Security and Robustness for Collaborative Monitors

Bas Testerink[1(✉)], Nils Bulling[2], and Mehdi Dastani[1]

[1] Utrecht University, Utrecht, Netherlands
{B.J.G.Testerink,M.M.Dastani}@uu.nl
[2] Delft University of Technology, Delft, Netherlands
n.bulling@tudelft.nl

Abstract. Decentralized monitors can be subject to robustness and security risks. Robustness risks include attacks on the monitor's infrastructure in order to disable parts of its functionality. Security risks include attacks that try to extract information from the monitor and thereby possibly leak sensitive information. Formal methods to analyze the design of a monitor with respect to these issues can help to create more secure designs and/or identify critical parts. In this paper we specify a model for analyzing robustness and security risks for collaborative monitors constructed from a network of local monitors.

Keywords: Monitoring · Runtime verification · Security

1 Introduction

Normative systems help to make sure that agents behave according to preset guidelines/norms in multi-agent systems [5]. One approach is provided by exogenous normative systems where norms are explicit. The normative aspect of the multi-agent system is captured by an exogenous—to the agents—organization or institution. With this approach it must be verified whether any norm violation occurs in the multi-agent system's execution. Monitoring large distributed multi-agent systems such as traffic, smart grids and economic markets requires decentralized approaches. Monolithic centralized monitors can impose a bottleneck due to the distributed nature of multi-agent systems and a single point of failure in case of break downs.

A major concern of many decentralized verification applications is their robustness and security. The data that is gathered from a multi-agent system can severely compromise the agents' privacy if leaked. Adversaries can also try to take down parts of the network to impede its functioning. Formal models of decentralized monitors allow for the analysis of critical parts in monitors in terms of robustness and security. Such an analysis allows the developers of decentralized monitors to invest more resources in critical parts. In this paper we present a formal model for decentralized monitors that supports their formal analysis to face the aspects of robustness and security when designing a monitor. As an

V. Dignum et al. (Eds.): COIN 2015, LNAI 9628, pp. 376–395, 2016.
DOI: 10.1007/978-3-319-42691-4_21

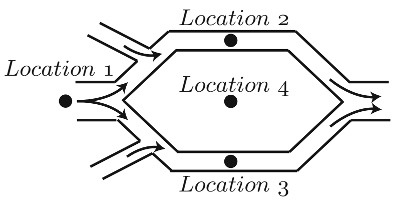

Fig. 1. Example scenario. Black dots indicate locations, arrows indicate traffic flow and double lines indicate roads.

example we shall use an abstract traffic monitoring scenario (Example 1). Traffic monitoring faces many challenges, including physical attacks on the monitor infrastructure and the privacy of individuals (cf. [7]).

In our approach we assume that monitors observe the execution trace of a system in order to detect specific properties of its behavior. These properties are expressed in linear-time temporal logic (LTL) [14]. Temporal logics have been used in the past to analyze normative systems (e.g. [1,2]). For the structure of decentralized monitors we draw inspiration from existing popular decentralized monitoring techniques such as wireless sensor networks (WSNs). The structure of a WSN is that a collection of information gathering nodes route sensor data towards a sink, which acts as the central data gathering point. Intermediate data aggregation is often used to increase security and save energy. In our decentralized framework a network of local monitors collaborates to verify properties. We call such a network a collaborative monitor. Each local monitor is assumed to make observations on its own and can in addition to that query other local monitors with respect to their (aggregated) observations. This allows to enhance privacy as monitors only obtain an aggregated value. This is similar to the frameworks proposed in [3,17]. However, different from those two frameworks we assume that information flows through the network without temporal delay. Also, instead of sharing observations as in [17] or progressed formulas as in [3], local monitors in our framework combine their input into a single evaluation (true, false, or 'yet unknown') and share that with other neighboring local monitors.

The contribution of this paper is a formal framework for specifying collaborative monitors. The model allows to analyze how critical specific local monitors are with respect to the security and robustness of the collaborative monitor. We believe that the presented design methodology is not only beneficial for the design and development of decentralized monitors using LTL, but can also provide insights into design-time and/or runtime analysis of robustness and security risks for other decentralized verification technologies. We leave a study of technical properties for future work.

Example 1 (Smart Infrastructure Scenario). Throughout this paper we give examples using a simple smart infrastructure scenario. We assume that there are various traffic streams that at some point merge together, as shown in Fig. 1. The aim of the smart infrastructure is to maximize throughput by minimizing traffic jams at locations 2 and 3. To this end there is a road side unit (RSU) at location 1 which informs passing vehicles whether location 2 or location 3 is jammed. Which, if this is the case, will hopefully cause vehicles to choose the non-jammed route. A local monitor at location 1 monitors whether vehicles that pass location 1 will end up in a traffic jam at either location 2 or 3. Each location has a local monitor with-short range communication capabilities that observes the vehicles which pass by, except for the monitor at location 4 which observes nothing, but can relay long distance messages. The monitors at locations 2 and 3 can, in addition to observing individual vehicles, also determine whether their location is jammed. The local monitor at location 1 should be able to verify whether a vehicle that passes location 1 does not end up in a traffic jam.

The rest of the paper is structured as follows. In Sect. 2 we summarize related work and background literature that influenced this paper. In Sect. 3 we present the model for collaborative monitors that we use to describe robustness in Sect. 4 and security in Sect. 5. In Sect. 6 we discuss possible changes in the framework's assumptions and future work.

2 Related Work and Background

The field of wireless sensor networks (WSNs) contains a vast amount of identified robustness and security risks as well as countermeasures (cf. [11,12]). Example risks include the malfunctioning of hardware and software and attempts by an adversary to eavesdrop on communication. Countermeasures include various techniques such as routing protocols and encryption. The aim of countermeasures is to keep the monitoring service online if local monitors malfunction and prevent sensitive information from being obtained by an adversary. The requirements of WSNs are commonly organized by: (1) data confidentiality (only intended receivers can see sensitive data), (2) data integrity and freshness (data is correct and new), (3) protection against Sybil attacks (the imitation of monitors) and (4) availability (continued operation of monitors).

Protection against confidentiality and availability attacks will be the main focus of this paper. Data integrity and freshness is assumed. I.e., we assume that monitors either work correctly or they are unavailable, but cannot for instance send false information. Different kinds of attacks can be categorized between attacks that change the network topology (e.g. physically compromising a node or communication line, or a wormhole attack that connects two nodes) and those that extract information from the network (capture and/or imitation of nodes). We shall address the case that a local monitor can malfunction. This encapsulates both aggressive and non-aggressive failures of local monitors. We shall also discuss the case that an attacker can query a local monitor without proper authorization. This encapsulates Sybil attacks.

WSNs also suffer from hardware constraints. Sensors tend to have limited power and communication capabilities. A common practice to limit energy usage is to use intermediate aggregation of data between the source of data and the sink. Data aggregation also helps to optimize any other monitoring system by reducing the amount of required communication. There are many works dedicated to security in data aggregation systems [10]. We note that the concept of privacy in data aggregation literature is generally interpreted as privacy of sensors. For instance, if a sensor computes the average velocity of all vehicles, then it may receive from other sensors the average velocities of partitions of the set of vehicles. Privacy in related literature would mean that it is not known which sensors provided what data to compute the total average velocity. In this paper we consider only privacy in the form of the privacy of agents. For instance, in our example scenario the location of a specific vehicle can be seen as privacy sensitive information. Therefore, instead of a monitor sharing separately the velocity of the vehicle and whether location 2 is jammed, it may share the evaluation of the conjunction of these two facts. If a receiver obtains this evaluation and it is false, then the receiver cannot derive whether the vehicle was not at location 2 or whether there was a traffic jam. Note that if the evaluation is true, then it is known that the vehicle was at location 2.

Security related papers on wireless networks for monitoring tend to focus on how to prevent security risks by using cryptography (e.g. [13]) and/or special routing protocols (e.g. [8]). Our approach is complementary to this. We do not look at runtime implementation techniques for preventing risks, but address design time questions and analysis to see where potential robustness and security risks lie. We believe design based analysis can help in further improving decentralized monitors. Depending on the practical limitations of an application it might not be possible to always make a perfect design. But our work can help in determining which parts of a network require more advanced/expensive hardware to increase safety.

WSNs usually concern sensor readings of continuous parameters. However, we take a logical approach with discrete values as this fits better with the declarative nature of normative systems theory. Many normative systems express norms as conditional obligations/prohibitions with deadlines (cf. [2]). Such constructions can often be expressed as properties about a system's behavior over time. This has led us to opting for linear temporal logic. Monitors that perform runtime verification of properties that are expressed in LTL can be modeled with automata [4] or progression systems [6]. For our framework it is not important which one is used. As for decentralized LTL verification there are the proposals from [3, 17]. However the framework in [3] is built on assumptions that do not support our intended scenarios. For one, in their framework all monitors are connected to each other whereas we want to investigate specific topologies. The framework from [17] does allow different topologies but data is not aggregated by local monitors. Also we have no notion of information delay, which both frameworks have.

As in [3, 17] we assume that the monitors work synchronously. This means that at any moment all monitors are (partially) observing the same behavior of the multi-agent system. In various decentralized monitoring communication protocols synchronization is introduced in order to have data freshness (e.g. SNEP [13]). We assume that the monitors are connected in an acyclic manner and that the aggregation operation of local monitors is taking a combination of the input, as explained in the next section.

3 Monitor Model

The task of a monitor is to verify whether the monitored system's behavior satisfies a property. The behavior of the system is a trace of states, and properties are expressed as LTL formulas. We first introduce preliminaries. Then, we explain the architecture of collaborative monitors and how views of local monitors and their in-between connections give rise to their locally verifiable properties. The basic idea of our verification model has been published as an extended abstract in [16].

3.1 Formal Setting

Let Π be a set of propositional symbols. A transition system (over Π) is given by a tuple $T = (S, R, V)$ where S is a state space, $R \subseteq S^2$ a serial transition relation, and $V : S \to 2^{\Pi}$ an evaluation function which returns the propositions that hold at a given state. We assume that Π, S, V a are fixed throughout this article if not said otherwise.

An *infinite trace* is defined as an infinite sequence $\sigma = s_0 s_1 \ldots$ of states interconnected by R, i.e. $(s_i, s_{i+1}) \in R$ for all $i \in \mathbb{N}_0$. Similarly, a *finite trace* is given by $s_0 \ldots s_k$. The set of infinite and finite traces over T is denoted by Tr^i_T and Tr^f_T, respectively. The set of all traces is denoted by $\mathsf{Tr}_T = \mathsf{Tr}^i_T \cup \mathsf{Tr}^f_T$. The *length* of a trace, i.e. the number of states on it, is denoted by $|\sigma|$; in particular, if $\sigma \in \mathsf{Tr}^i_T$ then $|\sigma| = \infty$. We use $\sigma[i]$ to refer to state i on σ where $0 \leq i < |\sigma|$. We shall often take traces as first-class citizen if it is not important to highlight the transition system which generated them; in that case, we omit mentioning T as subscript and whenever it is clear from context. Given a finite trace σ we write $\sigma\mathsf{Tr}^i$ to refer to all infinite traces in Tr^i that extend σ, i.e. which have σ as initial prefix. Similarly, we assume that Tr refers to a set of traces (over T) in the remainder of this paper.

We use linear-time temporal logic LTL [14] for specifying properties. Formulas of LTL are defined by the following grammar:

$$\varphi := p \mid \neg\varphi \mid \varphi \vee \psi \mid \bigcirc\varphi \mid \varphi\mathcal{U}\psi$$

where $p \in \Pi$ is a propositional symbol. As usual, we use $\Diamond\varphi$ as macro for $\top\mathcal{U}\varphi$ (sometime in the future φ holds) and $\Box\varphi$ (always φ) for $\neg\Diamond\neg\varphi$.

Definition 1 (Infinite LTL Semantics). *Let $T = (S, R, V)$ be a transition system, $\sigma \in \mathsf{Tr}^i$ be an infinite trace and $i \in \mathbb{N}_0$ an index. The infinite trace semantics for LTL is defined by relation $\models$ as follows:*

$$
\begin{array}{ll}
T, \sigma, i \models p & \Leftrightarrow p \in V(\sigma[i]) \\
T, \sigma, i \models \neg\varphi & \Leftrightarrow T, \sigma, i \not\models \varphi \\
T, \sigma, i \models \varphi \vee \psi & \Leftrightarrow T, \sigma, i \models \varphi \textit{ or } T, \sigma, i \models \psi \\
T, \sigma, i \models \bigcirc\varphi & \Leftrightarrow T, \sigma, i+1 \models \varphi \\
T, \sigma, i \models \varphi \mathcal{U} \psi & \Leftrightarrow \exists j \in [i, \infty] : T, \sigma, j \models \psi \textit{ and} \\
& \quad \forall k \in [i, j-1] : T, \sigma, k \models \varphi
\end{array}
$$

The finite trace semantics for LTL of [4] evaluates a formula to t (true), f (false) or ? (unknown). The intuitive reading for t is that the property holds for a given finite trace, therefore also for all finite and infinite extensions of that trace. Analogously, f means that the property does not hold independent of the future behavior of the system. Finally, ? means that the current finite trace can be extended such that the property holds or does not hold; the satisfaction of the formula is still open.

Definition 2 (Finite LTL Semantics). *Let $T = (S, R, V)$ be a transition system, $\sigma \in \mathsf{Tr}^f$ be a finite trace and $j \in [0, |\sigma| - 1]$ an index. The finite trace semantics for* LTL *is defined by relation $[\cdot]^T_{\sigma,j}$ as follows:*

$$
[\varphi]^T_{\sigma,j} = \begin{cases} t \ \ \textit{if } \forall \sigma' \in \sigma\mathsf{Tr}^i : T, \sigma', j \models \varphi \\ f \ \textit{if } \forall \sigma' \in \sigma\mathsf{Tr}^i : T, \sigma', j \not\models \varphi \\ ? \ \textit{otherwise} \end{cases}
$$

We write $[\varphi]^T_\sigma$ for $[\varphi]^T_{\sigma,0}$. Moreover, we use $T, \sigma, j \models^3 \varphi$ to refer to $[\varphi]^T_{\sigma,j} = t$, and $\mathsf{Tr}^f \models^3 \varphi$ if for all finite traces $\sigma \in \mathsf{Tr}^f$ we have that $T, \sigma, 0 \models^3 \varphi$.

3.2 Local and Collaborative Monitors

Before we define local monitors, we first discuss their capability of aggregating (ternary) evaluations of formulae into a single evaluation. Ultimately, we will use this aggregation to obtain the evaluation of an LTL property on a finite trace. A monitor aggregates evaluations using propositional formula α which we evaluate using Kleene's ternary semantics [9].

Definition 3 (Aggregation Formula). *An* aggregation formula with k variables *is a propositional formula α with k propositional symbols $x_1, \ldots, x_k$ which can take on the values in $\{t, f, ?\}$. Aggregation formulae are evaluated using Kleene's ternary semantics shown in Fig. 2. Given truth values $v_1, \ldots, v_k \in \{t, f, ?\}$ we write $\alpha(v_1, \ldots v_k)$ to refer to the evaluation of α if truth value v_i is assigned to variable x_i.*

Each local monitor has its own sensing capabilities which allow the local monitor to verify an LTL formula on any finite execution trace. The basic idea of an observation formula φ is that the monitor can distinguish all infinite traces where φ holds from those where φ does not hold. This is from a specification perspective. However, we are especially interested how the monitor makes a

a	$\neg a$
f	t
$?$	$?$
t	f

$a \wedge b$		b		
		f	$?$	t
a	f	f	f	f
	$?$	f	$?$	$?$
	t	f	$?$	t

$a \vee b$		b		
		f	$?$	t
a	f	f	$?$	t
	$?$	$?$	$?$	t
	t	t	t	t

Fig. 2. Truth definition of Kleene logic.

decision at run-time. For example, if the monitor is assumed to be able to observe $\Diamond p$, it does not mean that at run-time the monitor can always decide after a finite number of steps whether the current trace is a $\Diamond p$ trace or not. In addition to the monitor's observation formula we allow it to use inputs of other monitors according to a given aggregation formula.

Definition 4 (Local Monitor). *A local monitor is a tuple $m = (\alpha, \varphi)$, where α is an aggregation formula over $k+1$ variables for some $k \geq 0$, and φ is an LTL formula, called m's* observation formula.

For any aggregation formula α with $k+1$ variables we assume that the variables are named $o, x_1, \ldots, x_k$ and are ordered. Moreover, we shall also write $\alpha(o, x_1, \ldots, x_k)$ if we want to make the variable names explicit. Note the difference to $\alpha(v_0, \ldots, v_k)$ for truth values v_i, $i = 0, \ldots, k$. The intuition of $\alpha(o, x_1, \ldots, x_k)$ is that the monitor aggregates its current observation, encoded in o, with the input evaluations $x_1, \ldots, x_k$ of its neighboring local monitors according to α. By convention we reserve the first variable for the evaluation of the monitor's observation formula.

Example 2 (Local Monitor). For simplicity we talk about a specific vehicle in our examples. Local monitors m_1, m_2 and m_3 can observe the location of the vehicle (l_i stands for "the vehicle is at location i"). m_2 and m_3 can also observe traffic jams at their location (j_i means "location i is jammed"). We assume a transition system $T = (S, R, V)$ as a model of our scenario. The state space is $S = \{s_0, ..., s_5\}$. The transition relation consists of all pairs (s_0, s_i) and (s_i, s_5) with $i \in \{1, ..., 4\}$ and (s_5, s_5). The valuation function is given by $V(s_0) = \{l_1\}, V(s_1) = \{l_2\}, V(s_2) = \{l_2, j_2\}, V(s_3) = \{l_3\}, V(s_4) = \{l_3, j_3\}$ and finally $V(s_5) = \emptyset$. Hence, for each infinite trace in Tr that starts at s_0 there is a moment where either l_2 or l_3 is visited by the vehicle, and in that state either the vehicle is in a jam or not. As an example monitor we consider the monitor $m_1 = (\alpha_1, \varphi_1)$, where $\alpha_1 = o \wedge x_1$ is a formula with two variables. For instance given that o is the valuation of φ_1 and x_1 is an input valuation then if o's valuation is true (t) and $x_1 =?$ then $\alpha(t, ?)$ is evaluated to ? by Kleene's semantics. Monitor 1 can observe whether l_1 holds for a given state. This allows m_1 to monitor a formula $\varphi_1 = \Diamond l_1$ (the vehicle is at location 1 at some moment).

A collaborative monitor is modeled by a directed acyclic graph of local monitors. The main reason for acyclicity is to avoid unnecessary complexities for the formulation of collaborative LTL verification. The connections between local monitors is referred to as the query relation. This relation is given by a function that given a local monitor m returns the connected local monitors in the order of their input for the aggregation formula of m. In the following definition we use M^k for the set of monitor tuples of length k.

Definition 5 (Collaborative Monitor Specification). *A collaborative monitor C is specified by (M, qry), where M is a non-empty set of local monitors, $\mathsf{qry} : M \to \{\epsilon\} \cup \bigcup_{k=1}^{|M|} M^k$ is a query relation with $\mathsf{qry}(m) = (m_1, \ldots, m_j)$ for $j = 1, \ldots, |M|$ iff $m = (\alpha, \varphi)$ where α is an aggregation formula over $j + 1$ variables. Moreover, qry is assumed to be acyclic*[1]*. The empty sequence is denoted by ϵ.*

In the case that $\mathsf{qry}(m) = \epsilon$ (the empty sequence) the monitor cannot query other monitors.

Example 3 (Collaborative Monitor Specification). Figure 3 shows a representation of the example collaborative monitor. Monitors m_2 and m_3 are assumed to be able to see whether the vehicle is at their location, and whether there is a jam at their location. This allows monitor m_2 to distinguish traces where $\Box(\neg j_2 \vee \neg l_2)$ either holds or not. The formula is read as "The vehicle is never at location 2 whilst there is a jam at location 2". We use this formula as the observation formula of m_2. m_3 has the same observation formula, but with respect to location 3. Both m_2 and m_3 do not receive any input. m_4 has no observation capabilities. We set its observation formula to $\top$[2]. The monitor aggregates inputs from m_2 and m_3 which will represent the statement "the vehicle has been in a traffic jam". Finally, as discussed in the previous example, m_1 can observe $\Diamond l_1$ and aggregates, using input from m_4, whether the vehicle at some point has passed through location 1 and whether the vehicle has been in a traffic jam. Formally the collaborative monitor from the example scenario is specified by (M, qry) where:

- $M = \{m_1, m_2, m_3, m_4\}$.
- $\mathsf{qry}(m_1) = (m_4)$, $\mathsf{qry}(m_2) = \mathsf{qry}(m_3) = \varepsilon$ (no input), $\mathsf{qry}(m_4) = (m_2, m_3)$.

For a local monitor, let o be the observation formula's evaluation variable, and x and y be input evaluations variables. The local monitors are given by:

- $m_1 = (\alpha_1, \varphi_1)$, $\alpha_1(o, x) = o \wedge x$, $\varphi_1 = \Diamond l_1$.
- $m_2 = (\alpha_2, \varphi_2)$, $\alpha_2(o) = o$, $\varphi_2 = \Box(\neg j_2 \vee \neg l_2)$.
- $m_3 = (\alpha_3, \varphi_3)$, $\alpha_3(o) = o$, $\varphi_3 = \Box(\neg j_3 \vee \neg l_3)$.
- $m_4 = (\alpha_4, \varphi_4)$, $\alpha_4(o, x, y) = o \wedge x \wedge y$, $\varphi_4 = \top$.

[1] Firstly, let *reachable* be inductively defined as: m' is reachable from m if m' is among $\mathsf{qry}(m) = (m_1, \ldots, m_k)$. Furthermore by transitivity if m'' is reachable by m' and m' is reachable from m, then m'' is reachable by m. Acyclicity means that there is no $m \in M$ such that m is reachable from m.

[2] We note that the choice of $\top$ for "no observation" only makes sense because of m_4's aggregation formula $o \wedge x \wedge y$ as defined below.

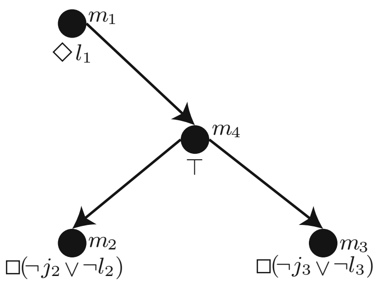
m_1
$\Diamond l_1$
m_4
$\top$
m_2
$\Box(\neg j_2 \vee \neg l_2)$
m_3
$\Box(\neg j_3 \vee \neg l_3)$

Fig. 3. Example collaborative monitor. Nodes are local monitors with at the right their name, arrows indicate the query relation, below monitors are the observation formulas.

For example, monitor m_4 is assumed to be able to distinguish all traces satisfying $\varphi = \Box(\neg j_2 \vee \neg l_2) \wedge \Box(\neg j_3 \vee \neg l_3) \equiv \Box((\neg j_2 \vee \neg l_2) \wedge (\neg j_3 \vee \neg l_3))$. This formula is evaluated to false (f) if and only if at some point the vehicle is at a jammed location. Monitor m_1 is assumed to be able to distinguish all traces satisfying $\varphi' \equiv \Diamond l_1 \wedge \varphi$. This formula is evaluated to false if and only if at some point the vehicle passes location 1 and if at some point the vehicle is at a jammed location. (Note that this follows from the way the model is constructed.)

Local monitors are models of runtime monitoring applications. In a runtime system the monitor does not have access to the infinite trace that a monitored system produces. Instead, the behavior of the system is revealed incrementally as it develops at runtime. Therefore, it is up to a monitor to determine whether some property is true, false or unknown given a finite trace. We shall therefore analyze local and collaborative monitors by using finite traces of the monitored system. Consider the local monitor $m_2 = (\alpha_2, \varphi_2)$ from our example scenario and an arbitrary trace $\sigma \in \mathsf{Tr}^f$ for an example model T of our scenario. The input that m_2 provides to m_4 given σ is given by the application of α_2 on the valuation $[\varphi_2]^T_\sigma$. This also holds for $m_3 = (\alpha_3, \varphi_3)$. The input that $m_4 = (\alpha_4, \varphi_4)$ provides to m_1 given σ is hence $\alpha_4([\varphi_4]^T_\sigma, \alpha_2([\varphi_2]^T_\sigma), \alpha_3([\varphi_3]^T_\sigma))$. The inputs of an aggregation formula are ultimately the evaluation of LTL formulas on a given trace. Hence, the input that for instance m_4 provides to m_1 given a trace σ is equivalent to the evaluation of some Boolean combination φ of the formulas φ_4, φ_2 and φ_3. We call this Boolean combination the m_4-aggregate. If a local monitor m can query m', then it means that m can query the evaluation of the m'-aggregate. It is important to note, however, that only the evaluation of the aggregate is communicated and not the truth values of its composed parts. This is an important feature of our model to ensure privacy and security properties. Because for each local monitor the aggregation formula is fixed, it means that given a collaborative monitor all the aggregates can be determined at design-time. Also note that due to acyclicity there are local monitors $m = (\alpha, \varphi)$ without neighbors and hence for such local monitors their m-aggregate is equivalent to a Boolean combination of the monitor's observation formula φ, providing four alternatives: φ, $\neg\varphi$, $\varphi \wedge \neg\varphi$ or $\varphi \vee \neg\varphi$. To see this, observe that any 3-valued function α on a single input necessarily maps ? to ? and any $x \neq ?$ to $\alpha(x) \neq ?$. Hence only four possible different functions are possible given a single variable aggregation formula.

Definition 6 (Aggregate, Agg_m). *Let T be a transition system and $C = (M, \mathsf{qry})$ be a collaborative monitor. A mapping $\mathsf{Agg} : M \rightarrow \mathsf{LTL}$ is called an M-aggregation iff for all $m = (\alpha, \varphi) \in M$ and all $\sigma \in \mathsf{Tr}^f$ we have that*

$$[\mathsf{Agg}(m)]^T_\sigma \equiv \alpha([\varphi]^T_\sigma, [\mathsf{Agg}(m_1)]^T_\sigma, \dots, [\mathsf{Agg}(m_k)]^T_\sigma).$$

where $\mathsf{qry}(m) = (m_1, \dots, m_k)$. We simply call $\mathsf{Agg}(m)$ the m-aggregate (wrt. Agg) and denote it by Agg_m.

The following proposition shows that an M-aggregation always exists and also how it can be constructed. We also note that the result makes use of the acyclicity of a collaborative monitor.

Proposition 1. *Let T be a transition system and $C = (M, \mathsf{qry})$ be a collaborative monitor. Then, an M-aggregation exists. Moreover, an M-aggregation can be effectively constructed such that for each $m = (\alpha, \varphi) \in M$ with $\mathsf{qry}(m) = (m_1, \dots, m_k)$ the m-aggregate Agg_m is given by $\mathsf{Agg}_m = \alpha[\varphi/o, \mathsf{Agg}_{m_1}/x_1, \dots, \mathsf{Agg}_{m_k}/x_k]$ where $\alpha[\psi/x]$ denotes that variable x is replaced by formula ψ in α.*

Proof (Sketch). Due to acyclicity there are local monitors $m = (\alpha, \varphi) \in M$ such that $\mathsf{qry}(m) = \epsilon$ returns no local monitors. For these local monitors α is an aggregation formula with one variable o. We can construct the m-aggregate of such a local monitor $m = (\alpha, \varphi)$ by syntactically replacing the variable o in α by φ and maintaining the Boolean connectives. Now we can proceed inductively. Let $m' = (\alpha'(o', x_1, ..., x_k), \varphi') \in M$ be a monitor where $\mathsf{qry}(m') = (m_1, \dots, m_k)$ and the aggregates or Agg_{m_i} for $i = 1, \dots, k$ are already constructed. We can construct the m'-aggregate by syntactically replacing in α' the first variable o' by φ', and each variable x_i by the m_i-aggregate for $i \in 1, ..., k$. Hence, proceeding bottom-up, we can construct the M-aggregation Agg. $\square$

Example 4 (Aggregate). The m_1-aggregate Agg_{m_1} is equal to $\Diamond l_1 \wedge \mathsf{Agg}_{m_4} = \Diamond l_1 \wedge \mathsf{Agg}_{m_2} \wedge \mathsf{Agg}_{m_3} = \Diamond l_1 \wedge \Box(\neg j_2 \vee \neg l_2) \wedge \Box(\neg j_3 \vee \neg l_3)$. This corresponds with the assumption from Example 3 that m_1 should be able to distinguish between traces where the vehicle passes through location one and a traffic jam, and those where this does not happen.

3.3 Monitorability and Expressivity

For various robustness and security issues, and from a design perspective, it is useful to determine what kind of formulas can be collaboratively verified by local monitors. We first recall the notion of (non)monitorability, which is an adaptation of the one from [4,15]. The difference is that we assume an underlying transition system. A formula φ is nonmonitorable after a finite trace σ if φ can never be evaluated to a conclusive true or false after σ given the finite LTL semantics. For φ to be monitorable it means that there is no finite trace such that φ is nonmonitorable after that trace. We highlight that monitorable does not mean that it will be possible to evaluate the truth of the formula, but only that the possibility is not excluded. Consider for example the formula $\Diamond p$ and a transition system with two states q_1 and q_2 with $R = \{(q_1, q_1), (q_1, q_2), (q_2, q_2)\}$ and only q_2 is labeled with a proposition p. Then, after each finite trace $\Diamond p$ is monitorable as each finite trace can be extended to evaluate the formula true or false. However, after each finite trace q_1^* no definite evaluation can be given.

Definition 7 ((Non)monitorable [4]**).** *Let T be a transition system, $\sigma \in \mathsf{Tr}^f$ be a finite trace, and φ be an arbitrary* LTL*-formula. We say that φ is* Tr*-nonmonitorable after σ iff for all finite traces $\sigma' \in \sigma\mathsf{Tr}^f$ it holds that $[\varphi]^T_{\sigma'} =?$. We say that φ is* Tr*-monitorable iff there is no finite trace $\sigma \in \mathsf{Tr}^f$ such that φ is* Tr*-nonmonitorable after σ.*

Note that a formula is defined to be nonmonitorable *after a specific* finite trace, and monitorable if no such trace exists. Hence nonmonitorable is a different concept than "not monitorable". For a formula to be "not monitorable" it means that there is a trace such that the formula becomes nonmonitorable after that trace.

Remark 1 (Monitorability). The definition of (non)-monitorability requires a transition system/set of traces. As a consequence, a formula can be not monitorable for one transition system, but monitorable for another. Consider for example the formula $\Box\Diamond p$. Let $T = (S, R, V)$ be a transition system with $S = \{s_0, s_1\}, R = S \times S$ and $V(s_0) = \{p\}$ and $V(s_1) = \emptyset$. For all finite traces $\sigma \in \mathsf{Tr}^f$ of T it holds that $[\Box\Diamond p]^T_\sigma =?$, hence $\Box\Diamond p$ is not Tr-monitorable. Let $T' = (\{s_0\}, \{(s_0, s_0)\}, V')$, such that $V'(s_0) = \emptyset$. The set of traces Tr' contains only traces in which p never holds. Hence, $[\Box\Diamond p]^{T'}_\sigma = f$ for each $\sigma \in \mathsf{Tr}^{f'}$, which makes the formula Tr'-monitorable.

The next proposition captures the observation that monitorability is invariant regarding equivalent formulae.

Proposition 2. *Let T be a transition system and φ and ψ* LTL*-formulae. If $\mathsf{Tr}^f \models^3 \varphi \leftrightarrow \psi$, then φ is* Tr*-monitorable iff ψ is* Tr*-monitorable.*

Proof (Sketch). Because $\mathsf{Tr}^f \models^3 \varphi \leftrightarrow \psi$ it means that $\forall\sigma \in \mathsf{Tr}^f : [\varphi]^T_\sigma = [\psi]^T_\sigma$. Hence, if φ is monitorable then for each trace $\sigma \in \mathsf{Tr}^f$ it holds that there must be a $\sigma' \in \sigma\mathsf{Tr}^f$ such that $[\varphi]^T_{\sigma'} \neq ?$ and by extension $[\psi]^T_{\sigma'} \neq ?$, hence ψ is then also monitorable. The other way around is exactly the same if we switch φ and ψ. $\Box$

We are interested in whether a local monitor m can verify a specific LTL formula φ for any trace. The m-aggregate is syntactically defined. If φ and Agg_m

give the same result on all traces of a transition system and φ is monitorable, or alternatively if Agg_m is monitorable (cf. Proposition 2) then the monitor can detect all definite evaluations of φ.

Example 5 (Monitorability). The formula $\psi = \Diamond l_1 \wedge \Box((\neg j_2 \vee \neg l_2) \wedge (\neg j_3 \vee \neg l_3))$ is equivalent to Agg_{m_1} from Example 4. As we assumed that the vehicle will pass maximally once through a location, and it will at some point go through either location 2 or 3, we have that for any trace where the vehicle has not passed location 2 or 3, there is an extension in which the vehicle passes these locations. Then, when it passes a location, there is either a traffic jam or not. After the vehicle passes location 2 or 3, all extensions of the trace will not contain another state where the vehicle passes location 2 or 3. Therefore, any trace in which either location is passed will evaluate ψ to t or f, which makes ψ Tr-monitorable.

We assume that each collaborative monitor has the purpose that one or more local monitors can observe a specific formula. We call the specification of this purpose the expressiveness constraint of the collaborative monitor which is a set of local monitor/formula pairs (m, φ) where the local monitor m must be able to observe formula φ.

Definition 8 (Expressiveness Constraint). *An expressiveness constraint for a collaborative monitor $C = (M, \mathsf{qry})$ is a relation $E \subseteq M \times \mathsf{LTL}$ consisting of pairs of local monitors and formulae. The collaborative monitor C Tr-satisfies the expressiveness constraint E iff for each $(m, \varphi) \in E$ it holds that $\mathsf{Tr} \models^3 \varphi \leftrightarrow \mathsf{Agg}_m$.*

Example 6 (Expressiveness Constraint). An expressiveness constraint for the example scenario is $E = \{(m_1, \varphi)\}$ where $\varphi = \Diamond l_1 \wedge \Box(\neg j_2 \vee \neg l_2) \wedge \Box(\neg j_3 \vee \neg l_3)$ is from Example 4. Because φ is equivalent to Agg_{m_1}, it means that the example collaborative monitor Tr-satisfies the expressiveness constraint E.

4 Robustness

Suppose we are given a collaborative monitor which satisfies some expressiveness constraint. For various reasons, such as physical sabotaging attacks, local monitors and/or communication links between them can malfunction. From a system designer's perspective it can make sense to construct a monitor with some redundancy such that the expressiveness constraint is still satisfied when some components malfunction. In this section we analyze the robustness of monitors; that is, to which degree local monitor failures affect the functioning of the collaborative monitor.

4.1 Monitor Malfunctioning

Conceptually, one may imagine that a failing local monitor is removed from the collaborative monitor. For another local monitor the malfunction may cause a situation where one of its inputs no longer exists, hence its aggregation formula has to be updated. We update the aggregation formula by removing the occurrences of the variable that corresponds to the failing monitor.

Definition 9 (Local Monitor Malfunctioning). *Let $C = (M, \mathsf{qry})$ be a collaborative monitor, and $F \subseteq M$ be a set of malfunctioning monitors. We define the collaborative monitor $C|_F = (M', \mathsf{qry}')$ as follows:*

- *$M' = M \setminus F$.*
- *for each $m \in M$ it holds that $\mathsf{qry}'(m)$ equals $\mathsf{qry}(m)$ but with each local monitors in F being removed from the tuple.*
- *M contains all $m = (\alpha, \varphi) \in M \setminus F$ where $\mathsf{qry}(m) = (m_1, \dots, m_k)$ but each $\alpha(o, x_1, ..., x_k)$ is replaced by α' which is obtained by the following procedure that is repeated until no variable x_i for $m_i \in F$ remains:*
  1. *If $l_i \in \{x_i, \neg x_i, \neg\neg x_i, \dots\}$ occurs in a disjunctive subformula $l_i \vee \varphi$ or conjunctive subformula $l_i \wedge \varphi$ of α then this subformula is rewritten to φ.*
  2. *If α equals l_i with $l_i \in \{x_i, \neg x_i, \neg\neg x_i, \dots\}$, then $\alpha' = \top$.*

Intuitively, the definition expresses that input from malfunction monitors are ignored in a sense that they can no longer help to classify traces.

Example 7 (Local Monitor Malfunctioning). If $F = \{m_4\}$ then monitor 4 is removed. The resulting collaborative monitor is $C|_F = (M', \mathsf{qry}')$ where $M' = \{m_1, m_2, m_3\}$ and $\mathsf{qry}'(m_i) = \mathsf{qry}(m_i)$ for $i = 2$ and $i = 3$, and $\mathsf{qry}'(m_1) = \varepsilon$ is the empty sequence. The malfunction causes $\alpha_1 = o \wedge x$ to be updated to $\alpha'_1 = o$, which means that Agg_{m_1} becomes equivalent to $\Diamond l_1$.

We limit ourselves to local monitor malfunctioning, but communication malfunctioning can be straightforwardly defined as well. Instead of removing local monitors, only the query relation is updated. For example, if communication from m_1 to m_4 malfunctions then the new query relation removes m_4 from m_1's input sequence of local monitors. The aggregation formula is updated following the same procedure proposed for local monitor malfunctioning. An extension of malfunctioning where monitors send false information is also interesting and left for future research.

4.2 Monitor Robustness

In a hostile environment local monitors can be damaged, but they can also malfunction for other reasons (e.g. running out of energy). We aim at quantifying robustness in terms of how much damage a collaborative monitor can take before its expressiveness constraint is not satisfied any more. This damage can be expressed as a set of potentially malfunctioning local monitors or a number specifying the number of malfunctioning local monitors. We consider monitors that do not occur in an expressiveness constraint to be supporting monitors. k-robustness is a measurement of how many of such monitors may fail before the expressiveness constraint is not satisfied.

Definition 10 (Collaborative Monitor Robustness). *Let $C = (M, \mathsf{qry})$ be a collaborative monitor* Tr*-satisfying the expressiveness constraint E, $M' = \{m \mid m \in M,\ (m, \varphi) \notin E\}$ and $F \subseteq M$ be a subset of local monitors. We say that C is* F-robust *for E and* Tr *if $C|_F$* Tr*-satisfies E. We say that C is* k-robust for E and Tr*, $k \in \mathbb{N}_0$, if C is F'-robust for E and* Tr *for any $F' \subseteq M'$ with $|F'| \leq k$.*

We note that given a collaborative monitor C, 0-robustness is equivalent to $\emptyset$-robustness for some E and Tr which simply means that C Tr-satisfies E. If every local monitor occurs in E with some φ then only 0-robustness can be obtained.

Example 8 (Collaborative Monitor Robustness). Let $F = \{m_4\}$ and E be as in Example 6. The collaborative monitor is not F-robust for E. However, assume we allow that m_2 and m_3 can potentially switch to more energy costly but long range communication. This can be modeled by including m_2 and m_3 in $\mathsf{qry}(m_1)$. Also assume that in that case Agg_{m_1} is equivalent to $\varphi_1 \wedge (\mathsf{Agg}_{m_4} \vee (\mathsf{Agg}_{m_2} \wedge (\mathsf{Agg}_{m_3})))$. In this scenario, if m_4 fails then the new m_1-aggregate becomes $\varphi_1 \wedge (\mathsf{Agg}_{m_2} \wedge \mathsf{Agg}_{m_3})$. Hence, the monitor would be F-robust. Also, given E the example collaborative monitor can only be 0-robust.

Aside from a specific attack we might wonder how much damage a monitor can take in general before it fails. This is especially useful in scenarios with many homogeneous local monitors such as botnets where attacks can be widespread and targeting any point in the network. This notion is captured by k-robustness. To determine the k of k-robustness, one has to consider the potential set of monitors which might fail and then check for each set of monitors of size k whether the collaborative monitor is robust with respect to those subsets and its expressiveness constraints. We leave a detailed investigation for future studies.

4.3 Fail Tolerance

Recall from Sect. 2 that we aimed at providing basic metrics to determine critical parts of a collaborative monitor, and that data availability is one of the topics that we address. In wireless sensor networks data availability is a concept that describes that data is available to monitor some property because enough sensors are working properly. Hence, robustness is related to data availability as it deals with scenarios where monitors fail. Intuitively we want to capture for a local monitor how critical its functioning is for the collaborative monitor. We shall use a fairly simple qualification called fail tolerance for determining how critical a local monitor is. This can be used as a basis for more sophisticated metrics. Recall that the presented analysis is for design purposes. In an implementation one has to deploy a mechanism for detecting whether a monitor has failed.

For a collaborative monitor C, a local monitor m in C, and an expressiveness constraint E, we call monitor m *k-fail tolerant* if alongside m at least $k-1$ other monitors must fail before C cannot satisfy E, and without m's failure the expressiveness constraint would be satisfied. In particular, m being 1-fail tolerant means that m's correct functioning is absolutely critical for the collaborative monitor, because alongside m zero other monitors must fail before the expressiveness constraint is not satisfied. Similarly, m being k-fail tolerant and $k > 1$ indicates that there is some redundancy wrt. m. For instance if m is 2-fail tolerant then the expressiveness constraint is not violated if only m fails. Therefore, there must be some other monitor that has some redundancy with m. Being ∞-fail tolerant would indicate that the collaborative monitor does not depend on m's functioning for satisfying its expressiveness constraints. That is, if the collaborative monitor does not satisfy an expressiveness constraint E due to any set of failing monitors, then it would still not satisfy E if m did not fail.

Definition 11 (Fail Tolerant). *Let T be a transition system, $C = (M, \mathsf{qry})$ be a collaborative monitor and E be an expressiveness constraint. For a monitor $m \in M$ we say m is k-fail tolerant wrt. E and Tr, $k < \infty$ iff there is a $F \subseteq M$ of size k such that:*

- *(1) $m \in F$, C is not F-robust for E and Tr, and for each subset $F' \subseteq F \setminus \{m\}$ C is F'-robust for E and Tr, and*
- *(2) there is no $F' \subseteq M$ such that (1) holds for F' and $|F'| < |F|$.*

If there does not exist a $k < \infty$ such that m is k-fail tolerant, then we say that m is ∞-fail tolerant.

Example 9 (Fail Tolerance). All monitors in our example scenario are 1-fail tolerant, as each of them observes vital information for the goal aggregate in m_1. However, see Fig. 4 for an illustration of more robust collaborative monitors. In the left monitor we assume only one expressiveness constraint such that m_7 must be able to aggregate $p \wedge q \wedge r$, hence m_7 is 1-fail tolerant. m_1 is 1-fail tolerant as its failure will immediately let the whole monitor fail. m_2 is 2-fail tolerant as its failure together with m_3 will let the whole monitor fail. Note that m_1 together with m_2 is not considered for m_2's fail tolerance as m_1 was already 1-fail tolerant. Monitors m_4 to m_6 are 3-fail tolerant.

In the right collaborative monitor of Fig. 4 we assume that m_5 must be able to aggregate $p \wedge q$ and hence it is 1-fail tolerant. All the other monitors are 2-fail tolerant, and the collaborative monitor as a whole is 1-robust.

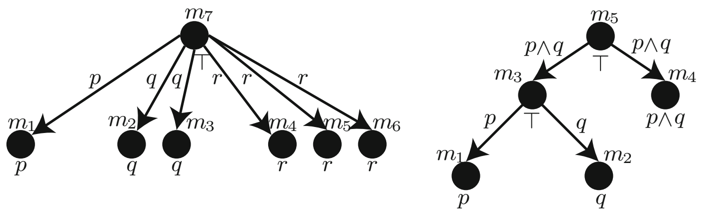

Fig. 4. Example collaborative monitors. Nodes are local monitors with on the top their name, arrows indicate the query relation, below monitors are the observation formulas. An arrow from a monitor m_i to m_j is labeled with the m_j-aggregate.

We note that if a monitor is k-robust, then the lowest i-fail tolerance that a local monitor can have aside from 1 is $(k+1)$.

Given a monitor design we can now see how critical a local monitor is in terms of its functioning. A straightforward expansion of this fail tolerance analysis is to not only check for F-robustness, but also look at which expressiveness pairs (m, φ) are not satisfied anymore in case the monitor is not F-robust. If not all expressiveness constraints are equally important, then for tolerance analysis this can be taken into account. Also looking at how many pairs are not satisfied is an important ingredient should one want to specify graceful degradation for collaborative monitors. At runtime the collaborative monitor can benefit from fail tolerance analysis by assigning higher reparation priorities to critical monitors, if possible.

5 Security

When it comes to the security of a collaborative monitor, we focus on the possible information an attacker can extract from the monitor. We say an attacker extracts an LTL property φ if the attacker can extract from the collaborative monitor the evaluation of φ given an arbitrary finite trace.

5.1 Information Extraction

In practice there are various ways in which an attacker can extract information. The attacker can intercept and interpret messages, query other monitors by pretending to be authorized and capture a monitor. To analyze intercepted messages it is required to know what messages are exactly exchanged, something which is not covered in our model. Also our model is not suited for analyzing how a monitor can be reprogrammed, given that we only have semantical representations and no program specifications of monitors. Therefore we focus on the situation where the attacker can ask a query by pretending to be authorized. This can in practice occur, for example, if the attacker pretends to be a local monitor from the network.

We assume that like monitors the attacker can aggregate extracted information by using some aggregation formula. However, for the attacker this is not a fixed formula. Hence, given the aggregates that it obtains from the collaborative monitor it can combine them by standard Boolean connectives. In the following $\mathsf{PL}(X)$ denotes all possible formulae that can obtained by applying Boolean connectives to elements in X.

Definition 12 (Monitor Attack). *Let $C = (M, \mathsf{qry})$ be a collaborative monitor. An attack $\mathsf{att} \subseteq M$ is a set of local monitors. The set of extracted properties $\mathcal{L}^C_{\mathsf{att}}$ is $\mathsf{PL}(\{\mathsf{Agg}_m | m \in \mathsf{att}\})$.*

If the framework is extended and other forms of attacks, such as eavesdropping on communication, can also be analyzed then these will contribute to the set of extracted properties. Note that if some φ is extracted from the monitor, then the attacker may still not know whether the system's behavior satisfies this property given a finite trace, because the evaluation may be inconclusive.

Example 10 (Monitor Attack). If an attacker can imitate m_4 then it will be able to query m_2 and m_3. In that case $\mathsf{att} = \{m_2, m_3\}$ and the extracted properties are $\mathsf{PL}(\{\mathsf{Agg}_{m_2}, \mathsf{Agg}_{m_3}\})$. For instance the attacker can determine given an arbitrary trace what the evaluation is of $\neg\Box(\neg j_2 \vee \neg l_2) \vee \neg\Box(\neg j_3 \vee \neg l_3)$ (somewhere in the future there was a jam at either location and/or the vehicle was at either location).

5.2 Safety

For monitor safety we look at whether a specific attack can be used to observe some given property from the collaborative monitor. We assume the attacker knows what an aggregate represents. That is, if it obtains information on the evaluation of the m-aggregate for a monitor m, then it knows that the m-aggregate is Agg_m.

Definition 13 (Monitor Safety). *Let T be transition system, C be a collaborative monitor,* att *be an attack and φ be an* LTL *formula. We say that C is* Tr*-safe for φ and* att *iff there is no $\psi \in \mathcal{L}^C_{\mathsf{att}}$ such that $\mathsf{Tr}^f \models^3 \varphi \leftrightarrow \psi$.*

Example 11 (Monitor Safety). Given $\mathsf{att} = \{m_1, m_2, m_3, m_4\}$ the example collaborative monitor is Tr-safe for $\varphi = \Diamond(l_1 \wedge \Diamond l_2)$ and att. It is also Tr-safe for $\psi = \Diamond(l_1 \wedge \Diamond l_3)$ and att. This means that even if the attacker can obtain all available aggregates, it still cannot determine for a trace whether the vehicle used or may use in the future the route through locations 1 and 2 or through locations 1 and 3, respectively.

The space of potential attacks is heavily restricted by practical details that are not covered by our model. For instance if in a network some local monitor only has wired connections to other local monitors in a safe environment, then it might be impossible that local monitor is targeted for an attack. Therefore we focus on analyzing security risks wrt. potentially attacked local monitors and with the assumption that the attack is practically feasible. The security constraint of a monitor consist of a set of monitors that can potentially be attacked and a set of properties that represent sensitive information. The analysis of what properties count as sensitive should be part of the system's design methodology. These will differ per practical real-world scenario. A monitor satisfies its security constraint if none of the considered attacks allows the attacker to monitor a sensitive property.

Definition 14 (Security Constraints). *Let T be a transition system and $C = (M, \mathsf{qry})$ be a collaborative monitor. A security constraint for C is defined as (A, P) where $A \subseteq M$ is a set of local monitors and $P \subseteq \mathsf{LTL}$ is a set of sensitive properties. C* Tr*-satisfies its security constraint iff for each and $\varphi \in P$, C is* Tr*-safe for φ and A.*

It should be noted that for some constraint (A, P) a not monitorable property $\varphi \in P$ might still be reasonable to consider as sensitive information. There is the possibility that given the behavior of the monitored system at runtime such a formula will always be evaluated to '?', but this is not guaranteed to be the case, unless φ is nonmonitorable after the empty trace.

Example 12 (Security Constraint). Let $M = \{m_1, m_2, m_3, m_4\}$, φ and ψ be as in Example 11 and $(A, P) = (M, \{\varphi, \psi\})$ be the scenario's security constraint. The example collaborative monitor Tr-satisfies its security constraint (A, P). This means that no matter which local monitors are attacked, the route of the vehicle remains private.

5.3 Attack Tolerance

For security we also want to know how critical an attack on a monitor can be. That is, if a certain monitor's aggregate can be obtained, how bad is that? In contrast to robustness, this is only interesting if the collaborative monitor *does not* satisfy its security constraint. Because that would indicate that there are combinations of monitors such that the attack on those monitors would reveal sensitive information. The attack tolerance of a local monitor m indicates how many other monitors need to be attacked in addition to m before sensitive information is leaked. A local monitor is maximally attack tolerant if it cannot contribute to any security leakage at all.

Definition 15 (Attack Tolerant). *Let $C = (M, \mathsf{qry})$ be a collaborative monitor and (A, P) be its security constraint. For an attack $\mathsf{att} \subseteq A$ and a local monitor $m \in \mathsf{att}$ we say that m is contributing to att iff $P \cap \mathcal{L}^C_{\mathsf{att}} \neq \emptyset$ and $P \cap \mathcal{L}^C_{\mathsf{att}'} \subset P \cap \mathcal{L}^C_{\mathsf{att}}$, where and $\mathsf{att}' = \mathsf{att} \setminus \{m\}$. For a monitor $m \in M$ we define:*

- *m is k-*attack tolerant *iff $\mathsf{att} \subseteq A$ is the smallest attack such that m is contributing to att and $k = |\mathsf{att}|$.*
- *m is ∞-attack tolerant iff there is no $\mathsf{att} \subseteq A$ such that $m \in \mathsf{att}$ and m is contributing to att.*

If $C = (M, \mathsf{qry})$ Tr-satisfies a security constraint then all local monitors are ∞-attack tolerant. If a local monitor m is 1-attack tolerant then the m-aggregate is equivalent to a sensitive property, or the aggregate's negation is. The maximal attack tolerance value for a local monitor aside from ∞ is $|M|$.

Example 13 (Attack Tolerant). In our example scenario the monitor Tr-satisfies the example security constraint and hence all monitors are ∞-attack tolerant. If in the right monitor of Fig. 4 the security constraint is $(\{m_1, \ldots, m_7\}, \{p \wedge q\})$ then m_1 and m_2 are 2-attack tolerant and the other local monitors are 1-attack tolerant.

Now we can determine how attack tolerant a local monitor is, which is an indicator for how critical the local monitor is for the security of the collaborative monitor. Based on this basic framework several extensions are possible. For instance if a monitor is easier to attack than another (i.e. a simple sensor in a WSN versus a sophisticated sink), then the tolerance of the easier target should be decreased relatively to the harder target. In a runtime environment if attacks are detected then new attack tolerance values can be computed with the knowledge that some local monitors are already attacked. This could for instance be countered with low attack tolerance local monitors switching to more secure, albeit energy and/or more overhead cost expensive, communication protocols.

6 Discussion and Conclusion

In this paper we presented a formal framework for collaborative monitors, which are networks of local monitors. We have drawn inspiration from the wireless sensor network literature to specify the interaction between local monitors. The local monitors in a collaborative monitor are models of monitors that aggregate local and distributed observations. The model for monitoring is suitable for scenarios where it is unfeasible to have a centralized monitor that can observe the entire state of the system that is monitored. Also, aggregation is useful in scenarios where communication is expensive, as it can reduce the cost of communication. Aggregation may also improve the security of a monitor. We discussed how aspects related to robustness and safety can be investigated in the framework. The model for robustness and safety allows a designer to detect the importance of the correct functioning and safety of a local monitor. In an application that implements the model this may help the designer to decide upon what counter measures to take against potentially failing or attacked monitors.

The contribution of this paper is just a first step towards a formal framework for modeling and analyzing collaborative monitor applications. At several points we explained how the framework could be extended. The runtime analysis of communication failures and the effects of different communication protocols from related literature provide interesting directions for future research. Also, the concepts of robustness and safety can be extended to model, e.g., graceful degradation of a collaborative monitor's functioning and safety. We also left the investigation of formal properties of the framework, e.g. complexity and synthesis results, for future work.

References

1. Agotnes, T., Van Der Hoek, W., Rodriguez-Aguilar, J., Sierra, C., Wooldridge, M.: On the logic of normative systems. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 1181–1186 (2007)
2. Alechina, N., Dastani, M., Logan, B.: Reasoning about normative update. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI 2013, pp. 20–26. AAAI Press (2013)

3. Bauer, A., Falcone, Y.: Decentralised LTL monitoring. In: Giannakopoulou, D., Méry, D. (eds.) FM 2012. LNCS, vol. 7436, pp. 85–100. Springer, Heidelberg (2012)
4. Bauer, A., Leucker, M., Schallhart, C.: Runtime verification for LTL and TLTL. ACM Trans. Softw. Eng. Methodol. **20**(4), 14:1–14:64 (2011)
5. Boella, G., Torre, L.V.D.: Introduction to normative multiagent systems. Comput. Math. Organ. Theory **12**, 71–79 (2006)
6. Havelund, K., Rosu, G.: Monitoring programs using rewriting. In: Proceedings of the 16th Annual International Conference on Automated Software Engineering (ASE 2001), pp. 135–143. IEEE (2001)
7. Hoh, B., Gruteser, M., Xiong, H., Alrabady, A.: Enhancing security and privacy in traffic-monitoring systems. IEEE Pervasive Comput. **5**(4), 38–46 (2006)
8. Karlof, C., Wagner, D.: Secure routing in wireless sensor networks: attacks and countermeasures. Elsevier's AdHoc Netw. J. Spec. Issue Sens. Netw. Appl. Protoc. **1**(2–3), 293–315 (2003)
9. Kleene, S.C.: Introduction to Metamathematics. North-Holland Publisher Company, The Netherlands (1952)
10. Ozdemir, S., Xiao, Y.: Secure data aggregation in wireless sensor networks: a comprehensive overview. Comput. Netw. **53**(12), 2022–2037 (2009)
11. Pathan, A., Lee, H.-W., Hong, C.S.: Security in wireless sensor networks: issues and challenges. In: The 8th International Conference on Advanced Communication Technology, ICACT 2006, vol. 2, pages 6 pp. -1048, February 2006
12. Perrig, A., Stankovic, J., Wagner, D.: Security in wireless sensor networks. Commun. ACM **47**(6), 53–57 (2004)
13. Perrig, A., Szewczyk, R., Tygar, J.D., Wen, V., Culler, D.E.: SPINS: security protocols for sensor networks. Wirel. Netw. **8**(5), 521–534 (2002)
14. Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium onFoundations of Computer Science, pp. 46–57, October 1977
15. Pnueli, A., Zaks, A.: PSL model checking and run-time verification via testers. In: Misra, J., Nipkow, T., Sekerinski, E. (eds.) FM 2006. LNCS, vol. 4085, pp. 573–586. Springer, Heidelberg (2006)
16. Testerink, B., Bulling, N., Dastani, M.: A model for collaborative runtime verification. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, pp. 1781–1782 (2015)
17. Testerink, B., Dastani, M., Meyer, J.-J.: Norm monitoring through observation sharing. In: Herzig, A., Lorini, E. (eds.) Proceedings of the European Conference on Social Intelligence, pp. 291–304 (2014)