Formal Verification of a Parameterized Data Aggregation Protocol

Sergio Feo-Arenis and Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Abstract. We report on our experiences on the successful verification of a parameterized wireless fault-tolerant data aggregation protocol. We outline our verification method that involves automatic verification of a model of the node processing algorithm under system topology constraints. The presented work forms the basis for a generalization to verification rules for aggregation protocols that integrate automatic verification into an inductive framework.

1 Introduction

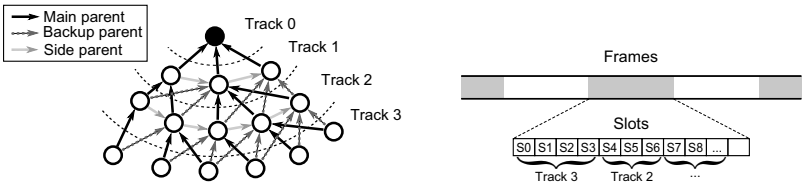
Data aggregation protocols are used in distributed systems to collect sensor data gathered by nodes of the system at dedicated *sink* nodes [5]. In case of unreliable wireless communication, a common correctness property of a data aggregation protocol is that, whenever there is a functioning communication path from a sensor node to its sink, then the data must be aggregated at the sink. One may, e.g., exploit the redundancy of radio communication, where more than one node may hear the transmissions of others, to provide multiple communication paths from a sensor to its sink. So-called *duplicate sensitive* data aggregation protocols have an additional correctness property which usually states that a sensor value should not be aggregated more than once at a sink node.

We consider the case of *parameterized* data aggregation protocols with a single sink and an arbitrary number of homogeneous nodes in a fixed *(network) topology*. For this case, we want to determine whether the correctness properties stated above are true of every configuration of the system by a semi-automatic, compositional approach. In general, this Parameterized Model Checking Problem is undecidable [1].

In this work, we report on the successful verification of the ridesharing protocol [6], that was proposed for use in DARPA's satellite cluster system F6 [4]. We applied a compositional approach that involves reasoning performed manually to derive verification conditions on the program running in the nodes. We were able to check those verification conditions fully automatically. We intend to generalize our experience from the ridesharing protocol into a general proof rule for a well-defined class of data aggregation systems which in particular comprises the ridesharing protocol.

Initially, we present an axiomatization of the system topology, the aggregation paths, and communication failures. Based on the axiomatization, we formalize

G. Brat, N. Rungta, and A. Venet (Eds.): NFM 2013, LNCS 7871, pp. 428–434, 2013.



(a) Network topology (b) Scheduling

Fig. 1. Ridesharing Protocol

the correctness property of the protocol. We outline a compositional method to prove correctness that integrates the automatic verification of the program running in the nodes. We report on the automatic verification of a Boogie [2] model that integrates the axiomatization and the program, and on how we validated our axiomatization using an interactive theorem prover.

2 The Ridesharing Topology

The ridesharing protocol [6] was proposed for use in DARPA's satellite cluster system F6 [4] where satellites communicate over unreliable radio links. It is supposed to aggregate data from nodes that are logically organized in a tree structure by a *main parent* relation (cf. Figure 1(a)). Each node has a unique main parent and, in order to provide redundancy, a set of *backup parents* on the same depth level (called *track*) as the main parent. Nodes on the same track may have a *side links* as target of requests for correction if a message from a child was lost. Communication within one track is assumed to be reliable. Data is aggregated cyclically using *schedule* to avoid message collisions. Time is split into *frames* which is further partitioned into slots (cf. Figure 1(b)). Each node is assigned exactly one slot to send data.[1] Furthermore, nodes can be assumed to be *memoryless* wrt. frames, that is, they are initialized at the beginning of each frame. Therefore it is w.l.o.g. sufficient to consider a single frame in the correctness proof.

Ridesharing Network. Formally, a *ridesharing network* is a labeled graph (N, E, V) comprising a finite set of nodes N including the designated node n_0, called the *sink*, and a set of edges $E \subseteq N \times N$ which is the union of the three pairwise relations E_m, E_b, and E_s that represent the main and backup parents, and the side links, respectively. In a network, the main parent relation induces a tree with the sink as root, i.e. (N, E_m) is a tree, and the side link relation E_s is acyclic. The labeling function $V : N \to \mathcal{D}$ assigns each node the sensor reading in the considered frame, i.e. a value from the domain of the possible sensor values $\mathcal{D}$. Additionally, there is an *aggregation function* $(\cdot \oplus \cdot)$ such that $(\mathcal{D}, \oplus)$ form a *monoid*. When no sensor data is available, the neutral element of $\oplus$ is assumed.

[1] For simplicity, we assume that the side link relation is acyclic. In general, the ridesharing protocol [6] admits that nodes are assigned multiple slots under certain side conditions.

Node n' is called *main parent* (*backup parent*, *side link*) of n, denoted by $\longrightarrow_{E_m}$ ($\longrightarrow_{E_b}$, $\longrightarrow_{E_s}$) if and only if $(n, n') \in E_m$ (E_b, E_s). We use $E_p = E_m \cup E_b$ to denote the *parent* relation and say that nodes n and n' are *directly connected*, denoted by $n \longrightarrow_{E_p} n'$, if and only if n' is either main or backup parent of n. We use, e.g., $\longrightarrow^*_{E_p}$ to denote the reflexive, transitive closure of $\longrightarrow_{E_p}$. The *track* of a node n, denoted by $track(n)$, is 0 if n is the sink, and $track(n') + 1$ if there exists a parent node n' of n. We denote the set of all nodes at track k with N_k. Side parents of a node have to be of the same track as the node itself.

Unreliable Communication and Schedule. We model unreliable communication between parents and children by the *communication function* $f : E \to \mathbb{B}$. For an edge $e = (n, n') \in E$, we assume $f(e) = 1$ if and only if the communication was successful between nodes n and n'. We use $n \Longmapsto n'$ to denote that there was successful communication between connected nodes, i.e. that $n \longrightarrow_{E_p} n'$ and $f(n, n') = 1$. Its reflexive transitive closure $n \Longmapsto^* n'$ denotes that there is a *working path* between nodes n and n'. Note that working paths are in general not unique.

For the schedule we assume that the slots are assigned guaranteeing that for all nodes n, the input nodes according to the topology relations are scheduled before n.

Aggregation Paths. Two further concepts are useful to clarify the conditions that define a successful aggregation and under which correctness must be satisfied.

First, a sequence of successful transmissions $n_0 \Longmapsto n_1 \Longmapsto \ldots \Longmapsto n_k$ is called *aggregation path from* n_0 *to* n_k if and only if n_{i+1} is the first parent of n_i that successfully receives from n_i, i.e., if

$$\forall 0 \leq i < k \; \forall n \in N_{i+1} \bullet \left(n_i \Longmapsto n \wedge n \longrightarrow^*_{E_s} n_{i+1}\right) \implies n = n_{i+1}$$

We say n_0 *has an aggregation path to* n_k, denoted by $n_0 \rightsquigarrow n_k$, if there exists an aggregation path from n_0 to n_k or if $n_0 = n_k$. Note that aggregation paths are unique in a ridesharing network for a given communication function.

Second, we introduce the term *responsible node*. A parent n' is *responsible* for aggregating the data of node n (and its children) if all preceding parents (by the side link relation) of n did not receive the transmission from n.

Correctness. Formally, a ridesharing protocol $\mathcal{P}$ can be described as a function that maps a ridesharing network with nodes N and edges E and a communication function f to a set of the nodes for which values were aggregated. I.e., $\mathcal{P} : (E \to \mathbb{B}) \to \mathbb{B}^N$. A ridesharing protocol is correct if and only if "If there is a working aggregation path between a node n and the sink then n's data is aggregated exactly once by the sink." I.e.

$$\mathcal{P} \text{ is correct} :\Longleftrightarrow \forall n \in N \bullet n \rightsquigarrow n_0 \implies \mathcal{P}(f)[n] = 1 \qquad (\textit{correctness})$$

where n_0 is the sink node.

Algorithm 1. Aggregation algorithm run by network nodes.

```
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 ∈ PC ∨ (c ∈ BC ∧ 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);
```

3 A Ridesharing Protocol Algorithm

We seek an algorithm which, given a topology, realizes a correct protocol if executed on every node according to the schedule. We recall the aggregation algorithm as proposed in [6] (cf. Algorithm 1).

We assume that each node in the given topology has a unique identity. In order to abstract from communication and data gathering functionality, we assume that the algorithm is executed once per frame on each node. Input *id* gives the identity of the node and PC (BC, SP) the finite set of *primary children* (*backup children*, *side parents*) of the node, i.e. the inverse of E_m (E_b, E_s). Input v gives the current sensor reading and *rcv* the messages received by *id* in the current frame. The algorithm computes the message to be sent by *id*, given v and *rcv*.

The set of network messages M consists of triples (A, P, E) 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, the *error vector* E indicates at each position, whether correction is required for the node at that position.

Aggregation starts by initializing A with the neutral element of the aggregation function and P and E with all zeroes. If node *id* has sensor data, it is aggregated to A and P updated accordingly. 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 as follows: 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 correction leaving A and P unchanged.

Executing Algorithm 1 once for each node in a network according to the schedule yields a history. A *history* h is a sequence of transmissions $\tau_1, \tau_2, \ldots, \tau_{|N|}$ where $\tau_j = (A_j, E_j, P_j)$ is the message transmitted by the node scheduled at slot j. Given the communication function f, that indicates which node received which transmission, there is the following relation between history h and the partial functions $rcv_n[\cdot] : N \nrightarrow M$: For each two connected nodes $n \longrightarrow_{E_p} n'$, $rcv_n[n'] = \tau_i$ if n'

is scheduled at slot i and $f(n, n')$. $rcv_n[n']$ is undefined otherwise. That is, rcv_{id} is passed as parameter rcv to the execution of Algorithm 1 on node id. The execution of Algorithm 1 once for each node in a network is a ridesharing protocol $\mathcal{P}$ as it maps a communication function f to the transmission of the sink, which is scheduled last, i.e., $\mathcal{P}(f) := P_{|N|}$.

4 Compositional Verification

In the formalizations presented in Sections 2 and 3, we have a formal model of all finite instances of Ridesharing, of which there are unboundedly many. We can model ridesharing networks for all numbers n of nodes (inducing length n for the vectors P and E) and all tree topologies (including all sizes up to n of PC and BC), each with $2^{|E_p|}$ failure scenarios. In general, correctness is, for this setting, undecidable.

Nonetheless, we have successfully verified the correctness of Algorithm 1 with respect to the correctness property of Section 2. Our approach focuses the verification efforts on any single node, due to the observation that the aggregation algorithm works symmetrically with respect to the id parameter. In principle, we verify whether, when a node in any given track receives *consistent* data from the subjacent track and its side parent, the track to which the node belongs also transmits *consistent* data.

In general, having correct data for an arbitrary node at its scheduled slot is a property of the complete earlier history, which again has an unbounded length. However, in this case, we can observe that for every node, only the exact structure of the network at the track immediately below and the own track is relevant. This observation allows us to produce an abstraction that partitions the history – and thus the input data for the node – in a finite manner according to whether the received data contains information for nodes on tracks below, on the same track, or on tracks above the node in question. We thus observed, that it is sufficient to assume that the nodes on a track aggregate data exclusively from the tracks below them, that no data is aggregated in a duplicate manner, and that the side parents do not transmit spurious correction messages. Formally, the data transmitted by track $N_k = \{n_1, \ldots, n_\ell\}$ in history h is the subsequence $\tau_1, \tau_2, \ldots, \tau_\ell$ where $\tau_i = (A_i, E_i, P_i)$ is the transmission of node n_i, $1 \leq i \leq \ell$. We call the data of track N_k *consistent* if and only if

$$\begin{aligned} &\forall n_i \neq n_j \in N_k \, \forall n \in N \bullet \big(\neg(P_i[n] \wedge P_j[n]) \\ &\quad \wedge (P_i[n] \implies n = n_i \vee \mathit{track}(n) > k) \wedge (E_i[n] \implies \mathit{track}(n) = k + 1)\big) \end{aligned} \tag{1}$$

This property allows us to give a specification for the aggregation algorithm, in the form of pre- and postconditions. The precondition is that the data in the rcv buffers of a node is consistent, in the sense of (1) and with respect to a communication function f. The algorithm should guarantee the postcondition that, for every possible role of a node, the bits in each position of the output vectors are set correctly with respect to the input data. That is, that no spurious aggregation occurred for nodes on the same track or tracks above, and that correction signals were correctly processed and generated such that no duplicate aggregation will

occur. Satisfying that condition allows us to conclude that the data output by the track of the node being verified is consistent.

We utilized Boogie [2] to perform that verification task automatically. We used the axiomatization from Section 2 and added our pre- and postconditions. Due to the loop in Algorithm 1, an invariant was necessary. Framing conditions for the loop variables and the fact that consistency is preserved across iterations of the loop were sufficient.

We ensured the consistency of our model by checking that the axioms that describe the topology and the environment are consistent. We utilized a combination of smoke testing[2] and debugging by examination of counterexamples using BVD [7]. Boogie required approximately 1 second and 13MB of RAM to verify a total of 35 partial verification conditions.

To increase our confidence on the successful Boogie verification results, we checked whether our axiomatization was consistent, i.e., whether our axioms imply non-empty topologies and thus whether the verification conditions are not trivially satisfied. For that purpose, we used HOL-Boogie [3] to translate the model together with its verification conditions into Isabelle [9] and reconstructed the proof. [3]. Only one manual lemma was necessary due to technicalities in the translation to Isabelle. The remaining proof was reconstructed automatically.

Having verified that tracks produce consistent output is sufficient to reason inductively and establish that for each topology of depth d, the messages of track 0 will be a correct aggregation of the nodes in the tracks below with respect to the communication function f. In our particular case, track 0 contains only the sink node.

Our induction proof is a double induction. Vertically, we consider ridesharing topologies of depth d and horizontally the width of tracks. The base case $d = 0$ is a track consisting of only leaf nodes. In the induction step, we inductively prove that for a depth $d + 1$ the chains of side links inside the track, starting at the leftmost node, preserve our consistency property where we can assume consistent data from track d.

5 Conclusions and Future Work

Applications of sensor networks for critical tasks commonly require robust data aggregation protocols. They represent an interesting instance of parameterized systems.

We presented the successful verification of the ridesharing protocol, a wireless aggregation protocol that employs redundant aggregation paths. The verification puts a "spotlight" [8] on a single node in a single track, while giving a finite abstraction of the data coming from the subjacent track. This allowed us to derive proof obligations on the aggregation algorithm which we discharged using automatic software model checking. We checked our axiomatization using interactive theorem proving. Overall we obtain a compositional approach which decouples

[2] In Boogie, adding `assert(false)` to each basic block to check for its reachability.
[3] Code available at: `http://www.informatik.uni-freiburg.de/~arenis/nfm13/`

the verification of the aggregation algorithm from the communication scheme. Using automatic software model checking increases the degree of automation and allows for an easy extension to a heterogeneous implementation: each implementation just needs to be verified to satisfy the proof obligations. To the extent of our knowledge, there are no previous works on the combination of deduction and automatic model checking for the verification of aggregation protocols.

In the future, we would like to generalize our approach. This amounts to a more general axiomatization of network topologies, e.g. lifting the restrictions on acyclicity and reliability of the side links, a deductive framework based on our inductive approach, and a simplification of the invariants required. Having narrowed down the conditions that are sufficient to ensure a finite case-split during verification, our generalization would then identify another 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 2005. LNCS, vol. 4111, pp. 364–387. Springer, Heidelberg (2006)
3. Böhme, S., Moskal, M., Schulte, W., Wolff, B.: HOL-Boogie - an interactive prover-backend for the verifying C compiler. J. Autom. Reasoning 44(1-2), 111–144 (2010)
4. Brown, O., Eremenko, P.: The value proposition for Fractionated space architectures. In: AIAA Space 2006, No. 7506. AIAA (2006)
5. Feng, J., Eager, D.L., Makaroff, D.: Aggregation protocols for high rate, low delay data collection in sensor networks. In: Fratta, L., Schulzrinne, H., Takahashi, Y., Spaniol, O. (eds.) NETWORKING 2009. LNCS, vol. 5550, pp. 26–39. Springer, Heidelberg (2009)
6. 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)
7. Le Goues, C., Leino, K.R.M., Moskal, M.: The boogie verification debugger (tool paper). In: Barthe, G., Pardo, A., Schneider, G. (eds.) SEFM 2011. LNCS, vol. 7041, pp. 407–414. Springer, Heidelberg (2011)
8. Wachter, B., Westphal, B.: The spotlight principle. On combining process-summarising state abstractions. In: Cook, B., Podelski, A. (eds.) VMCAI 2007. LNCS, vol. 4349, pp. 182–198. Springer, Heidelberg (2007)
9. Wenzel, M., Paulson, L.C., Nipkow, T.: The isabelle framework. In: Mohamed, O.A., Muñoz, C., Tahar, S. (eds.) TPHOLs 2008. LNCS, vol. 5170, pp. 33–38. Springer, Heidelberg (2008)