# Accepted Manuscript

Formal Analysis of a Calculus for WSNs from Quality Perspective

Xi Wu, Huibiao Zhu

PII: S0167-6423(17)30166-1
DOI: http://dx.doi.org/10.1016/j.scico.2017.08.007
Reference: SCICO 2130

To appear in: *Science of Computer Programming*

Received date: 30 April 2016
Revised date: 16 August 2017
Accepted date: 16 August 2017

## Highlights

- We extend our previous CWQ calculus to be a parametric framework, making it more flexible for modeling and reasoning about wireless networks with different topological structures.
- We develop a SAT-based robustness analysis for supporting the calculus to check whether some error configurations, caused by the unreliable communications in WSNs, will be reached. We also use the efficient SMT solver Z3 to check the satisfiability of program interesting points.
- We propose a data-driven probabilistic trust analysis to decouple the probability of receiving data from the probability of data trustworthiness, which makes more flexible probabilistic analysis possible.
- We give a real-world case study with the scenario of refueling a car to illustrate the applicability of the extended calculus and these two analysis approaches.

# Formal Analysis of a Calculus for WSNs from Quality Perspective [1]

Xi Wu[a,b], Huibiao Zhu[2][a,c]

[a]*Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China*
[b]*School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, QLD 4072, Australia*
[c]*College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China*

**Abstract**

In viewing the common unreliability problem in wireless communications, the CWQ calculus (a Calculus for Wireless sensor networks from Quality perspective) was recently proposed for modeling and reasoning about WSNs (Wireless Sensor Networks) and their applications from a quality perspective. The CWQ calculus ensures that sensor nodes, even though in an unreliable communication network, can still behave in a reasonable manner using default values. Nevertheless, the topological structure in CWQ calculus is considered at the network level and it is tightly coupled with the processes and other configurations; this may limit its flexibility. In this paper, we extend our previous CWQ calculus to be a parametric framework to make it more flexible to be able to model and reason about networks of different topological structures. In the parametric framework, we extract the topological structure of a network and make it to be a configuration so that all topological structure changes can be captured by this framework.

Besides, under the guide of program analysis, in this paper, we propose two analysis approaches for the extended CWQ calculus: 1) We develop a SAT-based analysis to check whether default values can always be available, so that some error configurations (e.g., deadlock processes), caused by unreliable communications in WSNs, will not be reached; 2) We also propose a data-driven probabilistic trust analysis to decouple the probability of receiving the expected input data from the probability of the trustworthiness of the data, so that the overall trustworthiness of the system decision is determined by performing a relational analysis to combine these two probability distributions. Finally, we give a real-world case study with the scenario of refueling a car to demonstrate the applicability of the extended calculus and these two analysis approaches on the extended calculus for WSNs.

*Keywords:* Process Calculus, Formal Analysis, WSNs, Unreliability, Probabilistic Trustworthiness

## 1. Introduction

WSNs (Wireless Sensor Networks), one of the key components of CPS (Cyber Physical Systems) [20], have drawn more and more attention recently from both academia and industry. Significant examples range from distributed computing to high trustworthy medical systems, traffic control and traffic security management systems, emergency rescue missions and to disaster recovery. One important feature of wireless systems is broadcast, and many process algebras have been proposed to study it at the modeling level. For example, broadcasting systems are specified by calculi [8, 33, 34], in which broadcast messages can reach every node in the network. However, this may not be well suit for wireless local broadcast in which only limited nodes that are in the transmission area of the sender can receive the messages broadcasted by the sender. Therefore, research efforts are made to define a formal process calculus for modeling and reasoning about the wireless local broadcast. In these models, different manners are defined to handle the neighborhood relationship (e.g., define it either as a part of the syntax [21] or as a part of the semantics [9, 19, 24, 26]). Nevertheless, none of the above models takes the service quality offered by the system into account.

In reality, communications in WSNs are often unreliable, which may be caused by deployment constraints and/or communication modalities; this may result in abnormalities and thus decrease the quality of service provided by the system. Therefore, it is of significant importance to ensure that wireless sensor nodes can still behave in a reasonable

[1]A short version of this paper appeared in FTSCS 2015: 4th International Workshop on Formal Techniques for Safety-Critical Systems [42].
[2]Corresponding author. Email address: hbzhu@sei.ecnu.edu.cn

manner even though they are in an unreliable communication network (e.g., by using approximate values to continue their work when the ideal behavior of a node fails). As a first step in this very important direction, Nielson et al. proposed a Quality Calculus [32]. It enforces robustness consideration on software components executed in an open and error prone environment, and specifies the behavior when communication links break down. Later, Vigo et al. [37] extended the Quality Calculus by considering broadcast communication. However, Vigo's work focuses on the process level and they want to consider "enriching the framework with a notion of network topology and spatially-bounded broadcast" in their future work.

**The CWQ Calculus.** To be taken one step further, the CWQ calculus [41] was recently proposed for modeling and reasoning about WSNs and applications based on WSNs. The CWQ calculus combines the wireless local broadcast together with quality predicate that is inherited from Nielson's Quality Calculus, so that it can not only capture the feature of WSNs, but also consider the communication quality of WSNs at the same time. Specifically, the topological structure is considered at the network level and different node behaviours are represented by processes. Broadcast actions and unicast actions are distinguished via different kinds of channels, and a distance function is given to check whether one node is inside the transmission area of another one or not. Moreover, default values (of the same type as the expected ones) are given to deal with the situations that the ideal behavior of a sensor node fails due to unreliable communications. Nevertheless, in the CWQ calculus, the topological structure is considered at the network level and it is tightly coupled with the processes and other configurations, which may limit its flexibility.

Besides, the CWQ calculus includes an input guard, binder, to specify the input to be performed before continuing. It makes the CWQ calculus have the flexibility that not all input data in a binder need to be received in order for the process to continue (e.g., when the quality predicate of a binder is $\exists$, details can be found in Section 2). System decision may have different levels of trustworthiness depending on which input data have actually been received. That is, different input data has different levels of trustworthiness (i.e., the trust of a data, which can be regarded as the utility of the data in the decision of the entire system, details can be found in Section 6). Specifically, the decision about a system of WSNs is expected to be made based on the data with the highest level of trustworthiness among data received from all network nodes in WSNs, and the system decision may have the highest level of trustworthiness if data from all its constituent network nodes are received and considered. In other words, from the perspective of a single network node, its locally stored data may not be sufficient for making the best decision about a system. In the literature, Nielson et al. [31] developed a novel probabilistic *trust* analysis for supporting the Quality Calculus [32] to indicate the trust that a user can have in the overall robustness of a system. However, it is not applicable to the CWQ calculus for WSNs because of the above characteristic of CWQ and WSNs.

**Contributions.** In this paper, to make the CWQ calculus more flexible for modeling and reasoning about networks of different topological structures, we extend it by modifying and simplifying it to be a parametric framework. In the parametric framework, we use an undirected graph to describe the topology of the entire network as a configuration, such that all topological structure changes can be easily captured by this framework. Instead of distinguishing between broadcast channels and unicast channels, we use two partial functions to illustrate the different transmission areas of different nodes when using different channels.

Furthermore, two formal analysis approaches are proposed. 1) We develop a SAT-based analysis to support the extended calculus by detecting whether some error configurations, caused by the unreliable communications in WSNs, can be reached or not. In our approach, all the interesting program points are characterized and represented by propositional formulae. To perform the analysis, we use an existing efficient SAT [22] and SMT [6] solver, Z3 [5], to check whether the formulae are satisfied or not; that is, the program points cannot be reached if and only if the corresponding formulae are not satisfied. 2) We also propose a data-driven probabilistic trust analysis for the CWQ Calculus. Instead of only giving the channel a trust value to illustrate the probability of receiving the expected data, we assume that the data received from the channel also have a trust value, where the trust value of a data represents the trust of the system decision made solely based on that data. Thus, we decouple the probability of receiving expected input data from the probability of trustworthiness levels of the receiving data. The overall trustworthiness of the system decision is determined by performing a relational analysis to combine these two probability distributions. Finally, to illustrate the applicability of the extended calculus and these two analysis approaches, we present a real-world case study with the scenario of refueling a car.

This paper combines and extends our previous work at TASE 2015 [40] and FTSCS 2015 [42]. In order to make the CWQ calculus more flexible for modeling and reasoning about WSNs, a parametric framework is proposed in the previous work [40] and a SAT-based analysis as an enforcement mechanism is developed to check whether default values in CWQ calculus can always be available, in case that the expected data is lost, to provide meaningful behaviors. However, even though a better quality of service may be obtained when making decisions on default or old data, rather than simply stopping in an error state, default data is not as useful as the expected data (i.e., the trustworthiness level of default data is not as high as the one of expected data), which may affect the trustworthiness of the final decision of WSNs. The work [42] focuses on a data-driven probabilistic trust analysis, which decouples the probability of receiving expected input data from the probability of the trustworthiness level of the receiving data. Through this approach, it is possible to observe the effect of the data trustworthiness produced on the system decision. Thus, we put these two approaches together in this paper to make a general and useful analysis framework for CWQ calculus and WSNs. The framework does not only capture the feature of the network, but also take the communication quality into consideration. More intuitive and detailed explanations about these two analysis approaches are added and a more comprehensive related work is given from four perspectives in this paper. We believe that this analysis framework can give a better guide for the design of WSNs and the implementation of their applications.

**Organization.** The paper is organized as follows. In the next section we present the parametric CWQ calculus from the process level and network level by introducing an undirected topology graph as configurations. The corresponding parametric Labeled Transition Semantics (LTS) is given in Section 3. In Section 4 we illustrate a motivating example with the scenario about refueling a car by modeling it using our calculus. A SAT-based analysis of the calculus is given in Section 5, in which the analysis of the motivating example is also presented. In Section 6, a data-driven probabilistic trust analysis is proposed and the need for this analysis approach is illustrated through the motivating example. Finally, Section 7 discusses related work, and Section 8 refers to the conclusion of this paper and presents future directions of our research.

Table 1: The Syntax of Parametric CWQ Calculus

Processes:

$P \ ::= \ 0 \mid Act.P \mid A(\tilde{x}) \mid \text{case } e \text{ of some}(y) : P_1 \text{ else } P_2$

$Act \ ::= \ b \mid c!d \qquad b \ ::= \ c?x \mid \&_q(b_1, ..., b_n)$

$d \ ::= \ c \mid v \mid y \qquad e \ ::= \ x \mid \text{some}(d) \mid \text{none}$

Networks:

Network $N$ has the form: $n_1[P_1] \parallel n_2[P_2] \parallel ... \parallel n_k[P_k]$

## 2. Syntax of the Parametric CWQ Calculus

In this section, we extend the CWQ calculus by modifying and simplifying it. We propose a two-level syntax for interpreting processes and networks of the extended calculus. We follow the standard definition of process calculi for the process part, but omit name restriction for simplicity.[3] We employ $P$, $Q$ to range over the set of all processes, and $N$ the set of all nodes. We also use the set $\mathcal{I}_n$ to denote the node identities (e.g., names together with IP addresses and port numbers), where $n_1$, $n_2$, ... range over $\mathcal{I}_n$.

The syntax of our calculus is illustrated by the Backus-Naur form in **Table 1**. 0 stands for the terminated process. $c!v$ denotes an output of a value $v$, while the corresponding reception of an output is represented by $c?x$ which receives a value via channel $c$ and binds it to the variable $x$. $A(\tilde{x})$ denotes a process with the (possibly recursive) definition of $A(\tilde{x}) \overset{df}{=} P$, where $A$ is a process constant and $\tilde{x}$ contains all free variables that appear in $P$.

[3]Name restriction can be considered as a future extension to our calculus. So we in this paper do not have any process like $(\nu x)P$ which denotes a process with the bounded name $x$.

One interesting and important thing in the calculus is the binder $b$, which is used to specify that the process will continue after the quality predicate being satisfied. The binder is first proposed in the Quality Calculus [32] with the form of $\&_q(b_1, ..., b_n)$, where $n$ is the total number of inputs and $q$ is a quality predicate to be satisfied, indicating to continue the process which sufficient inputs have to be received. Here, $q \in \{\forall, \exists, \exists!, m/n\}$ and the corresponding meanings of the four notations are as follows:

- $\forall$: all inputs are required, e.g., $\&_\forall(c_1?x_1, c_2?x_2, c_3?x_3)$ requires three sufficient inputs and it has the same effect as $\&_q(c_1?x_1, c_2?x_2, c_3?x_3)$ if $q(r_1, r_2, r_3)$ amounts to $r_1 \wedge r_2 \wedge r_3$.
- $\exists$: at least one of the inputs is required, e.g., $\&_\exists(c_1?x_1, c_2?x_2, c_3?x_3)$ requires one sufficient input to continue and it has the same effect as $\&_q(c_1?x_1, c_2?x_2, c_3?x_3)$ if $q(r_1, r_2, r_3)$ amounts to $r_1 \vee r_2 \vee r_3$.
- $\exists!$: only one input is required, e.g., $\&_{\exists!}(c_1?x_1, c_2?x_2, c_3?x_3)$ requires just one sufficient input to continue, otherwise, it stops.
- $m/n$: $m$ sufficient inputs of all $n$ inputs are required to be accepted, e.g., $\&_{2/3}(c_1?x_1, c_2?x_2, c_3?x_3)$ requires at least two sufficient inputs from the channels.

Moreover, nested binders are also allowed in this paper, such as $\&_\forall(\&_\exists(c_1?x_1, c_2?x_2), c_3?x_3)$, which represents that input must be received both over the channel $c_3$ and over either the channel $c_1$ or $c_2$. However, it is possible that some variables in the binder do not get proper values, due to that the corresponding inputs have not occurred. Therefore, we distinguish between *data* and *optional data*. We use term $d$ (including channel constant $c$, value $v$ and variable $y$) to denote data, which actually contains values inside the variables. Note that, because we do not have any operation to create a new channel in our syntax, we just regard the channel names as part of constants which exist in advance. Thus, in our syntax, constants can be divided into channel constants $c$ and values $v$. We use expression $e$ (including variable $x$ and two special expressions) to stand for optional data, which may not get proper values inside the variables; in particular, the expression some($d$) represents the presence of some data $d$ and none the absence of data. If the variable $x$ actually receives value $v$ from a channel, it can be evaluated into some($v$); otherwise, it will be evaluated into none. We use $\xi$ to stand for the evaluation function and the evaluation rules for terms and expressions are shown in Table 2.

Table 2: The Evaluation of Terms and Expressions

$$\xi(c) = c \qquad \xi(v) = v \qquad \xi(\text{none}) = \text{none}$$

$$\frac{\xi(d) = v}{\xi(\text{some}(d)) = \text{some}(v)}$$

$$\xi(x) = \begin{cases} \text{some}(d) & \text{if} \quad \text{x receives value via channel} \\ \text{none} & \text{if} \quad \text{otherwise} \end{cases}$$

Due to we cannot make sure which variable has actually received values at some time, there comes a construct case $e$ of some($y$) : $P_1$ else $P_2$ in the process part. It is used to check whether an expression $e$ evaluates to some data or not; that is, if the variable in the expression receives a value, then it can be evaluated into some($d$). If it does, then we bind it to $y$ and continue with $P_1$; otherwise, we continue with $P_2$.

Networks are collections of nodes running in parallel.[4] Each node, written as $n[P]$, is assigned a unique identity $n$ and runs a process $P$. The topology $T$ of a network is specified by an undirected graph $G$ and a radius constraint $Rad$,

[4]Fixed network form is given because of no restriction in process level.

Table 3: The Semantics for Processes

$$\text{(SEND) } c!v.P \xrightarrow{c!v} P \qquad \text{(RECV) } c!v \vdash c?x \rightarrow [\text{some}(v)/x]$$

$$\text{(CSE1) case some}(v) \text{ of some}(y) : P \text{ else } Q \;\rightarrow\; P[v/y]$$

$$\text{(CSE2) case none of some}(y) : P \text{ else } Q \;\rightarrow\; Q \qquad \text{(REC) } \frac{A(\tilde{x}) \overset{df}{=} P}{A(\tilde{y}) \rightarrow P[\tilde{y}/\tilde{x}]}$$

$$\text{(QSD1) } \frac{c!v \vdash b \rightarrow b' \quad b' ::_{\text{ff}} \theta}{b.P \xrightarrow{c?x} b'.P} \qquad \text{(QSD2) } \frac{c!v \vdash b \rightarrow b' \quad b' ::_{\text{tt}} \theta}{b.P \xrightarrow{c?x} P\theta}$$

$$\text{(QREC) } \frac{c!v \vdash b_i \rightarrow b_i'}{c!v \vdash \&_q(b_1, ..., b_i, ..., b_n) \rightarrow \&_q(b_1, ..., b_i', ..., b_n)}$$

$$\text{(JDG1) } [\text{some}(v)/x] ::_{\text{tt}} [\text{some}(v)/x] \qquad \text{(JDG2) } c?x ::_{\text{ff}} [\text{none}/x]$$

$$\text{(SAT) } \frac{b_1 ::_{\sigma_1} \theta_1 .. b_i ::_{\sigma_i} \theta_i .. b_n ::_{\sigma_n} \theta_n}{\&_q(b_1, .., b_i, .., b_n) ::_{\sigma} \theta_1 ... \theta_i ... \theta_n} \quad \text{where } \sigma = [\{q\}](\sigma_1, .., \sigma_i, .., \sigma_n)$$

where the graph $G$ consists of a finite set of nodes *Node* and the set of edges *Edge* between these nodes, as shown below.

$$T \;=\; (G, Rad) \qquad G \;=\; (Node, Edge)$$

We denote the graph in a given topology $T$ and the set of nodes in a given graph $G$ by $G(T)$ and $Node(G)$, respectively. *Rad* describes the transmission radius of a node in $G$, which does not only depend on the node itself, but also is related to the current channel the node uses. It is defined as a partial function of:

$$Rad \;:\; Chan * Node \hookrightarrow R_0^+$$

*Chan* is a finite set of channels. We can use this partial function to distinguish between different kinds of channels; for example, $Rad(c_1, n_1) = 0$ means that channel $c_1$ is used for the unicast communication between $n_1$ and local computers, while $Rad(c_2, n_2) = 3$ denotes that channel $c_2$ is used for the communication between $n_2$ and other nodes that are inside the transmission range of $n_2$ with a radius 3.

$$Edge \;:\; Node * Node \hookrightarrow R_0^+$$

*Edge* is also a partial function which assigns distances to node-pairs $(n_i, n_j)$ in $G$, which satisfies symmetry:

$$\forall n_i, n_j \in Node \;\; Edge(n_i, n_j) = Edge(n_j, n_i),$$

and the triangle inequality:

$$\begin{aligned} &\forall n_i, n_j, n_k \in Node \\ &\quad Edge(n_i, n_j) \;>\; |Edge(n_i, n_k) - Edge(n_j, n_k)| \\ &\quad Edge(n_i, n_j) \;<\; |Edge(n_i, n_k) + Edge(n_j, n_k)| \end{aligned}$$

## 3. Labeled Transition Semantics

In our simplified CWQ calculus, as the processes and networks are interpreted separately by a two-level syntax, the labeled transition system is also divided into two levels: transition for processes and transition for networks. The rules for processes have two forms: $P \to P'$ and $P \xrightarrow{\lambda} P'$. The former one represents the rules for the internal actions, while the latter one is used for the communication actions. The syntax of the signal $\lambda$ in the form $P \xrightarrow{\lambda} P'$ is defined as follows:

$$\lambda ::= c!v \mid c?x$$

Here, $c!v$ stands for sending a data $v$ via channel $c$, while $c?x$ represents the corresponding receiving and then assigning the value to the variable $x$. Some auxiliary relations are also used, as shown below:

$$c!v \vdash b \to b' \quad \text{and} \quad b ::_{\sigma} \theta \quad \text{where} \quad \sigma \in \{\mathrm{tt}, \mathrm{ff}\}$$

The former one specifies that the binder $b$ is changed to $b'$ after receiving an output $c!v$. The later one is used to check whether the required inputs in binder $b$ have already been satisfied ($::_{\mathrm{tt}}$) or not ($::_{\mathrm{ff}}$). If all the required inputs are satisfied, a substitution $\theta$ is constructed to replace all the variables with the receiving values, i.e., $c?x ::_{\mathrm{tt}} [\mathrm{some}(v)/x]$ and $c?x ::_{\mathrm{ff}} [\mathrm{none}/x]$. Here, we give an *id* to each $\theta$ to identify the execution order of these substitutions; thus, the composition $(\theta_1\theta_2)(x)$ is equivalent to $\theta_2(\theta_1(x))$ for all $x$.

**Table 3** contains the labeled transition semantics for processes. Rules (SEND) and (RECV) refer to the primitive output and input of values respectively. After receiving a value via channel $c$, a substitution $\theta$ is constructed as $[\mathrm{some}(v)/x]$. Rules (CSE1) and (CSE2) are standard for the case construct case $e$ of some($y$) : $P$ else $Q$, which evaluate the expression $e$ to a constant with the form some($c$) and none, respectively. Rule (QSD1) defines that after the binder $b$ receiving an output, the required inputs in $b$ still cannot be satisfied, thus more inputs are required, while the Rule (QSD2) denotes that no more inputs are needed. A more complicated situation is given in the rule (QREC) and the general idea is to record the binding of the value received in the appropriate position. For these three rules, they stand for the synchronization with quality binder, that is, the original binder is replaced by a new one recording the output just performed. Besides, as we mentioned before, the auxiliary relation $b ::_{\sigma} \theta$ is defined to evaluate the binder $b$ for checking whether or not a sufficient number of inputs has been performed (i.e., recorded in $\sigma$) and for computing the associated substitution $\theta$, which is shown by the rules (JDG1), (JDG2) and (SAT). The semantics of the example quality predicates are listed below:

- $[\{\forall\}](\sigma_1, ..., \sigma_n) = (|\{i \mid \sigma_i = \mathrm{tt}\}| = n) = \sigma_1 \wedge ... \wedge \sigma_n$
- $[\{\exists\}](\sigma_1, ..., \sigma_n) = (|\{i \mid \sigma_i = \mathrm{tt}\}| \geq 1) = \sigma_1 \vee ... \vee \sigma_n$
- $[\{\exists!\}](\sigma_1, ..., \sigma_n) = (|\{i \mid \sigma_i = \mathrm{tt}\}| = 1)$
- $[\{m/n\}](\sigma_1, ..., \sigma_n) = (|\{i \mid \sigma_i = \mathrm{tt}\}| \geq m)$

$|X|$ denotes the cardinality of the set $X$. Formally, $\&_{\exists}(x_1, \ldots, x_n)$ equals to $x_1 \vee \cdots \vee x_n$ and $\&_{\forall}(x_1, \ldots, x_n)$ equals to $x_1 \wedge \cdots \wedge x_n$. Here, we also allow to write the quality predicate as $\&_{[0\wedge(1\vee2)]}(x_1, x_2, x_3)$ which is equivalent to $x_1 \wedge (x_2 \vee x_3)$. Rule (REC) is a standard one for recursion. Finally, transitions can take place in contexts $C$ by rule (CON) and the replacement in $C$ is also allowed which is shown as follows:

$$(\mathrm{CON})\ \frac{P \xrightarrow{\lambda} P'}{C[P] \xrightarrow{\lambda} C[P']} \qquad \textit{where} \quad C ::= [\ ] \mid C|P \mid P|C$$

We now use a parameterized operational semantics to define the formal transitional rules for the networks of our calculus. Transitions are of the form $T \vdash N \xrightarrow{\alpha} N'$, where the action $\alpha$ is defined as:

$$\alpha ::= c!v©n \mid c?x©n \mid \tau$$

The parameter $T$ refers to the topology of the entire network at some moment and $N$ refers to the network composed by the nodes inside the given topology $T$. For the actions, $c!v©n$ denotes that a node identified $n$ sends a message $v$ to its neighbors using channel $c$; while $c?x©n$ refers to the corresponding receiving from a node identified $n$; $\tau$ is an internal action inside a network.

The labeled transition semantics for networks is defined in **Table 4**. Either one node can perform an internal action in rule (INT1) or it can keep unchanged in rule (INT2). Rule (BRO) denotes that a node, identified n, can send

Table 4: The Semantics for Networks

$$(\text{INT1})\ \frac{P \rightarrow P'}{T \vdash n[P] \xrightarrow{\tau} n[P']} \qquad (\text{INT2})\ T \vdash n[P] \xrightarrow{\tau} n[P] \qquad (\text{BRO})\ \frac{P \xrightarrow{c!v} P'}{T \vdash n[P] \xrightarrow{c!v\textcircled{c}n} n[P']}$$

$$(\text{REC1})\ \frac{P \xrightarrow{c?x} P' \ \wedge\ n,m \in Node(G(T)) \ \wedge\ Rad(c,n) \ \geq\ Edge(n,m)}{T \vdash m[P] \xrightarrow{c?x\textcircled{c}n} m[P']}$$

$$(\text{REC2})\ \frac{n,m \in Node(G(T)) \ \wedge\ Rad(c,n) \ <\ Edge(n,m)}{T \vdash m[P] \xrightarrow{c?x\textcircled{c}n} m[P]} \qquad (\text{REC3})\ \frac{n,m \in Node(G(T)) \ \wedge\ P \ \not\xrightarrow{c?\_}}{T \vdash m[P] \xrightarrow{c?x\textcircled{c}n} m[P]}$$

$$(\text{BSYN})\ \frac{\begin{array}{c} i,j \in \{1,...,k\} \ \wedge\ k \in \mathbb{N} \ \wedge\ n_1,...,n_k \in Node(G(T)) \ \wedge\ \forall j \neq i \bullet \\ T \vdash n_i[P_i] \xrightarrow{c!v\textcircled{c}n_i} n_i[P'_i] \qquad T \vdash n_j[P_j] \xrightarrow{c?x\textcircled{c}n_i} n_j[P'_j] \end{array}}{T \vdash n_1[P_1] \parallel ... \parallel n_i[P_i] \parallel ... \parallel n_k[P_k] \xrightarrow{c!v\textcircled{c}n_i} n_1[P'_1] \parallel ... \parallel n_i[P'_i] \parallel ... \parallel n_k[P'_k]}$$

$$(\tau\text{SYN})\ \frac{\forall i \in \{1,...,k\} \ \wedge\ k \in \mathbb{N} \ \wedge\ n_1,...,n_k \in Node(G(T)) \ \bullet T \vdash n_i[P_i] \xrightarrow{\tau} n_i[P'_i]}{T \vdash n_1[P_1] \parallel ... \parallel n_i[P_i] \parallel ... \parallel n_k[P_k] \xrightarrow{\tau} n_1[P'_1] \parallel ... \parallel n_i[P'_i] \parallel ... \parallel n_k[P'_k]}$$

a message $v$ via channel $c$ and the executing process $P$ evolves into $P'$. Three corresponding receivings are listed in rules (REC1), (REC2) and (REC3). Taking local broadcast into account, only the nodes that are located inside the transmission area of the sending node can receive the message according to rule (REC1). The other nodes, which are outside the transmission area of the sender or cannot execute receiving actions, will remain unchanged, based on rule (REC2) and (REC3), respectively. Note that the notation $P \ \not\xrightarrow{c?\_}$ in rule (REC3) represents that the process $P$ cannot execute the receive action from channel $c$ currently. Rule (BSYN) specifies the parallel composition of the entire network when the nodes execute the sending action, as well as the rule ($\tau$SYN) describes the parallel composition of the entire network when the nodes execute the internal action.

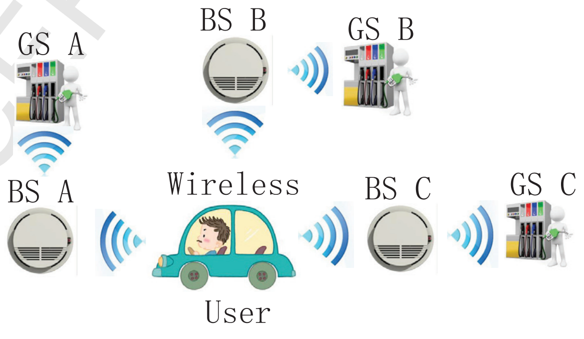

Figure 1: The Architecture of the Motivating Example

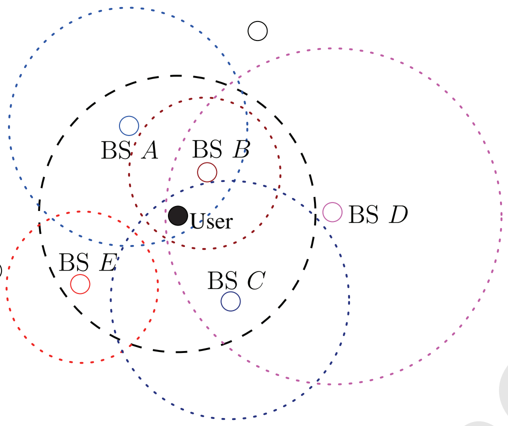

Figure 2: The Local Broadcast Communication in Motivating Example

## 4. Motivating Example

In this section, we present a case study to demonstrate the applicability of the proposed calculus in real-world applications. In our case study, we concentrate on the scenario that a car is running out of gas and the driver wants to find the nearest gas station to refuel the car. The architecture of the motivating example is shown in Figure 1.

The request of finding the closest gas station (GS) is accomplished by broadcasting the request in a wireless network and then receiving replying messages that contain locations of GSs. The wireless network consists of a set of base stations (BSs), where the user (e.g., the driver together with the car) can also be regarded as a BS. Each BS has a transmission area constraint (e.g., illustrated as dotted circles in Figure 2). That is, when a BS broadcasts a message, only other BSs that are within its transmission area (i.e., within a certain distance) can receive the message. We assume that each BS stores some information of GSs (i.e., locations of a subset of GSs) so that the location of a GS is stored in the local cache of the closest BS (or several closest BSs).

To find the closest GS to the location of the user, the user broadcasts a request to BSs in a wireless network and then waits for replies. Ideally, if every BSs in the wireless network replies its locally stored GSs to the user, then the user can obtain the closest GS by iterating through all the replied GSs. However, in real wireless networks, a BS cannot (or may not) send its stored GSs to the user due to several reasons. For example, 1) the BS does not receive the request since it is not in the transmission area of the user (e.g., BS $D$ in Figure 2); 2) the user is not in the transmission area of the BS even if the BS receives the request (e.g., BS $E$ in Figure 2); 3) although the BS sends its replying message to the user and the user is in its transmission area, the message may be lost in the transmission process due to unreliable wireless communications. Consequently, the user has to make a decision based on the locations of a subset of GSs it received, and the GSs stored at each BS have a probability to contain the closest GS to the user. In the worst case, the user may not even receive any replies. Therefore, we assume the user has a local computer (or other electronic devices) which caches previously searched closest GSs, and the user will choose the closest one among these locally cached GSs as a candidate if it receives no replies.

$$
\begin{aligned}
Network &\overset{df}{=} User \parallel (\prod_{i\in\mathbb{Z}} BS_i \parallel \prod_{k\in\mathbb{Z}} GS_k)\\
User &\overset{df}{=} n_{11}[P_{user}] \parallel n_{12}[Local_{user}] \parallel n_{13}[Timer_{user}]\\
BS_i &\overset{df}{=} n_{2i}[P_{bs}]\\
GS_k &\overset{df}{=} n_{3k}[P_{gs}]
\end{aligned}
$$

The whole *Network* consists of the *User*, several *BS* and several *GS*. The *User* stands for the driver together with his car in the motivating example, which is composed of three nodes $n_{11}$, $n_{12}$ and $n_{13}$. Each of these nodes runs its own process inside. The process $P_{user}$ is the main process of the subnetwork *User*, and it may be triggered by the actions (e.g., touching the screen or pressing the button) of the driver to start the communications. While the process $Local_{user}$ and $Timer_{user}$ play the roles of local computer and a counter, respectively. Note that, the rule for naming the node identification here is that, the first number represents the part this node belongs to and the second number denotes the index of the node. For example, node $n_{11}$ means that this node belongs to the first part *User* of the whole network and it is the first node in this part. The overall scenario modeled by CWQ calculus is shown above.

```
$P_{user} \overset{df}{=}$ $us!req.local!req.timer!(t_1, t_2).$
    $\&_{\forall}(timer?x_{t_1}, \&_{\exists}(us?x_{rep_A}, us?x_{rep_B}, us?x_{rep_C}, local?x_{rep'})).$
    case $x_{rep_A}$ of some($y_{rep_A}$) :
        case $x_{rep_B}$ of some($y_{rep_B}$) :
            case $x_{rep_C}$ of some($y_{rep_C}$) :$^1$ use($\widehat{\min}(y_{rep_A}, y_{rep_B}, y_{rep_C})$)
            else$^2$ use($\widehat{\min}(y_{rep_A}, y_{rep_B})$)
        else
            case $x_{rep_C}$ of some($y_{rep_C}$) :$^3$ use($\widehat{\min}(y_{rep_A}, y_{rep_C})$)
            else$^4$ use($y_{rep_A}$)
    else
        case $x_{rep_B}$ of some($y_{rep_B}$) :
            case $x_{rep_C}$ of some($y_{rep_C}$) :$^5$ use($\widehat{\min}(y_{rep_B}, y_{rep_C})$)
            else$^6$ use($y_{rep_B}$)
        else
            case $x_{rep_C}$ of some($y_{rep_C}$) :$^7$ use($y_{rep_C}$)
            else
                $\&_{\forall}(timer?x_{t_2}, \&_{\exists}(us?x_{rep_A}, us?x_{rep_B}, us?x_{rep_C}, local?x_{rep'})).$
                case $x_{rep_A}$ of some($y_{rep_A}$) :
                    case $x_{rep_B}$ of some($y_{rep_B}$) :
                        case $x_{rep_C}$ of some($y_{rep_C}$) :$^8$ use($\widehat{\min}(y_{rep_A}, y_{rep_B}, y_{rep_C})$)
                        else$^9$ use($\widehat{\min}(y_{rep_A}, y_{rep_B})$)
                    else
                        case $x_{rep_C}$ of some($y_{rep_C}$) :$^{10}$ use($\widehat{\min}(y_{rep_A}, y_{rep_C})$)
                        else$^{11}$ use($y_{rep_A}$)
                else
                    case $x_{rep_B}$ of some($y_{rep_B}$) :
                        case $x_{rep_C}$ of some($y_{rep_C}$) :$^{12}$ use($\widehat{\min}(y_{rep_B}, y_{rep_C})$)
                        else$^{13}$ use($y_{rep_B}$)
                    else
                        case $x_{rep_C}$ of some($y_{rep_C}$) :$^{14}$ use($y_{rep_C}$)
                        else
                            case $x_{rep'}$ of some($y_{rep'}$) :$^{15}$ use($y_{rep'}$)
                            else$^{16}$ 0
```

Figure 3: The Model of the Process $P_{user}$ in the Network *User*

Firstly, we give the details about the main process $P_{user}$ inside the node $n_{11}$ in Figure 3. There are three channels: *us* is used to communicate (i.e., broadcast and receive) with the base stations; while *local* and *timer* are used to communicate with the local computer and timer, respectively. We use 16 labels to mark 16 interesting points in the process model for the following analysis.[5] The driver initiates the process by broadcasting the request for finding the closest gas station to base stations through the wireless network. Note that, only the base stations within the driver's transmission area can receive the request. Here, we assume that the driver only has three neighbor base stations *BS A*, *BS B* and *BS C*, who are located inside the driver's transmission area.

At the same time, the driver also sends the request to its local computer which will return a historical record; this record will serve as the default value if the driver gets no reply from base stations when the timeout is reached. Then, the driver starts a timer and waits for at least $t_1$ time units but at most $t_2$ time units. Comparing with the transmission time between the driver and its local computer, the time for the message transmission between the driver and the base station is longer. Due to the historical record as a default value is not as useful as the fresh data from base stations, thus we enforce the driver to wait at least $t_1$ time units to leave enough time for the message transmission between the driver and the base stations.

When $t_1$ time units are reached, the driver will check the received replies. If at least one reply is received (maybe several replies), then the driver will check all the gas stations in the replies and choose the closest one to refuel the car, and the process continues; otherwise, the process still needs to wait $t_2 - t_1$ time units. Once $t_2$ time units are reached, the driver will check all the gas stations in the replies and choose the closest one to refuel the car; note that, if there is still no reply received, then the driver will use the gas station in the historical record, otherwise, the process will stop. More details of the definition for function $\widehat{\text{min}}()$ can be found in the end of this section. The meaning of the function use() is quite straightforward, which means that the process will take this reply message and perform some further actions.

The process $Local_{user}$ in node $n_{12}$ is used to search the previously computed closest gas station based on the driver's current location as a default value. Node $n_{12}$ is the local computer of *User*.

$$\begin{aligned} Local_{user} \overset{df}{=}\ & local?x_{req}.\text{case } x_{req} \text{ of some}(y_{req}) : local!\widehat{\text{min}}(\text{search}(y_{req})).Local_{user} \\ & \qquad\qquad \text{else } 0 \end{aligned}$$

$Timer_{user}$ is the process of node $n_{13}$, which is a counter of *User*. It will return a special signal ✓ when the timeout is reached. The meaning of the function count() is also quite straightforward, which is used to count a time interval. The process $Timer_{user}$ is defined as follows:

$$\begin{aligned} Timer_{user} \overset{df}{=}\ & timer?(x_{t_1}, x_{t_2}).\text{case } x_{t_1} \text{ of some}(y_{t_1}) : \\ & \qquad \text{count}(y_{t_1} - 0).timer!\checkmark.\text{case } x_{t_2} \text{ of some}(y_{t_2}) : \\ & \qquad \text{count}(y_{t_2} - y_{t_1}).timer!\checkmark.Timer_{user} \\ & \qquad \text{else } 0 \end{aligned}$$

Once a base station (e.g., *BS A*) receives the request, it checks the gas stations stored in its local cache. If there are any gas stations stored, then the base station chooses among all these gas stations the closest one to the driver that issues the request, and the position of the gas station is returned to the driver. Otherwise, the base station ignores the request. The *BS* is defined as follows.

$$\begin{aligned} BS \overset{df}{=}\ & n_2[P_{bs}] \\ P_{bs} \overset{df}{=}\ & us?x_{req}.\text{case } x_{req} \text{ of some}(y_{req}) : \\ & \qquad us!\widehat{\text{min}}(\text{search}(y_{req})).P_{bs} \\ & \qquad \text{else } 0 \end{aligned}$$

Here, search() is a function in the process for searching some records based on the request. Besides, we also define a new function $\widehat{\text{min}}()$ for choosing the minimal value from several values instead of using the traditional minimal choosing function min(). We want to distinguish these two functions by the different type of their signatures.

[5]Interesting points for analysis can either be chosen by designers or be generated automatically by some codes.

The traditional function min() is usually applied for numbers, whereas the new function $\widehat{\min}()$ is used on tuples, because the reply may contain several elements in one message (e.g., $reply = \; < send.receiver.distance >$). We give the definition of function $\widehat{\min}()$ below and we use a special function $\Xi(rep)()$ to get the distance element from the reply.

$$\widehat{\min}(rep_1, rep_2, ..., rep_k) \overset{df}{=} \begin{cases} rep_1, & \Xi(rep_1) \leq \Xi(\widehat{\min}(rep_2, ..., rep_k)) \vee k = 1; \\ \widehat{\min}(rep_2, ..., rep_k), & \Xi(rep_1) > \Xi(\widehat{\min}(rep_2, ..., rep_k)) \end{cases}$$

## 5. A SAT-Based Robustness Analysis

As we mentioned before, the ideal behavior of wireless sensor nodes may fail due to unreliable communications in WSNs. The CWQ calculus claims that it can provide a default value to ensure that the nodes can always behave in a reasonable manner even if they are in unreliable communication networks. However, the calculus does not enforce it; that is, we cannot ensure that whether default values can always be available or not. Thus, in this section, we provide an enforcement mechanism, which is a SAT-based analysis for CWQ calculus to check whether or not variables over optional data do indeed contain default data when the expected data is lost. All the interesting points of the program (or network) are described as propositional formulae which characterize the combinations of optional data. Default value would always be required to be available at some critical point in the program, thus we translate the formulae into the certain logical formulae and check whether they are satisfied by a SMT-solver Z3.

Optional data is encoded as a boolean formula, i.e., some($\cdot$) is coded as tt and none is coded as ff. For example, the formula $x_{rep1} \wedge (x_{rep2} \vee x_{rep3})$ represents that both $x_{rep1}$ and either $x_{rep2}$ or $x_{rep3}$ are available; the variables range over booleans.

Table 5: A SAT-Based Analysis Rules

$$\vdash \text{tt}@(n_1[P_1] \parallel ... \parallel n_k[P_k])$$

$$\frac{\vdash \varphi@(n_1[P_1] \parallel ... \parallel n_k[P_k])}{\vdash \varphi@P_1} \quad \cdots\cdots \quad \frac{\vdash \varphi@(n_1[P_1] \parallel ... \parallel n_k[P_k])}{\vdash \varphi@P_k}$$

$$\frac{\vdash \varphi@(\text{case } e \text{ of some}(y) : P_1 \text{ else } P_2) \quad \vdash e \rhd \varphi_e}{\vdash \varphi \wedge \varphi_e@P_1}$$

$$\frac{\vdash \varphi@(\text{case } e \text{ of some}(y) : P_1 \text{ else } P_2) \quad \vdash e \rhd \varphi_e}{\vdash \varphi \wedge \neg\varphi_e@P_2}$$

$$\frac{\vdash \varphi@(c!v.P)}{\vdash \varphi@P} \qquad \vdash x \rhd x \qquad \vdash \text{none} \triangleright \text{ff} \qquad \vdash \text{some}(v) \triangleright \text{tt}$$

$$\vdash c?x \blacktriangleright x \qquad \frac{\vdash \varphi@b.P \quad \vdash b \blacktriangleright \varphi_b}{\vdash \varphi \wedge \varphi_b@P} \qquad \frac{\vdash b_1 \blacktriangleright \varphi_1 \cdots \vdash b_k \blacktriangleright \varphi_k}{\vdash \&_q(b_1, \ldots, b_k) \blacktriangleright [\{q\}](\varphi_1, \ldots, \varphi_k)}$$

### 5.1. The Judgements

The main judgement of our analysis takes the form as follows:

$$\vdash \varphi@N$$

Table 6: Boolean Formulae for the Motivating Example

1: $x_{t_1} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'}) \wedge x_{rep_A} \wedge x_{rep_B} \wedge x_{rep_C}$
2: $x_{t_1} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'}) \wedge x_{rep_A} \wedge x_{rep_B} \wedge (\neg x_{rep_C})$
3: $x_{t_1} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'}) \wedge x_{rep_A} \wedge (\neg x_{rep_B}) \wedge x_{rep_C}$
4: $x_{t_1} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'}) \wedge x_{rep_A} \wedge (\neg x_{rep_B}) \wedge (\neg x_{rep_C})$
5: $x_{t_1} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'}) \wedge (\neg x_{rep_A}) \wedge x_{rep_B} \wedge x_{rep_C}$
6: $x_{t_1} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'}) \wedge (\neg x_{rep_A}) \wedge x_{rep_B} \wedge (\neg x_{rep_C})$
7: $x_{t_1} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'}) \wedge (\neg x_{rep_A}) \wedge (\neg x_{rep_B}) \wedge x_{rep_C}$
8: $x_{t_2} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'}) \wedge x_{rep_A} \wedge x_{rep_B} \wedge x_{rep_C}$
9: $x_{t_2} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'}) \wedge x_{rep_A} \wedge x_{rep_B} \wedge (\neg x_{rep_C})$
10: $x_{t_2} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'}) \wedge x_{rep_A} \wedge (\neg x_{rep_B}) \wedge x_{rep_C}$
11: $x_{t_2} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'}) \wedge x_{rep_A} \wedge (\neg x_{rep_B}) \wedge (\neg x_{rep_C})$
12: $x_{t_2} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'}) \wedge (\neg x_{rep_A}) \wedge x_{rep_B} \wedge x_{rep_C}$
13: $x_{t_2} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'}) \wedge (\neg x_{rep_A}) \wedge x_{rep_B} \wedge (\neg x_{rep_C})$
14: $x_{t_2} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'}) \wedge (\neg x_{rep_A}) \wedge (\neg x_{rep_B}) \wedge x_{rep_C}$
15: $x_{t_2} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'}) \wedge (\neg x_{rep_A}) \wedge (\neg x_{rep_B}) \wedge (\neg x_{rep_C}) \wedge x_{rep'}$
16: $x_{t_2} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'}) \wedge (\neg x_{rep_A}) \wedge (\neg x_{rep_B}) \wedge (\neg x_{rep_C}) \wedge (\neg x_{rep'})$

$N$ stands for the whole program (or network) in our calculus, while the formula $\varphi$ specifies the program point just before $N$. Intuitively, this judgement can be used to describe that under which situation (specified by $\varphi$), the whole program will reach the program point which is immediately before $N$. Here, we do not consider the situation of multiple occurrences of the same subprogram in the system. The semantic interpretation of the judgement is shown below:

$$\begin{array}{l} \text{if} \vdash \varphi @ n_1[P_1] \parallel n_2[P_2] \parallel ... \parallel n_k[P_k] \text{ and} \\ \qquad n_1[P_1] \parallel n_2[P_2] \parallel ... \parallel n_k[P_k] \rightarrow^* n_1[C[P_1\theta_1]] \parallel n_2[C[P_2\theta_2]] \parallel ... \parallel n_k[C[P_k\theta_k]] \\ \text{then } (\overline{\theta_1} \models \varphi) \vee (\overline{\theta_2} \models \varphi) \vee ... \vee (\overline{\theta_k} \models \varphi) \end{array}$$

As we mentioned in the previous section, $\theta$ is the substitution constructed to replace the variables with the receiving values. Thus, $\overline{\theta}$ is the mapping obtained by pointwise application of the encoding $\bar{\cdot}$ and $\overline{\theta} \models \varphi$ denotes the truth of $\varphi$ under the interpretation $\overline{\theta}$.

Besides, we also have two auxiliary judgements for binders and expressions respectively, as follows:

$$\vdash b \blacktriangleright \varphi \quad \text{and} \quad \vdash e \triangleright \varphi$$

The formula $\varphi$ in the former one describes the bindings of the variables that correspond to successful passing the binder $b$ and the semantic interpretation is:

$$\text{if } \vdash b \blacktriangleright \varphi \quad \text{and} \quad b ::_{tt} \theta \quad \text{then} \quad \overline{\theta} \models \varphi$$

The formula $\varphi$ in the later one denotes the result of evaluating the expression $e$ and the semantic interpretation is:

$$\text{if } \vdash e \triangleright \varphi \quad \text{and} \quad e \triangleright v \quad \text{then} \quad \models (\varphi = \overline{v})$$

Table 7: Satisfiability Results of the Motivating Example Using Z3

| | |
|---|---|
| 1: | $[x_{t_1} \mapsto \mathrm{tt}; x_{rep_A} \mapsto \mathrm{tt}; x_{rep_B} \mapsto \mathrm{tt}; x_{rep_C} \mapsto \mathrm{tt}; x_{rep'} \mapsto \mathrm{tt}]$ |
| 2: | $[x_{t_1} \mapsto \mathrm{tt}; x_{rep_A} \mapsto \mathrm{tt}; x_{rep_B} \mapsto \mathrm{tt}; x_{rep_C} \mapsto \mathrm{ff}; x_{rep'} \mapsto \mathrm{tt}]$ |
| 3: | $[x_{t_1} \mapsto \mathrm{tt}; x_{rep_A} \mapsto \mathrm{tt}; x_{rep_B} \mapsto \mathrm{ff}; x_{rep_C} \mapsto \mathrm{tt}; x_{rep'} \mapsto \mathrm{tt}]$ |
| 4: | $[x_{t_1} \mapsto \mathrm{tt}; x_{rep_A} \mapsto \mathrm{tt}; x_{rep_B} \mapsto \mathrm{ff}; x_{rep_C} \mapsto \mathrm{ff}; x_{rep'} \mapsto \mathrm{tt}]$ |
| 5: | $[x_{t_1} \mapsto \mathrm{tt}; x_{rep_A} \mapsto \mathrm{ff}; x_{rep_B} \mapsto \mathrm{tt}; x_{rep_C} \mapsto \mathrm{tt}; x_{rep'} \mapsto \mathrm{tt}]$ |
| 6: | $[x_{t_1} \mapsto \mathrm{tt}; x_{rep_A} \mapsto \mathrm{ff}; x_{rep_B} \mapsto \mathrm{tt}; x_{rep_C} \mapsto \mathrm{ff}; x_{rep'} \mapsto \mathrm{tt}]$ |
| 7: | $[x_{t_1} \mapsto \mathrm{tt}; x_{rep_A} \mapsto \mathrm{ff}; x_{rep_B} \mapsto \mathrm{ff}; x_{rep_C} \mapsto \mathrm{tt}; x_{rep'} \mapsto \mathrm{tt}]$ |
| 8: | $[x_{t_2} \mapsto \mathrm{tt}; x_{rep_A} \mapsto \mathrm{tt}; x_{rep_B} \mapsto \mathrm{tt}; x_{rep_C} \mapsto \mathrm{tt}; x_{rep'} \mapsto \mathrm{tt}]$ |
| 9: | $[x_{t_2} \mapsto \mathrm{tt}; x_{rep_A} \mapsto \mathrm{tt}; x_{rep_B} \mapsto \mathrm{tt}; x_{rep_C} \mapsto \mathrm{ff}; x_{rep'} \mapsto \mathrm{tt}]$ |
| 10: | $[x_{t_2} \mapsto \mathrm{tt}; x_{rep_A} \mapsto \mathrm{tt}; x_{rep_B} \mapsto \mathrm{ff}; x_{rep_C} \mapsto \mathrm{tt}; x_{rep'} \mapsto \mathrm{tt}]$ |
| 11: | $[x_{t_2} \mapsto \mathrm{tt}; x_{rep_A} \mapsto \mathrm{tt}; x_{rep_B} \mapsto \mathrm{ff}; x_{rep_C} \mapsto \mathrm{ff}; x_{rep'} \mapsto \mathrm{tt}]$ |
| 12: | $[x_{t_2} \mapsto \mathrm{tt}; x_{rep_A} \mapsto \mathrm{ff}; x_{rep_B} \mapsto \mathrm{tt}; x_{rep_C} \mapsto \mathrm{tt}; x_{rep'} \mapsto \mathrm{tt}]$ |
| 13: | $[x_{t_2} \mapsto \mathrm{tt}; x_{rep_A} \mapsto \mathrm{ff}; x_{rep_B} \mapsto \mathrm{tt}; x_{rep_C} \mapsto \mathrm{ff}; x_{rep'} \mapsto \mathrm{tt}]$ |
| 14: | $[x_{t_2} \mapsto \mathrm{tt}; x_{rep_A} \mapsto \mathrm{ff}; x_{rep_B} \mapsto \mathrm{ff}; x_{rep_C} \mapsto \mathrm{tt}; x_{rep'} \mapsto \mathrm{tt}]$ |
| 15: | $[x_{t_2} \mapsto \mathrm{tt}; x_{rep_A} \mapsto \mathrm{ff}; x_{rep_B} \mapsto \mathrm{ff}; x_{rep_C} \mapsto \mathrm{ff}; x_{rep'} \mapsto \mathrm{tt}]$ |
| 16: | not satisfied |

Here, we use $\overline{v}$ to stand for the boolean encoding of the value $v$ and we usually use $\varphi_1 = \varphi_2$ as a shorthand for the formula $(\varphi_1 \wedge \varphi_2) \vee (\neg\varphi_1 \wedge \neg\varphi_2)$.

More details about the definition of $\vdash \varphi @ N$ are shown in Table 5, which is given by the inference system. The intuitive meaning of these rules is that if the whole program wants to reach the program point immediately before some processes (e.g., output, case structure), the boolean formula $\varphi$ must be satisfied. And the rules also specify how the boolean formula will change after the whole program executes these processes. The main goal for us to propose these inference rules is that we can easily judge whether some program points will be reached by checking whether the boolean formulae immediately before these program points are satisfied or not. For example, in our motivating case study in Section 4, we do not want the whole program to execute a 0 process, which means that the whole program goes into a deadlock state. Thus, we can easily check whether the boolean formula immediately before the 0 process will be satisfied or not under the given combination of receiving values.

The idea inherits from the Quality Calculus, operating in a top-down manner instead of a more conventional bottom-up manner, and gets started by an axiom $\vdash \mathrm{tt} @ (n_1[P_1] \parallel ... \parallel n_k[P_k])$, saying that the program (or network) is reachable. On the second line, there are two inference rules for parallel composition that if $\varphi$ describes the program point just before the whole network $n_1[P_1] \parallel ... \parallel n_k[P_k]$, then it also describes the program point just before each of the $k$ processes.

Next two rules for the case construct use the auxiliary analysis judgement $\vdash e \rhd \varphi_e$ to get the result of evaluating the expression $e$, and the result information will be put together with $\varphi$ to describe the program point just before the selected branch. The rule for output is very straightforward. The next three inference rules define the judgement $\vdash e \rhd \varphi$ for expressions with the idea we mentioned at the beginning of this section. The rule for binding makes use of another auxiliary analysis judgement $\vdash b \blacktriangleright \varphi_b$. The bindings of the variables that correspond to successful passing the binder $b$ are denoted by information $\varphi_b$, which together with $\varphi$ will describe the program point before $P$. According to the semantics of formula schemes $[\{q\}](\varphi_1, \ldots, \varphi_k)$ in Section 3, last rule encodes the effect of quality predicates $q$.

### *5.2. A SAT-Based Robustness Analysis on Motivating Example*

In this section, based on the analysis rules given in Table 5, we firstly compute the boolean formulae at the program interesting points of the main process $P_{user}$, then we will use the SMT solver Z3 to check whether the formulae are satisfied or not.

In Section 4, we use our calculus to model a real-world case study with the scenario about refueling a car. Taking one important subnetwork *User* into consideration, we identified 16 critical program interesting points with different labels in the process $P_{user}$ inside one network node. Based on the inference rules we give above, we want to compute the analysis results for these points. Starting with $\vdash \mathrm{tt}@n_{11}[P_{user}]$, we get 16 boolean formulae at labels in Table 6.

Here we use the same variable names as what we used in the model. The formula $x_{t_1} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'})$, as well as the one $x_{t_2} \wedge (x_{rep_A} \vee x_{rep_B} \vee x_{rep_C} \vee x_{rep'})$, with the meanings that both time $t_1$ (or time $t_2$) and any receiving variable are reached, refer to the condition for passing the quality binder in the second line and the thirteenth line, respectively. The remainder identifies the condition for reaching the given labels. Then we use Z3 to check whether the program interesting point with the label is reachable or not, which means that whether the corresponding boolean formula is satisfied or not. Satisfiability results of the motivating example given by Z3 are listed in Table 7, which shows that the 0 process in node $n_{11}[P_{user}]$ will never be executed.

## 6. Data-Driven Probabilistic Trust Analysis

As mentioned in related work [11, 35], communication in WSNs will be affected by local conditions, platform characteristics and power consumption constraints, and sensors may be lost during an engagement. Thus, the effectiveness of the system decision making depends on the quality of information it received. Specifically, the system decision of a WSN is expected to be made based on the highest quality data from all network nodes in the WSN, and the decision may have the highest level of trustworthiness if data from all its constituent network nodes are received and considered. However, in reality, there comes the problem that one cannot be sure what data has actually been received in a WSN when making the decision. Consequently, system decisions may differ in trustworthiness depending on which input data have actually been received, and hence it comes the need to analyze the trust of the robustness of a system.

In the literature, Nielson et al. [31] developed a novel probabilistic *trust* analysis for supporting the Quality Calculus [32] to indicate the trust that a user can have in the overall robustness of a system. The trustworthiness of the system decision is determined by the probability that expected input will be absent. However, it is not applicable to the CWQ Calculus for WSNs, because in the CWQ Calculus and WSNs, the trustworthiness of the system decision is not only decided by the probability of absence of expected data, but also based on the trustworthiness of the data itself. In CWQ Calculus, it includes an input guard, *binder*, to specify the inputs to be performed before continuing. It has the flexibility that not all input data in a binder need to be received in order for the process to continue. The subsequent process can determine whether a particular data has actually been received (e.g., by using the case construct case $x$ of some($y$)), and decisions can be made accordingly. In other words, from the perspective of a single network node, its locally stored data may not be sufficient for making the best system decision. In the section below, we will illustrate this characteristic of WSNs in more details through the motivating example of refueling a car by using the information of gas stations stored in base stations.

Thus, in this section, we propose a data-driven probabilistic trust analysis of the CWQ Calculus for WSNs. Instead of only giving the channel a trust value to illustrate the probability of absence of expected data, we also assume that the data received from a channel have a trust value, where the trust value of a data represents the trust of the system decision made solely based on that data. Intuitively, data received from a channel of a network node is of high trustworthiness level if it is essential for making a high-quality decision of the system, and it is not otherwise; for simplicity, we assume the data received from a channel has a probability distribution of trust values.

To facilitate our probabilistic analysis, we change the syntax of binders in CWQ Calculus for WSNs to the form of $\&_q^{\pi}(c_1^{l_1}?x_1, \cdots, c_n^{l_n}?x_n)$, where $\pi \in \mathcal{D}(\{x_1, \cdots, x_n\} \to \{t, \perp\})$ denotes whether an input data $x_i$ is received (i.e., $t$) or not received (i.e., $\perp$) over channel $c_i$ for $1 \leq i \leq n$, and $l_i \in \mathcal{D}(\{L, M, H\} \to \mathbb{R})$ is a probability distribution of the trust of the input data received over channel $c_i$ (i.e., $l_i$ is a probability distribution of the trust of $x_i$). Note that, the available data is classified into low ($L$), medium ($M$) or high ($H$) trust and the trust of a data here can be regarded as the utility of the data in the decision of the entire system. Consequently, we consider data trustworthiness instead of channel trustworthiness, and decouple the probability of receiving input data from the probability of the trustworthiness level of the receiving data, which makes a more flexible probabilistic analysis possible (e.g., for analyzing systems based on WSNs). The overall trustworthiness of the system decision is determined by performing relational analysis to combine the probability distributions of $\pi$ and $l_i (\forall 1 \leq i \leq n)$.

Table 8: Trust Propagation

$$\vdash 1, \pi_\circ, \mathbb{L}_\circ @ (n_1[P_1] \parallel ... \parallel n_k[P_k])$$

$$\frac{\vdash p, \pi, \mathbb{L} @ (n_1[P_1] \parallel ... \parallel n_k[P_k])}{\vdash p, \pi, \mathbb{L} @ P_i} \qquad \forall i \in \{1, \cdots, k\}$$

$$\frac{\vdash p, \pi, \mathbb{L} @ (\text{case } x \text{ of some}(y) : P_1 \text{ else } P_2)}{\vdash p \cdot \pi_{[x \neq \perp]}, (\pi_{\downarrow[x \neq \perp]})|^C_x, \mathbb{L} \oplus l_x[y := x] @ P_1} \qquad \text{if } \pi_{[x \neq \perp]} \neq 0$$

$$\frac{\vdash p, \pi, \mathbb{L} @ (\text{case } x \text{ of some}(y) : P_1 \text{ else } P_2)}{\vdash p \cdot \pi_{[x = \perp]}, (\pi_{\downarrow[x = \perp]})|^C_x, \mathbb{L} @ P_2} \qquad \text{if } \pi_{[x = \perp]} \neq 0$$

$$\frac{\vdash p, \pi, \mathbb{L} @ (c!v.P)}{\vdash p, \pi, \mathbb{L} @ P} \qquad \frac{\vdash p, \pi, \mathbb{L} @ (b.P) \quad \vdash b \blacktriangledown \pi_b}{\vdash p, (\pi|^C_{\text{bv(b)}}) \otimes \pi_b, \mathbb{L} @ P}$$

### *6.1. Trust Propagation*

The judgement of our analysis is of the form $\vdash p, \pi, \mathbb{L} @ P$. Here, $p$ is the probability that we will reach the process $P$, $\pi$ is a distribution from $\mathcal{D}(V \to \{t, \perp\})$ where $V = \{x_1, \cdots, x_{n'}\}$ is a set of optional data variables, and $\mathbb{L} = \{l_1, \ldots, l_{m'}\}$ is a set of distributions of the trust level of data variables $y_i$ for $1 \leq i \leq m'$ (i.e., distributions from $\mathcal{D}(\{L, M, H\} \to \mathbb{R})$. The mappings of $V \to \{t, \perp\}$ indicate whether optional data are received or not and $\pi$ specifies the distribution of these mappings when $P$ is reached. Similarly, $\mathbb{L}$ specifies the distributions of the trust levels of data variables $y$, and we assume $l_i$ and $l_j$ ($i \neq j$) are independent. Note that, this judgement is different from that in related work [31], which is of the form $\vdash p, \pi @ P$.

Through these rules, we can easily get the information about in what probability the whole program will reach the process $P$, what the probability it is that the expected data may be absent when the program reaches $P$ and what the probability it is that the receiving data has some specific trustworthiness levels (i.e., Low, Medium and High) when reaching $P$. The main goal for proposing these rules is that through the analysis, it gives a guide to design a better binder to get a higher trustworthiness of the system decision under a given probability of absence of expected data.

The main judgement is of the form $\vdash 1, \pi_\circ, \mathbb{L}_\circ @ N$ as shown in Table 8, where $N$ stands for the entire program (or network) in the CWQ Calculus, and the choice of $p = 1$ reflects that the main process must be called in order to reach other program points. Here, we let $\pi_\circ = \emptyset$ and $\mathbb{L}_\circ = \emptyset$, since there are no optional data variables or data variables when reaching the main process. Note that, it is also possible to incorporate constants into $\pi$ (and $\pi_\circ$) and $\mathbb{L}$ (and $\mathbb{L}_\circ$); however, we omit the constants in these distributions for ease of presentation, and all constants are assumed to exist and be of the highest trustworthiness.

#### *6.1.1. Operations on $\pi$ and $\mathbb{L}$*

To perform the trust propagation, we need to define several operations on $\pi$ and $\mathbb{L}$. First is *lookup* on a name for $\pi$. That is, given a distribution $\pi : \mathcal{D}(V \to \{t, \perp\})$ and a name $u$, we want to know the probabilities $\pi_{[u \neq \perp]}$ and $\pi_{[u = \perp]}$, corresponding to the probabilities that $u$ is received or not, respectively.

$$\pi_{[u=\perp]} = \textstyle\sum_{(\sigma \in \pi \ s.t. \ \sigma(u) = \perp)} \pi(\sigma)$$

Here, $\sigma$ is a mapping $\sigma : V \to \{t, \perp\}$. $\pi_{[u \neq \perp]}$ is similarly defined, and moreover $\pi_{[u \neq \perp]} = \pi_{[u = t]} = 1 - \pi_{[u = \perp]}$. These operations are used in the analysis of the construct case $x$ of some($y$) : $P_1$ else $P_2$; $\pi_{[x \neq \perp]}$ is the probability that the first branch is taken and $\pi_{[x = \perp]}$ is that the second branch is taken.

The next operation for $\pi$ is *selection* on a name. That is, given a distribution $\pi : \mathcal{D}(V \to \{t, \perp\})$ and a name $u$, we want to construct a new distribution $\pi_{\downarrow[u \neq \perp]}$ that gives 0 probability to all mappings $\sigma$ with $\sigma(u) \neq \perp$ and rescales the remaining probabilities, and this is defined only if $\pi_{[u \neq \perp]} \neq 0$.

$$(\pi_{\downarrow[u\neq\perp]})(\sigma) = \begin{cases} \frac{\pi(\sigma)}{\pi_{[u\neq\perp]}} & \text{if } \sigma(u) \neq \perp, \\ 0 & \text{otherwise.} \end{cases}$$

Similarly, we define $\pi_{\downarrow[u=\perp]}$. These two operations are used in the analysis of the construct case $x$ of some($y$) : $P_1$ else $P_2$; $\pi_{[x\neq\perp]}$ is the distribution of the first branch if it is taken and $\pi_{[x=\perp]}$ is the distribution of the second branch if taken.

The next operation for $\pi$ is *projection* on a subset of names. That is, given a distribution $\pi : \mathcal{D}(V \to \{t, \perp\})$ and a subset of names $U \subseteq V$, we want to obtain the distribution $\pi|_U$ in $\mathcal{D}(U \to \{t, \perp\})$. It is defined as,

$$(\pi|_U)(\sigma) = \textstyle\sum_{(\sigma' \in \pi \ s.t. \ \sigma = \sigma'|_U)} \pi(\sigma').$$

Here, $\sigma'|_U$ is the restriction of the mapping $\sigma' : V \to \{t, \perp\}$ to the domain of $U$; that is $(\sigma'|_U)(u) = \sigma(u)$ if $u \in U$, and $(\sigma'|_U)(u)$ is undefined otherwise. Similarly, we define the projection on the complement of $U$, $\pi|_U^C$, which is the same as $\pi|_{V \setminus U}$. These operations are used to reduce the size of a distribution.

The last operation of $\pi$ is *product* of two distributions. That is, given two distributions $\pi_1 : \mathcal{D}(V_1 \to \{t, \perp\})$ and $\pi_2 : \mathcal{D}(V_2 \to \{t, \perp\})$ over two disjoint sets of names (i.e., $V_1 \cap V_2 = \emptyset$), we construct a new distribution $\pi_1 \otimes \pi_2$ in $\mathcal{D}(V_1 \cup V_2 \to \{t, \perp\})$. It is defined as,

$$(\pi_1 \otimes \pi_2)(\sigma) = \pi_1(\sigma|_{V_1}) \cdot \pi_2(\sigma|_{V_2}).$$

They are used when combining two stochastically independent distributions.

For $\mathbb{L}$, we define two operations, *replacement* and *addition*. Given a distribution $l_x$ in $\mathcal{D}(\{L, M, H\} \to \mathbb{R})$ and a name $y$, the replacement operation $l_x[y := x]$ is to construct another distribution $l_y$ in $\mathcal{D}(\{L, M, H\} \to \mathbb{R})$ with the same probabilities as $l_x$; that is, $l_y(t) = l_x(t), \forall t \in \{L, M, H\}$. That is, the replacement operation is to replace the name of a distribution while all other information remains unchanged. Given a set of distributions $\mathbb{L}$ and a distribution $l_y$, the addition operation $\mathbb{L} \oplus l_y$ is to add $l_y$ into $\mathbb{L}$ (i.e., $\mathbb{L} \cup \{l_y\}$). Both operations are used in the analysis of the construct case $x$ of some($y$) : $P_1$ else $P_2$. That is, when the optional data $x$ is actually received, then the process will continue on the first branch $P_1$; since $y$ instead of $x$ will be visible and used in $P_1$, the distribution of the trust level of $x$ is copied and stored into data $y$ to be prepared for being used in $P_1$.

### *6.1.2. Propagation*

Armed with the above operations on $\pi$ and $\mathbb{L}$, the detailed trust propagation is shown in Table 8. The logic-flow of our analysis is similar to that in the program analysis [30] and in the Quality Calculus [32]. That is, the propagation operates in a top-down manner instead of a more conventional bottom-up manner. As shown in Table 8, our propagation starts from an axiom $\vdash 1, \pi_\circ, \mathbb{L}_\circ @ (n_1[P_1] \parallel ... \parallel n_k[P_k])$ saying that the program (or network) is reachable. The inference rule for parallel composition is presented at the second row; it means that if $p, \pi, \mathbb{L}$ describe the program point just before the entire network $n_1[P_1] \parallel ... \parallel n_k[P_k]$, then they also describe the program point just before each of the $k$ constitute processes.

For the case construct case $x$ of some($y$) : $P_1$ else $P_2$, there are two inference rules as shown at the third and fourth rows. If $\pi_{[x\neq\perp]} \neq 0$, then there is a non-zero probability that the optional data $x$ can be received. Thus, we will continue with process $P_1$ with probability $p \cdot \pi_{[x\neq\perp]}$. Now since we are sure that $x \neq \perp$ (since we reach $P_1$), we need to do a selection on $\pi$ conditioned on the fact that $x \neq \perp$; we can also do a project on the set of names excluding $x$ to simplify the distribution. Moreover, the data $y$ is assigned and may be used in $P_1$; since the trust level of $y$ is the same as the optional data $x$, we construct a new distribution by replacing the $x$ in $l_x$ with $y$, and add the new distribution to $\mathbb{L}$. Note that, the set of distributions $\mathbb{L}$ is used for conducting trust analysis at program points. If $\pi_{x=\perp} \neq 0$, then there is a similar inference rule for continuing with process $P_2$.

The last row illustrates inference rules for output and input, respectively. The rule for output is straightforward, as $p, \pi, \mathbb{L}$ directly pass forward to the following process. The rule for input binding makes use of another auxiliary judgement $\vdash b \blacktriangledown \pi_b$, which obtains the distribution $\pi_b$; note that, $\pi_b$ is computed by using standard statistical inference, based on the probability distributions of all the optional data and channels in $b$. The notation of bv(b) represents the set of bounded variables of $b$. When reaching $P$ (i.e., successfully passing $b$), the distribution $\pi$ will be augmented by $\pi_b$, while $p$ and $\mathbb{L}$ remain the same.

### 6.1.3. Remarks

Note that the probabilistic analysis of CWQ Calculus proposed above is different from that conducted by Nielson et al. for probabilistic trust analysis of the Quality Calculus. Firstly, the CWQ Calculus has a unique characteristic that is not part of the Quality Calculus. Secondly, we decouple the probability of receiving input data from the probability of the trustworthiness level of receiving data. That is, the judgement of our analysis is of the form $\vdash p, \pi, \mathbb{L}@P$ where $\pi$ and $\mathbb{L}$ are distributions with $\pi : \mathcal{D}(V \to \{t, \perp\})$, while the judgement of the analysis in past work [31] is of the form $\vdash p, \pi'@P$ with $\pi' : \mathcal{D}(V' \to \{L, M, H, \perp\})$.

Note that, the set of distributions, $\mathbb{L}$, in our analysis can be also regarded as a distribution as follows. Given $\mathbb{L} = \{l_1, \ldots, l_{m'}\}$ with each $l_i$ being a distribution $l_i : \mathcal{D}(\{L, M, H\} \to \mathbb{R})$, we can construct a new distribution $\mathbb{L}' = L_1 \otimes \cdots \otimes L_{m'}$ where $L_i : \mathcal{D}(x_i \to \{L, M, H\})$ and $\otimes$ is the *product* operation. It is easy to show that $\mathbb{L}'$ is equivalent to $\mathbb{L}$. In this paper, we consider the set of distributions, $\mathbb{L}$, due to its compact form and the independence of $l_i$ and $l_j$ $(i \neq j)$; note that, the size of $\mathbb{L}'$ is much (i.e., exponentially) larger than that of $\mathbb{L}$.

The analysis can be implemented using programming languages, such as Standard ML. Each distribution can be represented as a list of pairs $(\sigma, p)$; for example, the distribution $\pi$ can be represented in the form as shown in Tables 9 and 10. More details and explanations about these two tables will be given in Section 6.3 via the motivating example. Other improvements towards the representation and the probability inference are also possible, we omit the discussions in this paper since it is orthogonal to the content of this paper.

## 6.2. Trust Analysis

Now, we show how to extract information about outputs from the analysis. Firstly, we consider an output of the form $c!v$; that is, we want to compute the trust level of the value $v$ sent over channel $c$. Assume the analysis gives the form $\vdash p, \pi, \mathbb{L}@c!v.P$ when reaching $P$; this means that $P$ is reached with probability $p$, and the distributions of trust levels of data $y$, which may be used in $v$ or $P$, are given in $\mathbb{L}$. The trust level of $v$ over channel $c$ can be represented as a distribution $\phi$ in $\mathcal{D}(\{L, M, H\} \to \mathbb{R})$ and is defined as follows,

$$\phi(t) = \textstyle\sum_{\sigma \in \mathbb{L} \ s.t. \ \sigma(v)=t} \mathbb{L}(\sigma).$$

Note that, for ease of presentation, we assume $\mathbb{L}$ is in the form of a distribution as discussed in above. Thus, if $v$ consists of a single data $y$, then $\phi(y)$ is the same as $l_y \in \mathbb{L}$. Otherwise, $v$ is of the form $f(y_1, \ldots, y_n)$ where $f(\cdot)$ is a function (e.g., $\widehat{\min}()$ in the SAT-based robustness analysis). Given a set of data $\{y_1, \ldots, y_n\}$ with trust levels $\{t_1, \ldots, t_n\}$, respectively, the trust of the function $f(y_1, \ldots, y_n)$ is assume to be the greatest lower bound of $\{t_1, \ldots, t_n\}$ in the trust lattice $\mathcal{L}$, where $\mathcal{L} \overset{df}{=} (\{L, M, H\}, \leq)$. For example, given $y_1, y_2, y_3$ of trust $H, M, L$, respectively, the trust of $\widehat{\min}(y_1, y_2, y_3)$ is $H$; that is, $y_1$ is the most important and sufficient data for the function.

Secondly, we consider all outputs of the form $c!\cdot$; that is, we want to compute the trust level of the decision of the system in the form $c!\cdot$ across all branches of the case constructs. For simplicity, we assume that no occurrence of $c!\cdot$ prefixes another. Then, the distribution $\Phi_c$ in $\mathcal{D}(\{L, M, H, \perp\} \to \mathbb{R})$ is defined as follows,

$$\Phi_c(t) = \textstyle\sum_{\vdash p, \pi, \mathbb{L}@c!v.P} \sum_{\sigma \in \mathbb{L} \ s.t. \ \sigma(v)=t} \mathbb{L}(\sigma).$$

The probability of the trust level of $\perp$ is $\Phi_c(\perp) = 1 - \sum_{t \in \{L,M,H,\perp\}} \Phi_c(t)$.

## 6.3. Probabilistic Trust Analysis on Motivating Example

We illustrate how to compute such a trust of the system decision through two examples in the following. Details about the motivating example have already been given in Section 4. To facilitate our probabilistic analysis, we rewrite the CWQ model of the overall scenario as follows.

There are three channels: *us* is used to communicate (i.e., broadcast and receive) with the base stations; while *local* and *timer* are used to communicate with the local computer and timer, respectively. As we mentioned at the beginning of Section 6, to facilitate our probabilistic analysis, we change the syntax of receiving action in CWQ calculus for WSNs to the form of $c^l?x$, where $l \in \mathcal{D}(\{L, M, H\} \to \mathbb{R})$ is a probability distribution of the trust of the input data received over channel $c$ (i.e., $l$ is a probability distribution of the trust of $x$). Note that, the available data is classified into low ($L$), medium ($M$) or high ($H$) trust and the trust of a data here can be regarded as the utility of the data in the decision of the entire system. For instance, in CWQ calculus, default values (of the same type as the required ones)

Table 9: $\pi$ for binder 1

| id | $x_{t_1}$ | $x_{rep_A}$ | $x_{rep_B}$ | $x_{rep_C}$ | $x_{rep'}$ | $p$ |
|---|---|---|---|---|---|---|
| $\pi_1$ | $t$ | $t$ | $t$ | $t$ | $t$ | 0.1750 |
| $\pi_2$ | $t$ | $t$ | $t$ | $t$ | $\perp$ | 0.0438 |
| $\pi_3$ | $t$ | $t$ | $t$ | $\perp$ | $t$ | 0.1167 |
| $\pi_4$ | $t$ | $t$ | $t$ | $\perp$ | $\perp$ | 0.0292 |
| $\pi_5$ | $t$ | $t$ | $\perp$ | $t$ | $t$ | 0.1167 |
| $\pi_6$ | $t$ | $t$ | $\perp$ | $t$ | $\perp$ | 0.0292 |
| $\pi_7$ | $t$ | $\perp$ | $t$ | $t$ | $t$ | 0.1167 |
| $\pi_8$ | $t$ | $\perp$ | $t$ | $t$ | $\perp$ | 0.0292 |
| $\pi_9$ | $t$ | $t$ | $\perp$ | $\perp$ | $t$ | 0.0778 |
| $\pi_{10}$ | $t$ | $t$ | $\perp$ | $\perp$ | $\perp$ | 0.0196 |
| $\pi_{11}$ | $t$ | $\perp$ | $t$ | $\perp$ | $t$ | 0.0778 |
| $\pi_{12}$ | $t$ | $\perp$ | $t$ | $\perp$ | $\perp$ | 0.0196 |
| $\pi_{13}$ | $t$ | $\perp$ | $\perp$ | $t$ | $t$ | 0.0778 |
| $\pi_{14}$ | $t$ | $\perp$ | $\perp$ | $t$ | $\perp$ | 0.0196 |
| $\pi_{15}$ | $t$ | $\perp$ | $\perp$ | $\perp$ | $t$ | 0.0513 |

Table 10: $\pi$ for binder 2

| id | $x_{t_1}$ | $x_{rep_A}$ | $x_{rep_B}$ | $x_{rep_C}$ | $x_{rep'}$ | $p$ |
|---|---|---|---|---|---|---|
| $\pi_1$ | $t$ | $t$ | $t$ | $t$ | $t$ | 0.1860 |
| $\pi_2$ | $t$ | $t$ | $t$ | $t$ | $\perp$ | 0.0465 |
| $\pi_3$ | $t$ | $t$ | $t$ | $\perp$ | $t$ | 0.1245 |
| $\pi_4$ | $t$ | $t$ | $t$ | $\perp$ | $\perp$ | 0.0315 |
| $\pi_5$ | $t$ | $t$ | $\perp$ | $t$ | $t$ | 0.1245 |
| $\pi_6$ | $t$ | $t$ | $\perp$ | $t$ | $\perp$ | 0.0315 |
| $\pi_7$ | $t$ | $\perp$ | $t$ | $t$ | $t$ | 0.1245 |
| $\pi_8$ | $t$ | $\perp$ | $t$ | $t$ | $\perp$ | 0.0315 |
| $\pi_9$ | $t$ | $t$ | $\perp$ | $\perp$ | $t$ | 0.0827 |
| $\pi_{10}$ | $t$ | $t$ | $\perp$ | $\perp$ | $\perp$ | 0 |
| $\pi_{11}$ | $t$ | $\perp$ | $t$ | $\perp$ | $t$ | 0.0827 |
| $\pi_{12}$ | $t$ | $\perp$ | $t$ | $\perp$ | $\perp$ | 0 |
| $\pi_{13}$ | $t$ | $\perp$ | $\perp$ | $t$ | $t$ | 0.0827 |
| $\pi_{14}$ | $t$ | $\perp$ | $\perp$ | $t$ | $\perp$ | 0 |
| $\pi_{15}$ | $t$ | $\perp$ | $\perp$ | $\perp$ | $t$ | 0.0514 |

are given to deal with the situations that the ideal behavior of a sensor node fails due to unreliable communications. However, default values are not as useful as the expected data, and the final decision of the system may have a low trustworthiness depending on the default value. Thus, $L$ is marked on channel *local*. Similarly, the trustworthiness level of the data via channel *us* may be H(igh) or M(edium), whereas the trustworthiness level of the data via channel *timer* must be H(igh).

$$\begin{aligned}
Network &\overset{df}{=} User \parallel (\prod_{i\in\mathbb{Z}} BS_i \parallel \prod_{k\in\mathbb{Z}} GS_k)\\
User &\overset{df}{=} n_{11}[P_{user}] \parallel n_{12}[Local_{user}] \parallel n_{13}[Timer_{user}]\\
BS_i &\overset{df}{=} n_{2i}[P_{bs}]\\
GS_k &\overset{df}{=} n_{3k}[P_{gs}]\\
&\text{Using } us^{H\vee M}, local^{L}, timer^{H}.
\end{aligned}$$

Also, we give details of the main process $P_{user}$ in Figure 4, which is the main subject of our probabilistic *trust* analysis, and omit details of other processes.

**Assumption.** Before we discuss the different design of binders, we give the assumptions we will use as follows.

- In $P_{user}$ in Figure 4, for ease of exposition we assume that there are three BSs (i.e., BSs $A$, $B$, and $C$ in Figure 1) that can communicate with the user.
- $l_{rep_A}$ is a probability distribution in $\mathcal{D}(\{L, M, H\} \to \mathbb{R})$ for variable $x_{rep_A}$ and let us assume that $l_{rep_A}(H) = l_{rep_A}(M) = 0.5$. $l_{rep_B}$ and $l_{rep_C}$ are similarly defined as $l_{rep_A}$, while $l_{t_1}$ and $l_{rep'}$ have deterministic trust $H$ and $L$, respectively.
- For presentation simplicity, we assume that $l_i, l_j$ $(i \neq j)$ are independent.
- The process of receiving input data through channels *us* and *local* are exponentially distributed with rates $\lambda_{us}$ and $\lambda_{local}$, respectively. For ease of presentation, let us assume the probability of receiving replying messages

through channels *us* and *local* are $p_{us} = 0.6$ and $p_{local} = 0.8$, respectively; that is, the probability of not receiving replying messages through channels *us* and *local* are $1 - p_{us} = 0.4$ and $1 - p_{local} = 0.2$, respectively.

- For presentation simplicity, we assume that there is only one timeout (i.e., $t_1$) in $P_{user}$ in Figure 4; that is, only labels, $1, \ldots, 6, 16$, are reachable, while those branches corresponding to labels $7, \ldots, 15$ are ignored.

```
$P_{user} \overset{df}{=}$ $us^{H\vee M}!req.local^{L}!req.timer^{H}!(t_1, t_2).$
    $\&_{\forall}(timer^{H}?x_{t_1}, \&_{\exists}(us^{H\vee M}?x_{rep_A}, us^{H\vee M}?x_{rep_B}, us^{H\vee M}?x_{rep_C}, local^{L}?x_{rep'})).$
    case $x_{rep_A}$ of some($y_{rep_A}$) :
        case $x_{rep_B}$ of some($y_{rep_B}$) :
            case $x_{rep_C}$ of some($y_{rep_C}$) :$^1$ use($\widehat{\min}(y_{rep_A}, y_{rep_B}, y_{rep_C})$)
            else$^2$use($\widehat{\min}(y_{rep_A}, y_{rep_B})$)
        else
            case $x_{rep_C}$ of some($y_{rep_C}$) :$^3$ use($\widehat{\min}(y_{rep_A}, y_{rep_C})$)
            else$^4$ use($y_{rep_A}$)
    else
        case $x_{rep_B}$ of some($y_{rep_B}$) :
            case $x_{rep_C}$ of some($y_{rep_C}$) :$^5$ use($\widehat{\min}(y_{rep_B}, y_{rep_C})$)
            else$^6$ use($y_{rep_B}$)
        else
            case $x_{rep_C}$ of some($y_{rep_C}$) :$^7$ use($y_{rep_C}$)
            else
                $\&_{\forall}(timer^{H}?x_{t_2}, \&_{\exists}(us^{H\vee M}?x_{rep_A}, us^{H\vee M}?x_{rep_B}, us^{H\vee M}?x_{rep_C}, local^{L}?x_{rep'})).$
                case $x_{rep_A}$ of some($y_{rep_A}$) :
                    case $x_{rep_B}$ of some($y_{rep_B}$) :
                        case $x_{rep_C}$ of some($y_{rep_C}$) :$^8$ use($\widehat{\min}(y_{rep_A}, y_{rep_B}, y_{rep_C})$)
                        else$^9$use($\widehat{\min}(y_{rep_A}, y_{rep_B})$)
                    else
                        case $x_{rep_C}$ of some($y_{rep_C}$) :$^{10}$ use($\widehat{\min}(y_{rep_A}, y_{rep_C})$)
                        else$^{11}$ use($y_{rep_A}$)
                else
                    case $x_{rep_B}$ of some($y_{rep_B}$) :
                        case $x_{rep_C}$ of some($y_{rep_C}$) :$^{12}$ use($\widehat{\min}(y_{rep_B}, y_{rep_C})$)
                        else$^{13}$ use($y_{rep_B}$)
                    else
                        case $x_{rep_C}$ of some($y_{rep_C}$) :$^{14}$ use($y_{rep_C}$)
                        else
                            case $x_{rep'}$ of some($y_{rep'}$) :$^{15}$ use($y_{rep'}$)
                            else$^{16}$ 0
```

Figure 4: Model of the process $P_{user}$ with trust

**Discussion.** In order to investigate how the design of the binder will affect the final decision of the system, we will take two equivalent binders with different forms into consideration, and use our probabilistic trust analysis approach to see what the trustworthiness of the final decision is depending on these two binders respectively.

The binder of the second line in Figure 4, is shown as follows:

$$\&_{\forall}(timer^{H}?x_{t_1}, \&_{\exists}(us^{H\vee M}?x_{rep_A}, us^{H\vee M}?x_{rep_B}, us^{H\vee M}?x_{rep_C}, local^{L}?x_{rep'})).$$

According to the semantics of the quality predicate, formally, $\&_{\exists}(x_1, \ldots, x_n)$ is equivalent to $x_1 \vee \cdots \vee x_n$ and $\&_{\forall}(x_1, \ldots, x_n)$ is equivalent to $x_1 \wedge \cdots \wedge x_n$. Here, we also allow to write the quality predicate as $\&_{[1\wedge(2\vee 3)]}(x_1, x_2, x_3)$ which is equivalent to $x_1 \wedge (x_2 \vee x_3)$. Thus, we rewrite this binder and denote it as binder 1,

$$\&_{[1\wedge(2\vee3\vee4\vee5)]}(timer^{H}?x_{t_1}, us^{H\vee M}?x_{rep_A}, us^{H\vee M}?x_{rep_B}, us^{H\vee M}?x_{rep_C}, local^{L}?x_{rep'}).$$

That is, when $t_1$ time units are reached, the process continues when the input from either BSs $A$, or $B$, or $C$, or the local computer is performed. Alternatively, we might use the equivalent binder, denoted as binder 2,

$$\&_{[1\wedge((2\wedge3)\vee(2\wedge4)\vee(3\wedge4)\vee5)]}(timer^{H}?x_{t_1}, us^{H\vee M}?x_{rep_A}, us^{H\vee M}?x_{rep_B}, us^{H\vee M}?x_{rep_C}, local^{L}?x_{rep'}).$$

This binder requires that at least two BSs from $\{A, B, C\}$ must reply before the process can proceed when $t_1$ time units are reached. We will show through probabilistic trust analysis in Section 6.3 that the probability of high trustworthiness of the final decision depending on binder 2 is better than the one based on binder 1 as far as the quality of the GS (i.e., how close is it to the user), obtained by the system, is concerned. More details are given below:

**Example 1.** We take the binder, binder 1, into consideration,

$$\&^{\pi}_{[1\wedge(2\vee3\vee4\vee5)]}(timer^{H}?x_{t_1}, us^{H\vee M}?x_{rep_A}, us^{H\vee M}?x_{rep_B}, us^{H\vee M}?x_{rep_C}, local^{L}?x_{rep'}).$$

Here, the trust $l_{rep_A}$ of $x_{rep_A}$ received over channel $us$ may be either $H$ or $M$ (i.e., $H \vee M$). One can show that the distribution $\pi$, indicating whether input data are received or not, is computed as that in Table 9, with $\pi(\delta) = 0$ for all other cases.

Now, we illustrate how to obtain the trust of the decision of the system. First, let us consider $\pi_1(x_{t_1} \mapsto t, x_{rep_A} \mapsto t, x_{rep_B} \mapsto t, x_{rep_C} \mapsto t, x_{rep'} \mapsto t) = 0.1750$ in Table 9. Since all optional data, $x_{rep_A}$, $x_{rep_B}$ and $x_{rep_C}$, have actually been received, the decision of the system is made based on the combination of these three data; note that, $\pi_1$ and $\pi_2$ in Table 9 together correspond to label 1 in Figure 4. Thus, when reaching label 1, the trust of the decision is $M$ with probability $l_{rep_A}(M) \times l_{rep_B}(M) \times l_{rep_C}(M) = 0.125$, and it is $H$ with probability $1 - 0.125 = 0.875$; recall that each $l_x \in \mathcal{D}(\{L, M, H\} \to \mathbb{R})$ is a probability distribution. Based on the above, we can see that the probability that the trust of the decision is $H$ includes $0.1750\times0.875$, and the probability to be $M$ includes $0.1750\times0.125$. Similarly, $\pi_3$ and $\pi_4$ in Table 9 together correspond to label 2 in Figure 4, and $\pi_3 = 0.1167$. When reaching label 2, the trust of the decision is $M$ with probability $l_{rep_A}(M)\times l_{rep_B}(M) = 0.25$, and it is $H$ with probability $1-0.25 = 0.75$. Thus, the probability that the trust of the decision is $H$ also includes another $0.1167 \times 0.75$, and the probability to be $M$ also includes another $0.1167 \times 0.25$.

Overall, the probability that the trust of the decision is $H$ is $(\pi_1+\pi_2)\times0.875+(\pi_3+\cdots+\pi_8)\times0.75+(\pi_9+\cdots+\pi_{14})\times 0.5 = 0.2188\times0.875+0.4377\times0.75+0.2922\times0.5 = 0.6658$, and the probability that the trust of the decision is $M$ is $(\pi_1+\pi_2)\times0.125+(\pi_3+\cdots+\pi_8)\times0.25+(\pi_9+\cdots+\pi_{14})\times0.5 = 0.2188\times0.125+0.4377\times0.25+0.2922\times0.5 = 0.2829$.

**Example 2.** Now, we consider binder 2 which is discussed in the case study, as follows,

$$\&_{[1\wedge((2\wedge3)\vee(2\wedge4)\vee(3\wedge4)\vee5)]}(timer^{H}?x_{t_1}, us^{H\vee M}?x_{rep_A}, us^{H\vee M}?x_{rep_B}, us^{H\vee M}?x_{rep_C}, local^{L}?x_{rep'}).$$

Here, $l_{rep_A}$, $l_{rep_B}$, $l_{rep_C}$, and $l_{rep'}$ are the same as in Example 1 in above. Note that, the process $P_{user}$ needs to be modified accordingly; we omit the details here. Similar to Example 1, one can show that the distribution $\pi$, indicating whether input data are received or not, is computed as that in Table 10.

We illustrate how to obtain the trust of the decision of the system. First, let us consider $\pi_1(x_{t_1} \mapsto t, x_{rep_A} \mapsto t, x_{rep_B} \mapsto t, x_{rep_C} \mapsto t, x_{rep'} \mapsto t) = 0.1860$ in Table 10. Since all optional data, $x_{rep_A}$, $x_{rep_B}$ and $x_{rep_C}$, have actually been received, the decision of the system is made based on the combination of these three data. Thus, similar to that in Example 1, the probability that the trust of the decision is $H$ includes $0.1860 \times 0.875$, and the probability to be $M$ includes $0.1860 \times 0.125$. Similarly, when considering $\pi_3 = 0.1245$, the probability that the trust of the decision is $H$ also includes another $0.1245 \times 0.75$, and the probability to be $M$ also includes another $0.1245 \times 0.25$. Overall, the probability that the trust of the decision is $H$ is 0.6785, and the probability that it is $M$ is 0.2701.

**Remark.** By comparing the above two examples, we can see that the probability of the trust of the decision based on binder 2 to be $H$ is 0.6785 and it is larger than the probability of the trust of the decision based on binder 1 to be $H$ which is 0.6658. Thus, by using probabilistic analysis we can quantify the trustworthiness of decisions based on different binders, based on which we choose the better one.

## 7. Related Work

**Non-Process-Algebraic Methods for Quality in Networks.** In reality, communications in wireless networks are often unreliable, which may be caused by deployment constraints (e.g., bad environment and time delay) and/or communication modalities (e.g., nodes switch between an active (on) and a sleeping (off) mode, for saving energy); this may result in abnormalities and thus decrease the quality of service provided by the system. There are a number of efforts focusing on the communication quality of networks in general. Zhao et al. [43, 44] investigate the fundamental relationship between node density and transmission delay in large-scale wireless ad hoc networks with unreliable links from percolation perspective. They try to answer the question, how transmission delay varies in accordance with node density, to provide a guidance for determining the number of nodes to meet the delay requirement when designing a network. Related work [7, 17] show the relationship between transmission delay and connectivity between nodes with dynamic links in wireless networks. Kong et al. [16] also study the connectivity and transmission latency in wireless networks with unreliable links from a percolation-based perspective. Each link of the network is active with some probability depending on the distance between two nodes in a static model. Whereas, the link is active or inactive according to a Markov on-off process in a dynamic model, and a transmission delay exists because of the dynamic behaviors of the links.

We are inspired by these work for our further investigations in our future work, but we find that none of these studies has built a general framework of the communication quality for wireless networks. Therefore, our framework aims to capture the features of WSNs and observe the communication quality of wireless networks from a process algebra perspective, which is more abstract and allows formal modeling and analysis of the networks. Process algebra, as a very important part of the computer science theory, has already been widely used in many fields. It is also commonly accepted in the formal methods community that the modeling and analysis of wireless systems in design can reduce implementation errors considerably [1, 39].

**Process Algebra on Networks.** On the process algebra side, the process calculi CCS (a Calculus of Communicating Systems) [27], CSP (Communicating Sequential Processes) [14] and $\pi$ calculus [28] are usually used to describe the point-to-point communications in reactive system. However, they cannot specify the broadcast communication behavior, which is one of the most important features of WSNs. Thus, the calculus CBS (a Calculus of Broadcasting Systems) [33] and b$\pi$ [8] are presented, which can specify the broadcast communications.

Both CBS and b$\pi$ describe the global broadcast communication, which means that the calculi assume that all the receivers inside the network can receive the message broadcasted by the sender. In reality, the communication in WSNs is local broadcast, which represents that only nodes within the transmission area of the sender can receive the broadcast message. To capture the feature of local broadcast, a set of process calculi, AWN [9], CMAN [12], CMN [23], TCWS [24], CBS$\sharp$ [29], CWS [26], the $\omega$-calculus [36], RBPT [10, 13], are presented as process algebraic modeling languages for wireless networks. In these models, a message that is sent by a node could only be received by the nodes "inside the transmission range" and the neighborhood relationship is defined in different ways (e.g., define it either as a part of the syntax or as a part of the semantics). Nevertheless, none of the above models consider the service quality provided by the wireless system.

**Quality Calculus.** Taking service quality of wireless systems into consideration, it is of significant importance to ensure that wireless sensor nodes can always behave in a reasonable manner even though they are in an unreliable communication network. As a first step in this direction, Nielson et al. proposed the Quality Calculus [32, 37], which enforces robustness of software components in an open and error prone environment, specifies the behavior when communication links break down, and also considers global broadcast communications. However, their work focus on the process level and they want to do some further researches, such as "enriching the framework with a notion of network topology and spatially-bounded broadcast" in their future work.

As the combination of communication quality and features of WSNs (e.g., local broadcast, node mobility) is still a challenging topic, to be taken one step further, the CWQ calculus [41] was recently proposed for modeling and reasoning about WSNs and applications based on WSNs. The CWQ calculus combines the wireless local broadcast together with quality predicates, and extracts the topological structure of the network as a parametric configuration. Besides, partial functions are given to illustrate the different the different transmission areas of different nodes using different channels. Default values (of the same type as the required ones) are given to deal with the situations that the ideal behavior of a sensor node fails due to unreliable communications. Moreover, we develop a SAT-based analysis approach to support the CWQ calculus for WSNs to check whether the whole network will reach some error configurations or not.

**Probabilistic Trust Analysis.** Nielson et al. [31] developed a novel probabilistic *trust* analysis for supporting the Quality Calculus to indicate the trust that a user can have in the overall robustness of a system. They assume each channel has a trust value and change the syntax of binders into $\&_q^{\pi}(c_1^{l_1}?x_1, \cdots, c_n^{l_n}?x_n)$. Here, $\pi \in \mathcal{D}(\{x_1, \cdots, x_n\} \to \mathcal{L}_\perp)$ is a probability distribution indicating the probability of the various inputs having been received where $\perp$ denotes the absence of input and $\mathcal{L}_\perp$ is the lifted trust lattice obtained from $\mathcal{L}$ by adding $\perp$ as the new least element. Then, they use the information about the probabilities that expected input will be absent to associate probability distributions with all program points of interest, where the probabilities indicate the trust level of the data.

The trust analysis performed for Quality Calculus, in related work [31], however is not applicable to the CWQ Calculus for WSNs. The reason is as follows: related work [11, 35] declare that, the effectiveness of the decision of WSNs making depends on the quality of available information, which means that the quality of the data may affect the system decision as well. Specifically, local stored data inside one single node may not be sufficient for making the best system decision. Thus, a better system decision of a WSN is expected to be made based on the data with the highest level of trustworthiness if data from all its constituent network nodes are received and considered.

Thus, in this paper, we propose a new data-driven probabilistic trust analysis of the CWQ calculus for WSNs. In our analysis, we also take data trustworthiness into account, and decouple the probability of receiving input data from the probability of the trustworthiness level of receiving data, which makes a more flexible probabilistic analysis possible (e.g., for analyzing systems based on WSNs). The overall trustworthiness of the system decision is determined by performing relational analysis to combine these two probability distributions.

## 8. Conclusion and Future Work

In this paper, we extended the CWQ calculus by modifying and simplifying it to be a parametric framework to make it more flexible for modeling and reasoning about networks of different topological structures. In our new framework, we used an undirected graph to describe the topology of the entire network as a configuration, such that all topological structure changes can be easily captured by this parametric framework. We used two partial functions to illustrate the different transmission areas of different nodes when using different channels. Moreover, we proposed two formal analysis approaches for CWQ calculus based on the theory of static analysis:

1) We developed a SAT-based robustness analysis for supporting the calculus to check the vulnerability of the whole network. We formulated all the interesting program points as propositional formulae, and used an existing efficient SMT solver Z3 to check the satisfiability of the formulae which then determines whether the corresponding program point can be reached or not.

2) We proposed a data-driven probabilistic trust analysis to decouple the probability of receiving data from the probability of the trustworthiness of receiving data, which makes a more flexible probabilistic analysis possible (e.g., for analyzing systems based on WSNs). The overall trustworthiness of the system decision is then determined by performing a relational analysis to combine these probability distributions.

Finally, we gave a real-world case study with the scenario of refueling a car to illustrate the applicability of the extended calculus and these two analysis approaches. Several interesting topics are discussed below.

- **Correctness and Soundness.** In supporting to analyse the robustness and trustworthiness of CWQ calculus and WSNs, we propose two formal analysis approaches in Section 5 and Section 6, respectively. The transitions in these analysis rules are given based on the semantic rules shown in Section 3. According to the theories in type system [3, 38], the correspondence between these inference rules and semantic rules is left to be proved. Besides, under the assistance of the theorem provers (e.g., Maude [4, 25] and Coq [2]), the correctness and soundness of these inference rules may also be proved in the future.

- **Probability Density Function for the Distance.** Taking our probabilistic trust analysis into consideration, it will be a very interesting topic if the probability distribution for receiving data depends on the distance between two nodes, which means that the $\pi(x)$ will not be a single probability, but a probability density function for the distance. We are inspired from the work [16, 43] that we can also learn from the percolation theory to investigate the relationship between the probability distribution of receiving data and the distance.

In the future, we are continuing to explore more on the reasoning of the wireless networks based on the formal semantics [15], including the denotational semantics and algebraic semantics. Linking theories between these semantics and a deduction system of the calculus may also be interesting topics. Furthermore, we want to take the mobility into consideration to form the mCWQ calculus (CWQ calculus with Mobility) and introduce the probability into the syntax to form the pCWQ calculus (CWQ calculus with Probability) to capture more features of WSNs. It is also possible to use PRISM [18] for automatic probabilistic analysis. Besides, we will consider to give an explicit inference rule to deal with the analysis of recursion in the robustness analysis approach and also add a name restriction into the calculus.

**Acknowledgement.** This work was partly supported by the Danish National Research Foundation and the National Natural Science Foundation of China (Grant No. 61361136002) for the Danish-Chinese Center for Cyber Physical Systems. It was also supported by National Natural Science Foundation of China (Grant No. 61321064) and Shanghai Collaborative Innovation Center of Trustworthy Software for Internet of Things (No. ZF1213).

## References

[1] J. Abrial. Formal Methods: Theory Becoming Practice. *J. UCS*, 13(5):619–628, 2007.

[2] Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

[3] L. Cardelli. Type Systems. In A. B. Tucker, editor, *The Computer Science and Engineering Handbook*, pages 2208–2236. CRC Press, 1997.

[4] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. F. Quesada. Maude: Specification and Programming in Rewriting Logic. *Theor. Comput. Sci.*, 285(2):187–243, 2002.

[5] L. M. de Moura and N. Bjørner. Z3: An Efficient SMT Solver. In *Proc. 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, (ETAPS 2008), Budapest, Hungary, March 29–April 6, 2008*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.

[6] L. M. de Moura and N. Bjørner. Satisfiability Modulo Theories: Introduction and Applications. *Commun. ACM*, 54(9):69–77, 2011.

[7] O. Dousse, P. Mannersalo, and P. Thiran. Latency of wireless sensor networks with uncoordinated power saving mechanisms. In *Proc. 5th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, (MobiHoc 2004), Roppongi Hills, Tokyo, Japan, May 24–26, 2004*, pages 109–120. ACM, 2004.

[8] C. Ene and T. Muntean. A Broadcast-based Calculus for Communicating Systems. In *Proc. 15th International Parallel & Distributed Processing Symposium (IPDPS 2001), San Francisco, CA, April 23–27, 2001*, page 149. IEEE Computer Society, 2001.

[9] A. Fehnker, R. J. van Glabbeek, P. Höfner, A. McIver, M. Portmann, and W. L. Tan. A Process Algebra for Wireless Mesh Networks. In *Proc. 21st European Symposium on Programming Languages and Systems (ESOP 2012)*, volume 7211 of *Lecture Notes in Computer Science*, pages 295–315, Tallinn, Estonia, March 2012. Springer.

[10] F. Ghassemi, W. Fokkink, and A. Movaghar. Restricted Broadcast Process Theory. In *Proc. 6th IEEE International Conference on Software Engineering and Formal Methods, (SEFM 2008), Cape Town, South Africa, 10–14 November 2008*, pages 345–354. IEEE Computer Society, 2008.

[11] D. Gillies, D. J. Thornley, and C. Bisdikian. Probabilistic Approaches to Estimating the Quality of Information in Military Sensor Networks. *Comput. J.*, 53(5):493–502, 2010.

[12] J. C. Godskesen. A Calculus for Mobile Ad Hoc Networks. In *Proc. 9th International Conference on Coordination Models and Languages, (COORDINATION 2007), Paphos, Cyprus, June 6–8, 2007*, volume 4467 of *Lecture Notes in Computer Science*, pages 132–150. Springer, 2007.

[13] J. C. Godskesen and S. Nanz. Mobility Models and Behavioural Equivalence for Wireless Networks. In *Proc. 11th International Conference on Coordination Models and Languages, (COORDINATION 2009), Lisboa, Portugal, June 9–12, 2009*, volume 5521 of *Lecture Notes in Computer Science*, pages 106–122. Springer, 2009.

[14] C. A. R. Hoare. Communicating Sequential Processes. *Communications of the ACM*, 21(8):666–677, AUG 1978.

[15] Y. Huang, J. He, H. Zhu, Y. Zhao, J. Shi, and S. Qin. Semantic Theories of Programs with Nested Interrupts. *Frontiers of Computer Science*, 9(3):331–345, 2015.

[16] Z. Kong and E. M. Yeh. Connectivity and Latency in Large-Scale Wireless Networks with Unreliable Links. In *Proc. 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, (INFOCOM 2008), 13–18 April 2008, Phoenix, AZ, USA*, pages 11–15. IEEE, 2008.

[17] Z. Kong and E. M. Yeh. On the Latency for Information Dissemination in Mobile Wireless Networks. In *Proc. 9th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, (MobiHoc 2008), Hong Kong, China, May 26–30, 2008*, pages 139–148. ACM, 2008.

[18] M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *Proc. 23rd International Conference on Computer Aided Verification, (CAV 2011)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[19] I. Lanese and D. Sangiorgi. An Operational Semantics for a Calculus for Wireless Systems. *Theor. Comput. Sci.*, 411(19):1928–1948, 2010.

[20] E. A. Lee. Architectural Support for Cyber-Physical Systems. In *Proc. 12th International Conference on Architectural Support for Programming Languages and Operating Systems, (ASPLOS 2015), Istanbul, Turkey, March 14–18, 2015*, page 1. ACM, 2015.

[21] S. Liu, Y. Zhao, H. Zhu, and Q. Li. A Calculus for Mobile Ad Hoc Networks from a Group Probabilistic Perspective. In *Proc. 13th IEEE International Symposium on High-Assurance Systems Engineering (HASE 2011)*, pages 157–162. IEEE Computer Society, 2011.

[22] S. Malik and L. Zhang. Boolean Satisfiability from Theoretical Hardness to Practical Success. *Commun. ACM*, 52(8):76–82, 2009.

[23] M. Merro. An Observational Theory for Mobile Ad Hoc Networks (full version). *Inf. Comput.*, 207(2):194–208, 2009.

[24] M. Merro and E. Sibilio. A Timed Calculus for Wireless Systems. In *Proc. 3rd IPM International Conference on Fundamentals of Software Engineering, (FSEN 2009)*, volume 5961 of *Lecture Notes in Computer Science*, pages 228–243. Springer, 2009.

[25] J. Meseguer. Maude. In D. A. Padua, editor, *Encyclopedia of Parallel Computing*, pages 1095–1102. Springer, 2011.

[26] N. Mezzetti and D. Sangiorgi. Towards a Calculus For Wireless Systems. *Electr. Notes Theor. Comput. Sci.*, 158:331–353, 2006.

[27] R. Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[28] R. Milner. *Communicating and Mobile Systems: The PI-Calculus*. Cambridge University Press, New York, NY, USA, 1999.

[29] S. Nanz and C. Hankin. Formal Security Analysis for Ad-Hoc Networks. *Electronic Notes in Theoretical Computer Science*, 142:195–213, 2006.

[30] F. Nielson, H. R. Nielson, and C. Hankin. *Principles of Program Analysis*. Springer, 1999.

[31] H. R. Nielson and F. Nielson. Probabilistic Analysis of the Quality Calculus. In *Proc. Joint IFIP WG 6.1 International Conference on Formal Techniques for Distributed Systems, (FMOODS/FORTE 2013), Held as Part of the 8th International Federated Conference on Distributed Computing Techniques (DisCoTec 2013), Florence, Italy, June 3–5, 2013*, volume 7892 of *Lecture Notes in Computer Science*, pages 258–272. Springer, 2013.

[32] H. R. Nielson, F. Nielson, and R. Vigo. A Calculus for Quality. In *Proc. 9th International Symposium on Formal Aspects of Component Software, (FACS 2012), Mountain View, CA, USA, September 12–14, 2012. Revised Selected Papers*, volume 7684 of *Lecture Notes in Computer Science*, pages 188–204. Springer, 2012.

[33] K. Prasad. A Calculus of Broadcasting Systems. *Science of Computer Programming*, 25(2–3):285–327, 1995.

[34] K. Prasad. A Prospectus for Mobile Broadcasting Systems. *Electr. Notes Theor. Comput. Sci.*, 162:295–300, 2006.

[35] V. Sachidananda. Quality of Information in Wireless Sensor networks. In *Proc. the Joint Workshop of the German Research Training Groups in Computer Science, Algorithmic synthesis of reactive and discrete-continuous systems, (AlgoSyn 2010), May 31–June 2, 2010*, page 115. Verlagshaus Mainz, Aachen, Germany, 2010.

[36] A. Singh, C. R. Ramakrishnan, and S. A. Smolka. A Process Calculus for Mobile Ad Hoc Networks. *Sci. Comput. Program.*, 75(6):440–469, 2010.

[37] R. Vigo, F. Nielson, and H. R. Nielson. Broadcast, Denial-of-Service, and Secure Communication. In *Proc. 10th International Conference on Integrated Formal Methods (IFM 2013), Turku, Finland, June 10–14, 2013*, volume 7940 of *Lecture Notes in Computer Science*, pages 412–427. Springer, June 2013.

[38] D. M. Volpano, C. E. Irvine, and G. Smith. A Sound Type System for Secure Flow Analysis. *Journal of Computer Security*, 4(2/3):167–188, 1996.

[39] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. S. Fitzgerald. Formal Methods: Practice and Experience. *ACM Comput. Surv.*, 41(4), 2009.

[40] X. Wu, H. R. Nielson, and H. Zhu. A SAT-Based Analysis of a Calculus for Wireless Sensor Networks. In *Proc. 9th IEEE International Symposium on Theoretical Aspects of Software Engineering (TASE 2015)*, pages 23–30. IEEE Computer Society, 2015.

[41] X. Wu and H. Zhu. A Calculus for Wireless Sensor Networks from Quality Perspective. In *Proc. IEEE 16th International Symposium on High Assurance Systems Engineering (HASE 2015)*, pages 223–231, Daytona Beach, FL, USA, January 2015.

[42] X. Wu and H. Zhu. Probabilistic Analysis of a Calculus for Wireless Sensor Networks. In *Proc. 4th International Workshop on Formal Techniques for Safety-Critical Systems, (FTSCS 2015), Paris, France, November 6–7, 2015. Revised Selected Papers*, volume 596 of *Communications in Computer and Information Science*, pages 155–171. Springer, 2015.

[43] S. Zhao, L. Fu, X. Wang, and Q. Zhang. Fundamental Relationship between Node Density and Delay in Wireless Ad Hoc Networks with Unreliable Links. In *Proc. 17th Annual International Conference on Mobile Computing and Networking, (MOBICOM 2011), Las Vegas, Nevada, USA, September 19–23, 2011*, pages 337–348. ACM, 2011.

[44] S. Zhao and X. Wang. Node Density and Delay in Large-Scale Wireless Networks With Unreliable Links. *IEEE/ACM Trans. Netw.*, 22(4):1150–1163, 2014.