Yi, W. (eds.) TACAS. LNCS, vol. 2031, pp. 82–97. Springer (2001)