Knowledge Theoretic Design for Data Transmission Protocols of Wireless Sensor Networks

Ioanna Symeou
Real Time Systems Group
Computer Science Department
University of York
UK

February 12, 2007

Abstract

In this report, the use of knowledge-theoretic principles to analyse Wireless Sensor Networks and design protocols is examined. Nodes in the network are viewed as knowledge-agents which are able to evaluate formulas using a knowledge-model of the network, and make informed decisions during the run time of a knowledge based data transmission protocol. Such a protocol compares with standard data transmission protocols in terms of resource consumption, protocol overhead, and whether or not the protocol can be verified. Such knowledge-theoretic protocols seem to improve some aspects of network performance, but they have a potential drawback of a large protocol overhead of the data structures needed for manipulating knowledge. It is thus interesting to further investigate appropriate knowledge models and design methodologies for Wireless Sensor Network protocols that will provide a tradeoff between systematic protocol design, performance, resource consumption, and protocol overhead.

1 Introduction

Sensor networks have been used in industry for monitoring purposes for many years. Systems using CCTV cameras, smoke detectors, heat sensors and so on are part of everyday life. Lately, advances in MicroElectroMechanical Systems, wireless networking, the growth in semiconductor technology and the miniaturization of energy capacity have enabled sensors to be miniaturized and communication to become wireless, thus enabling Wireless Sensor Networks (WSNs) to emerge. A WSN is a unique distributed system, which consists of a collection of nodes which monitor the physical environment and wirelessly communicate by exchanging messages.

These networks differ from other kinds of ad hoc distributed systems since they have very limited resources, can be quite unpredictable due to their close relation with the physical environment, are more prone to failures, and use a data centric paradigm [1]. Research has focused on designing and implementing new protocols and services for WSNs that suit their special requirements. It is nevertheless equally important to provide novel design and modelling methodologies for these networks, in order to efficiently abstract all aspects of WSNs.

One such aspect is the notion of knowledge of the nodes in the network. Since a WSN is a distributed information gathering system, it seems natural to model and abstract what individual or groups of nodes in the network know about the monitored area and about each other. With each message received, or with each timeout for any expected message, nodes are able to draw conclusions about either their neighbouring nodes, or the environment in general. Those conclusions could thus enable nodes to develop *knowledge* and *beliefs* not only about events occurring in the environment, but also about the status of other nodes in their communication/sensing group. Considering the knowledge developed by nodes during the network's operation, nodes can be viewed as *knowledge agents* and the network can be modelled as a *knowledge-theoretic multi-agent system.* The contribution of the work presented in this report, is the exploration of the potential of adopting a knowledge theoretic abstraction for WSNs and their protocols, and investigation of the advantages and drawbacks of this approach.

The rest of this report is organized as follows. Section 2 provides motivation in abstracting WSNs as knowledge theoretic systems. In section 3 some basic notions of the methodology used in this work are given. Sections 4 and 5 examine a knowledge based data transmission protocol for WSN and evaluate the protocol against standard data transmission protocols. The report concludes with section 7, which includes a discussion for directions of future research and a summary of contributions made.

2 Motivation

As it was pointed out by Stankovic et al. [11], few results have been recorded for WSNs using analytical and formal models, thus providing a systematic method for abstracting a WSN and its protocols. One way to model WSNs and provide analytical results is by abstracting nodes as knowledge agents and take into account their knowledge of various facts about the network. To the best of the author's knowledge, no attempt has yet been made to explicitly examine, model, and provide appropriate data structures for representing nodes' knowledge about their neighbours, including knowledge about *their neighbours' knowledge and beliefs.*

A node's knowledge is an important resource that can effect decision making during run-time, and thus WSN protocols become more dynamic and adjust to the state of knowledge of the nodes. Moreover, certain information that needs to be communicated under specific protocols might already be known, and thus message exchange for that piece of information is not necessary, hence saving battery power. It is also possible to derive bounds for the number of messages exchanged to achieve a degree of "knowledge quality", such as a degree of confidence for detecting events in the environment, or percentage of neighbours for which certain facts are

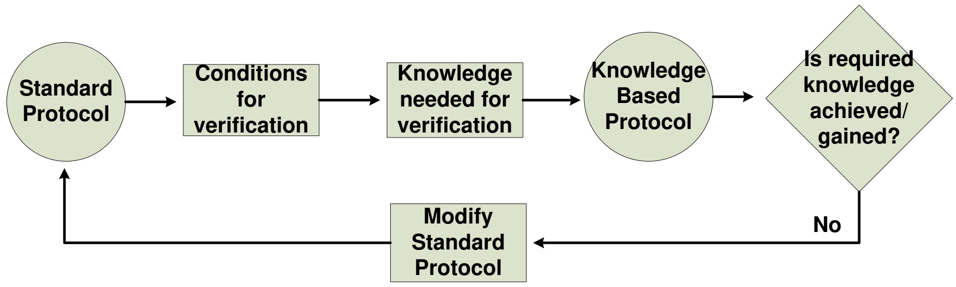

Figure 1: Analyzing SPs by creating KBPs

known. Furthermore, one could identify through off-line analysis what facts need to be known by the nodes so that a specification for a protocol is satisfied, and thus design and verify the protocol according to the knowledge of the nodes.

Having a representation of nodes' knowledge under different protocols could also help in evaluating the protocols, in terms of under which protocol nodes know more accurate facts, how many "unnecessary" messages are being exchanged for information that is already known, and so on. Protocols can also be compared by examining the knowledge of nodes and identifying inconsistencies between what nodes believe about each other and their actual state, or between the nodes' perception of their environment and/or group and the user's beliefs.

An application of a knowledge-theoretic view of a WSN which has been taken into consideration by the author in designing and examining a *Knowledge Based Protocol* (KBP). In knowledge based programmes [8] (and protocols) before an agent proceeds with an action, a knowledge test is performed. Based on this principle, Standard Protocols (SP) for WSNs can be enhanced, by adding an extra knowledge check level. Before certain actions take place, a node checks its knowledge by performing a knowledge-test about a certain fact. If the node cannot decide based on its knowledge, the standard test of the protocol takes place. One could therefore potentially systematically design and verify protocols by examining when is a SP verified, what knowledge is thus required for the nodes, and how this knowledge can be gained or transferred, thus creating a KBP. This process is outlined in Fig. 1.

In order for such a KBP to be designed, it is necessary to consider an appropriate data structure, or a *knowledge-model*, that would allow nodes to store and manipulate knowledge related facts. Such a model would have to be large enough to capture all information necessary for the nodes to evaluate their knowledge, but at the same time realistically small to be stored and processed on memory and storage constrained nodes. This is a challenging and interesting investigation, also due to the potential explosion of alternative beliefs nodes could deduce based on such a model.

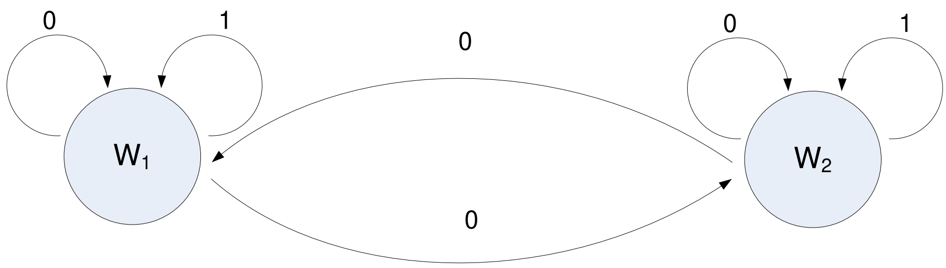

Figure 2: Kripke structure representing a system with two agents.

3 Knowledge Theoretic Basics

Modelling and formalizing knowledge in various systems generally uses *epistemic logic* principles [3] which is a logic for reasoning about what agents *know* and *believe*. Epistemic logic is used to analyse and model distributed message passing systems and prove properties such as: the infeasibility of simultaneous action under certain error conditions, the impossibility to guarantee termination of knowledge-transfer protocols, and upper message bounds for knowledge communication [3, 6]. This methodology has also been used to verify and enhance various message transmission protocols [4], including the Internet's Transmission Control Protocol [12]. Epistemic logic thus appears to be appropriate for abstracting WSN as knowledge-theoretic systems.

3.1 Possible worlds structures

One of the fundamentals of epistemic logic is the notion of a world [5] an agent considers possible in a system, according to a specific protocol. These possible worlds are usually represented by graph-based *Kripke structures* [7]. The vertices of a Kripke structure represent different possible worlds, and the edges correspond to accessibility relations showing which worlds the agents consider indistinguishable. Moreover, it is trivially true that for any agent, any world is indistinguishable to itself. Consider for example two nodes 0 and 1 which exist in an environment with lossy communication. The nodes follow a protocol such that whenever a node detects an event it must broadcast a message. Assume that node 0 detected an event, but did not receive a message from 1. There are thus two possible worlds considered by node 0: w_1 in which node 1 detected the event and its message was lost, and w_2 where 1 did not detect the event at all. The Kripke structure for this system is shown in Fig.2.

3.2 Formulas

Epistemic logic uses the same operators and propositions as propositional logic, with extra operators to express knowledge. Formulas consist of propositions $p_0, p_1, ...$, the operators $\neg$, $\wedge$,$\vee$, $\rightarrow$, and the epistemic logic operators, out of which only the knowledge operator is of relevance in this report: K_i, agent i *knows*. An epistemic logic formula is defined in a similar manner as a propositional logic formula, with the addition that if φ is a formula, so is $K_i\varphi$, meaning that agent i knows φ.

When a formula holds in a world w of a model $\mathcal{M}$ the notation $(\mathcal{M}, w) \models \varphi$ is used. Epistemic logic formulas are evaluated using the basic rules of propositional logic; also $K_i\varphi$ is evaluated as: $(\mathcal{M}, w) \models K_i\varphi$ if and only if $(\mathcal{M}, w') \models \varphi$ for all w' indistinguishable to w for agent i. Consider again the example with nodes 0 and 1, and let p be the proposition "node 1 detected the event". Since in w_2 it is assumed that 1 did not detect the event then $(\mathcal{M}, w_2) \not\models p$, but in w_1, $(\mathcal{M}, w_1) \models p$ holds. Since both w_1 and w_2 are indistinguishable for 0, then in neither worlds K_0p holds, thus formalizing that node 0 does not know if 1 detected the event. If node 0 receives 1's message, world w_2 is not possible anymore and the only world indistinguishable to w_1 from node 0's perspective is w_1 itself. Therefore $(\mathcal{M}, w_1) \models K_0\varphi$ now holds.

4 Protocol Analysis

This section presents and discusses an example of a KBP for WSNs, inspired by epistemic logic principles. The KBP implements a knowledge test on top of a simple data transmission SP for WSNs. Both protocols and the network assumptions are kept simple, since this work focuses on an experimental approach.[1] Also, the simplicity of the protocols enables the derivation of evaluation results related more to the knowledge-theoretic part of the algorithm which are less affected by the details of the underlying protocol. The goal of the design and implementation of this KBP, is to investigate: (i) if knowledge-theoretic models can be used in sensor networks at run time, so that nodes would make decisions evaluating their knowledge, (ii) if such a knowledge-theoretic structure could be used to verify WSN protocols under certain assumptions, (iii) what would the performance and overhead of a simple KBP be, and (iv) how one could design a KBP based on a simple SP for data transmission. The KBP is evaluated through experimental simulation against the SP and through analysis of the protocol overhead and resource usage.

[1] Current research is concentrating in performing knowledge-theoretic analysis and enhancement to an actual WSN protocol, with more realistic network assumptions.

4.1 Network and protocol assumptions

The following properties were assumed for the WSN: a node has sensing, communication, and processing units, and limited battery power. The network consists of groups of nodes, and each group has several subgroups of N nodes. Members of a subgroup are synchronized nodes that sleep and wake up approximately at the same time. Moreover, nodes in a subgroup are *neighbours* and are assumed to know each other's unique IDs. Neighbouring nodes have similar sensing and communication capabilities, meaning that all nodes in a subgroup can detect an event and receive a message broadcasted to the subgroup. Nodes have dual-radios [2, 9, 10] and can use a high power radio for data packets, and a low power one for small control packets. Any message can be lost with probability p_m.

Nodes periodically wake up and monitor a specific value of the environment, for instance temperature. If a node senses a change in the value over a threshold since the last time it took a measurement, it detects an event of interest. It is assumed that nodes do not falsely identify non-existing events. Moreover, an event lasts long enough for all nodes in a subgroup to detect it, but a node *could* miss an event with probability p_e. It is however assumed that any event is always detected by at least one node in the subgroup.

To provide for battery savings, the detection quality is assumed to be such that it is sufficient for one node to send a data message for an occurring event.[2] A data transmission protocol should hence satisfy the property that *for each occurring event, exactly one node in a subgroup sends a data packet to a leader node*: if no nodes transmit, the event not communicated, and if more than one node sends a packet the overall battery of the subgroup is needlessly reduced. Finally, to ensure that nodes consume energy at equal rates, nodes take turns in sending data messages.

4.2 Standard protocol

Two different versions of the SP were considered: a version where nodes consume fewer resources and transmit data messages based on a fixed order (V1), and a second version (V2) where nodes listen for data message thus adjusting the sending order. The SP does the following: if an event of interest is detected, a node performs a *standard test* to decide based on an increasing ID order, if it is its turn to transmit a data message.

It is quite clear, if an event is not detected by a node n_i then the sending order is jeopardized, since n_i has a different view than its neighbours of when it is its turn to send. In the second version of the SP therefore, nodes listen on the wireless channel for data message transmission and adjust the

[2]It is true that the message might be lost, but our main concern is to verify that a protocol satisfies a given condition, and the protocols discussed can be expanded to satisfy a condition saying that $k > 1$ messages must be sent.

sending order. If a node receives two or more data messages for an event, it adjusts its order according to the data message whose sender had the highest ID. Similarly, if a node both sent and received a data message for an event, it only adjusts its perception of the order if the message received was sent from someone with a higher ID. In some ways, this is a knowledge algorithm where the nodes do not explicitly perform knowledge tests to make their decisions.

The pseudocode for the two versions is the following:

First version (V1):

```
1 loop until battery runs out {
2   sleep for t time units
3   take measurement
4   if (value change ≥ threshold and standardTest()) send data message
5 }
```

Second version (V2):

```
1 loop until battery runs out {
2   sleep for t time units
3   receive data messages
4   adjust sending order
5   take measurement
6   if (value change ≥ threshold and standardTest()) send data message
7 }
```

When a node performs a standard test in lines 4 and 6 of V1 and V2 respectively, it checks whether or not it is its turn to send according to the ID based sending order.

4.3 Knowledge-based protocol

This sections discusses the reasoning behind the design of the KBP, and its basic operation.

4.3.1 Reasoning of protocol design

The two versions of the SP cannot guarantee to satisfy the condition of one message being sent for each event. In order for the first version of SP to be verified, it would be necessary for *all* nodes to detect *all* events. The second version would be verified if: the node whose turn it is to transmit *always* detects the event, and if a node that did not detect an event receives a data message before its turn to transmit, thus properly adjusting its sending order. Even though the verification requirements for the second version does not seem to be as strict as the requirements of the first version, the second version of the SP is quite resource consuming.

A KBP is thus considered, taking into account *what the nodes must know* so that the condition is satisfied, and trying to establish this knowledge using as few resources as possible. It was observed that to meet the protocol condition of one data message being sent for each occurred event, it is sufficient that: nodes in each subgroup which missed an event come to know, or become *aware* that the event occurred, and thus the sending order is maintained. In the KBP therefore, when nodes detect an event they broadcast a *small Low Power Message* (LPM). The small message contains only the sender ID, the message number and the number of data messages the sender transmitted so far.[3]

The LPM therefore is a mean for the nodes to communicate knowledge between them. A node is aware of an event e if it either detected e, or it *became* aware of e by receiving a LPM from some neighbour that identified the event. Since it is assumed that any event e is detected by at least one node, and if it is further presumed that nodes which did not detect e receive at least one LPM, then all nodes are aware of e. The KBP thus appears to be verified if a node which did not detect an event receives at least one LPM. This requirement does not appear to be stricter than the verification requirements of the two versions of the SP.

Since battery power is a limited resource for nodes, a good design methodology for the WSN would be that nodes use their knowledge to decide who should transmit the next data message, according to which node they believe has the most battery. A node n_i therefore decides if a neighbour would be the next sender based on what is knows about its neighbour's battery *and* if the neighbour under consideration *also knows it should sent*; there is no point in deciding a neighbour will be the sender, if the neighbour itself does not decide to transmit.

4.3.2 Protocol basics and pseudocode

In the KBP, nodes evaluate certain facts using a Kripke structure-based *knowledge model*, which reflects all the possible realities (worlds) for the subgroup up to the specific round of the KBP. Each world in the model is assigned with a probability indicating how possible it is for that world to be the reality. Nodes use the model to probabilistically decide who the next sender is. The model is updated and the node changes its view of possible realities as new information is learned through LPM reception, or when the node is uncertain about the real state of its neighbours. New worlds are created in the model when the node is uncertain if neighbours received a LPM, detected an event, or transmitted a data message due to the possibility of its decision about the next sender being wrong.

[3] The message number and the number of data messages are needed so that nodes can identify the correct state of their neighbours and change the knowledge model accordingly.

In each world considered in the knowledge-model, n_i makes a *world decision* about the next sender, based on which neighbours it believes have the most battery that could also decide to transmit. During the world decision process, the decision function is recursively called for all other nodes in the group. This way, n_i uses knowledge about *its neighbours' knowledge* to estimate which of its neighbours would decide to transmit a data message. Out of the neighbours n_i believes could decide to transmit, and itself, the node decides on the one with the most battery or the least ID if the batteries are similar[4]. Finally, n_i makes a world decision about itself if it meets the battery criteria, or if it believes that no neighbour will decide on itself while some neighbour *would* decide that n_i should be the sender.

Once the node has made its world decisions, it makes an *overall decision* based on the probabilities of the worlds it considers possible. An overall decision is made, if in all worlds with probabilities over a certain threshold the world decision is for the *same node*. If however there are at least two high probability worlds in which different nodes are decided, then the node cannot make an overall decision. In such a case, the node being unable to decide would simply fall back to the standard protocol and transmit based on the sending order.

Finally, the KBP is designed assuming the following fault model: it is not possible for any node to loose more than m LPMs transmitted for an event e, for any node not to receive more than m' consecutive LPMs from the same sender, for more than v nodes to miss an even, and for any node to miss more than v' consecutive events.

The pseudocode of the KBP a node n_i runs is:

```
LOOP UNTIL BATTERY RUNS OUT {
  SLEEP FOR t TIME UNITS
  TAKE MEASUREMENT
  IF (VALUE CHANGE ≥ THRESHOLD) {
    SEND AND RECEIVE LPMs
    ELIMINATE IMPOSSIBLE AND CREATE NEW WORLDS
    sender_node = n_i.DECISION()
    IF (sender_node == n_i) SEND DATA MESSAGE // if n_i decided itself
    IF (sender_node == NULL) {
      // if no decision could be made
      IF (STANDARDTEST()) SEND DATA MESSAGE
     }
  }
  ELSE {
    // no event was detected
```

[4] Two nodes have similar batteries if the difference between their batteries is less than a specified amount d. That amount d is such that two nodes are considered to have non-similar batteries, only if they transmitted different number of data messages.

```
14   RECEIVE LPMs
15   ELIMINATE IMPOSSIBLE AND CREATE NEW WORLDS
16   IF (EVENT MISSED) {
17     TAKE MEASUREMENT
18     sender_node = n_i.DECISION()
19     IF (sender_node == n_i) SEND DATA MESSAGE
20     IF (sender_node == NULL) {
21       IF (STANDARDTEST()) SEND DATA MESSAGE
22     }
23   }
24 }
25}
```

The function DECISION() operates as such:

```
FOR EACH WORLD:
  FOR EACH NEIGHBOUR n_j {
    sender_node = n_j.DECISION()
    IF (sender_node == n_j) ADD n_j TO LIST // get any node that decides itself
  }
  IF (LIST IS EMPTY AND THERE WAS sender_node == n_i) {
     // no neighbour decided itself, but someone decided n_i
    WORLD DECISION IS n_i
    // n_i decides itself
  }
  IF (LIST IS NOT EMPTY) {
    MAKE WORLD DECISION FROM NODES IN LIST BASED ON BATTERY CRITERIA
  }
}
  // overall decision follows
IF (THERE IS sender_node SO THAT sender_node IS WORLD DECISION IN
ALL HIGH PROB.  WORLDS) {
  OVERALL DECISION = sender_node
}
CREATE DECISION UNCERTAINTY WORLDS
RETURN OVERALL DECISION
```

Note that in line 16 of the KBP, if a node comes to learn that it missed an event e through LPM exchange, it proceeds in the same manner as if it has actually detected e. Also, there is an infinite recursion in the decision pseudocode, which is dealt with by introducing levels in the knowledge model, discussed below.

4.3.3 Knowledge model structure

This section includes more details about the knowledge-model nodes use to

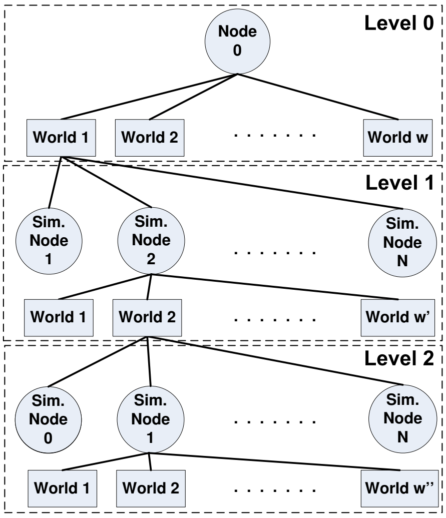

Figure 3: Knowledge-tree structure used to model knowledge

evaluate formulas. The model consists of different levels, such that each level represents a *depth of knowledge*. *Level 0* consists of level 0 worlds corresponding to the perception of node n_i which runs the KBP, about the entire subgroup. In each such world, the neighbouring nodes are represented by *simulated node* structures, which are in *level 1* of the model. These structures model n_i's view about each neighbour. To model different views about a neighbour n_j, in each simulated node there are several *level 1* worlds that could be possible realities for the specific neighbour. In each level 1 world therefore the variables of n_j, such as battery and number of messages transmitted, are different.

Since the KBP requires nodes to evaluate their neighbours' knowledge by estimating their neighbours' decisions, it is necessary to model a node's perception about its neighbour's perception. Therefore, each level 1 possible world recursively includes *level 2* simulated nodes. Each level 2 simulated node thus models what the node believes about its neighbours' perceptions of *their* neighbours. Similar to level 1 worlds, for each level 2 simulated node there are several *level 2* possible worlds. Nonetheless, since it is not necessary to further model neighbours' views about their neighbours' beliefs, there is no need to include simulated node structures in level 2 worlds. The model is thus a tree-structure restricted in three levels and it is sufficient for capturing the depth of knowledge required by the KBP. The knowledge-model is shown in Fig.3.

The terms used is this report are: *node* or *actual node* for the node that exists in the network and stores the knowledge model, and *intermediate* and *leaf node* for level 1 and level 2 simulated nodes respectively. When the term *simulated node* is used, then both level 1 and 2 nodes are referred to.

When saying that a simulated node n_j "received/transmitted" a message or "detected" an event e, it means that the actual node believes that the node corresponding to n_j received/transmitted the message or detected e. Also, when a simulated node n_j is said to make a decision, this refers to what the actual node believes n_j's decision would be based on n_j's knowledge. Since leaf nodes however do not have knowledge about other nodes, are thus assumed by default to always decide themselves as the sender. This also serves as a way to deal with the infinite recursion of the decision algorithm.

4.3.4 Knowledge-model updates: world creation and elimination

While the KBP runs and LPMs are being exchanged or lost, the knowledge-model is updated so that worlds are created reflecting uncertainty about the status of the group, and eliminated with the reception of new information. This section describes in more detail the process of world creation and elimination. A running example used in this section to demonstrate the knowledge-update procedure is the example of nodes 0 and 1, mentioned in section 3.1: nodes 0 and 1 monitor the environment for a value of interest, and transmit a message whenever they detect an event (LPM). Throughout the example it is assumed that it is the first time the nodes run, and they have equal batteries.

World creation

First and foremost, worlds where the fault model assumptions are violated, are not created. However, the assumption that nodes are aware of each event is *not* part of the fault model, and hence worlds in which a node is not aware of an event could also exist.

Each new world is a copy of a world already in the model, in which something different has happened in the group. In the original world for instance it might be assumed that a node received a message m, and that in the new world m was lost. The world probabilities, and the simulated node variables in the old and new worlds are updated accordingly, to reflect the reality each world represents. New worlds are created in the following situations.

LPM transmission/reception: When a node receives or sends a LPM, it cannot know if the rest of the nodes in the group received the message. New worlds for each simulated node n_j are created, as a copy of each world already assumed for n_j. In the existing world w, n_j's variables are updated to reflect the possibility of n_j *receiving* the LPM. In the copy w_c of w in which n_j is assumed to *lost* the LPM, n_j's variables do not change. Also, the world probabilities are accordingly updated to reflect n_j receiving or not the LPM, based on p_m: the initial world w in which the simulated node received the LPM has probability $P(w) = P(w) \times (1 - p_m)$, and the copy w_c where the node is presumed to have lost the LPM has probability $P(w_c) = P(w) \times p_m$.

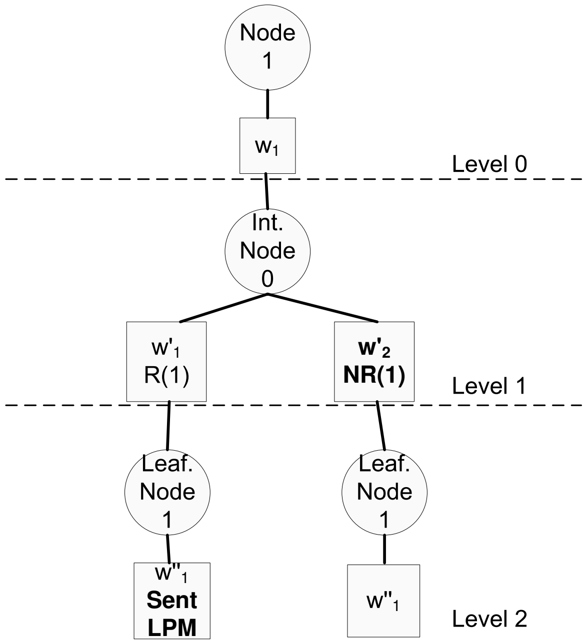

Figure 4: World creation for LPM uncertainty

Simulated nodes that are assumed to have transmitted/received a LPM recursively perform the same procedure, thus creating new level 2 worlds.

Consider the example with nodes 0 and 1, and assume that node 1 detects an event and transmits a LPM. Node 1 however cannot know if node 0 received the LPM or not. The knowledge-model of node 1 after creating worlds for LPM uncertainty is shown in Fig.4. There are two possible worlds for node 0: world w_1' where it received the LPM from 1 (denoted by $R(1)$) and the copy of w_1', world w_2' in which the LPM was not received (denoted by $NR(1)$). In world w_1', the perception of node 0 for node 1 also changes, since it knows that 1 has sent a LPM. In the level 2 world therefore representing the different realities for node 1 as seen by node 0, node 1 is assumed to have transmitted a LPM (denoted by $SentLPM$). The changes in the model are shown with bold face letters.

Event detected: When a node is aware of an event e it expects that its neighbours having detected e will transmit a LPM. If it does not receive LPMs from certain nodes after a timeout, it assumes that either the node did not detect e, or that the LPM was lost. A copy w_c of each world w is created for each LPM not received, such that in w_c it is assumed that the neighbour n_j *did not detect* e, whereas in w it is presumed that e *was detected and the LPM lost.* In each w, the simulated node corresponding to n_j is updated to reflect the detection of e and the transmission of the LPM. Again, the probabilities of w and w_c reflect the possibility of the realities

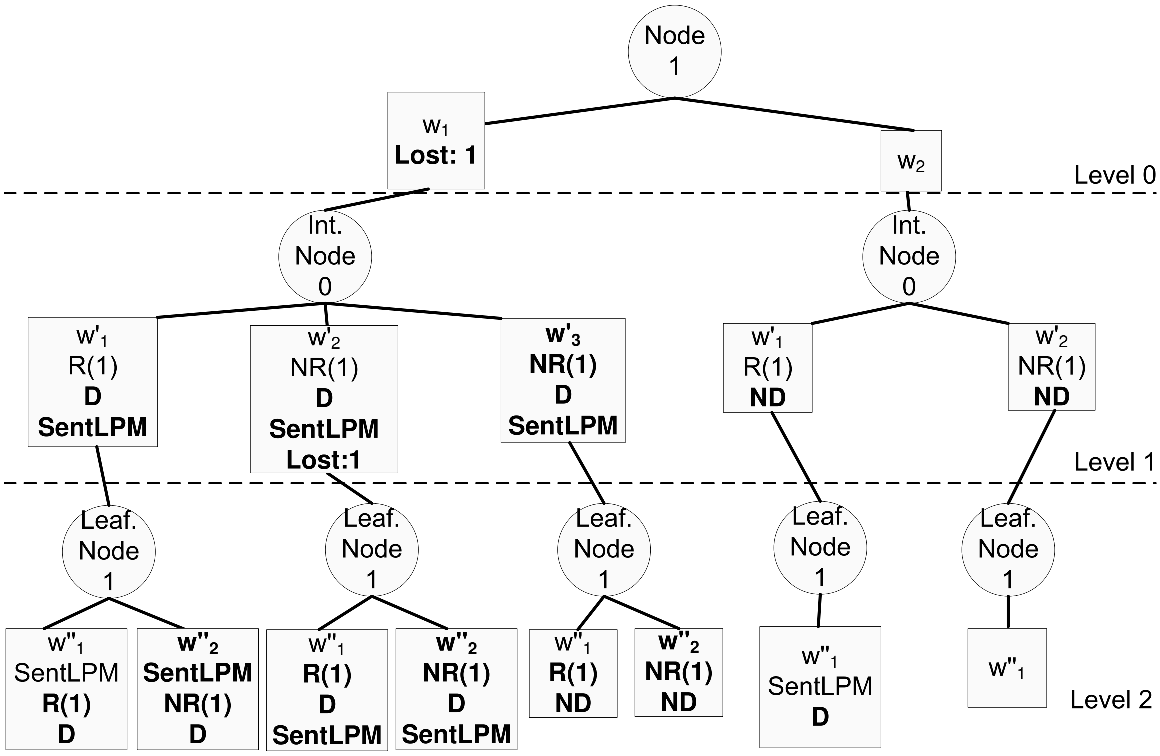

Figure 5: World creation for event detection uncertainty.

they correspond to: world w where the event was detected has probability $P(w) = P(w) \times (1 - p_e)$, and its copy w_c where it is assumed that the event was missed has probability $P(w_c) = P(w) \times p_e$. All intermediate nodes presumed to be aware of e also perform this procedure in order to recursively create new worlds. New leaf node worlds, to account for nodes detecting or missing an event are *not* created, since it is assumed that leaf nodes do not have knowledge of any other nodes in the subgroup.

Continuing the example with nodes 0 and 1, assume that node 1 did not receive a LPM from 0. Therefore, node 1 does not know if 0 detected the event and the LPM was lost, or if 0 did not detect the event at all. The new knowledge-model is shown in Fig.5. Node 1 in level 0 creates a new world w_2, in which it assumes that its neighbour node 0, did not detect the event. In the initial world w_1 therefore, it is presumed that node 0 detected the event and sent a LPM which was lost. The number of possible lost LPMs in w_1 is thus 1. Consider intermediate node 0 in level 1, when it is assumed to have detected the event (intermediate node in level 0 w_1), and when it is assumed to have missed the event (intermediate node in level 0 w_2)

- **Event detected (w_1).** To denote that node 0 detected the event, D is included in all its level 1 worlds. Since the event was detected, if node 0 it did not received the LPM transmitted from its neighbour node 1, it would not know if 1 detected the event or not. This reality corresponds to w'_2. Node 0 would thus proceed in the same manner to create event detection uncertainty worlds. A copy of w'_2 is thus created, w'_3, in which node 0 would presume that its neighbour node 1 did not detect this event. In worlds w'_1 and w'_2 it is assumed that node 1 detected the event, denoted by D in the possible level 2 worlds of leaf node 1.

Moreover, it is assumed that node 0 transmitted a LPM which was lost. It thus follows that node 0 not knowing if its neighbour node 1 received the LPM, would update its model in the manner shown in Fig.4, and update the possible worlds for node 1. There are therefore now two possible level 2 worlds for the leaf node: w''_1 where the LPM was received, and w''_2 in which it is presumed that node 1 missed the LPM.

- **Event missed (w_2).** Level 1 worlds where it is assumed that node 0 did not detect the event are denoted by ND. In world w'_1 where it is assumed that node 0 receives the LPM sent by node 1, it becomes aware that an event occurred. Node 0 would thus change its perception for its neighbour node 1, since it now knows that 1 detected an event. This is shown by including D in the level 2 worlds considered possible for node 1, by node 0. Finally, if node 0 both misses the event, and the LPM transmitted from node 1, corresponding to world w'_2, then no changes happen.

Decision uncertainty: Since a node makes probabilistic decisions about the next sender, it cannot be 100% confident that the decisions it makes are correct. Therefore, worlds must be created to reflect this uncertainty. When the actual node decides on itself, it does not create such worlds since it is certain that it will transmit a data message. For each simulated node n_j nonetheless which decided to send a data message, new worlds should be created which correspond to a reality where the decision about n_j is wrong and thus no data message is sent. In the worlds where the decision is presumed correct, n_j's variables change to reflect the fact that it transmitted a data message, and in the worlds where the decision is assumed wrong the variables remain the same. For each world w therefore where the world decision of n_j was itself, a new world w_c is created assuming the decision was *wrong* and thus the node respective to n_j does not transmit a data message. World probabilities are accordingly updated depending on the probabilities of the decisions being wrong or right, and whether the overall decision made was also for n_j. Worlds that reflect wrong decisions always have lower probabilities than those that reflect right decisions. Consider $P(D_wW)$ and $P(D_wR)$ to denote the probabilities of a world decision being

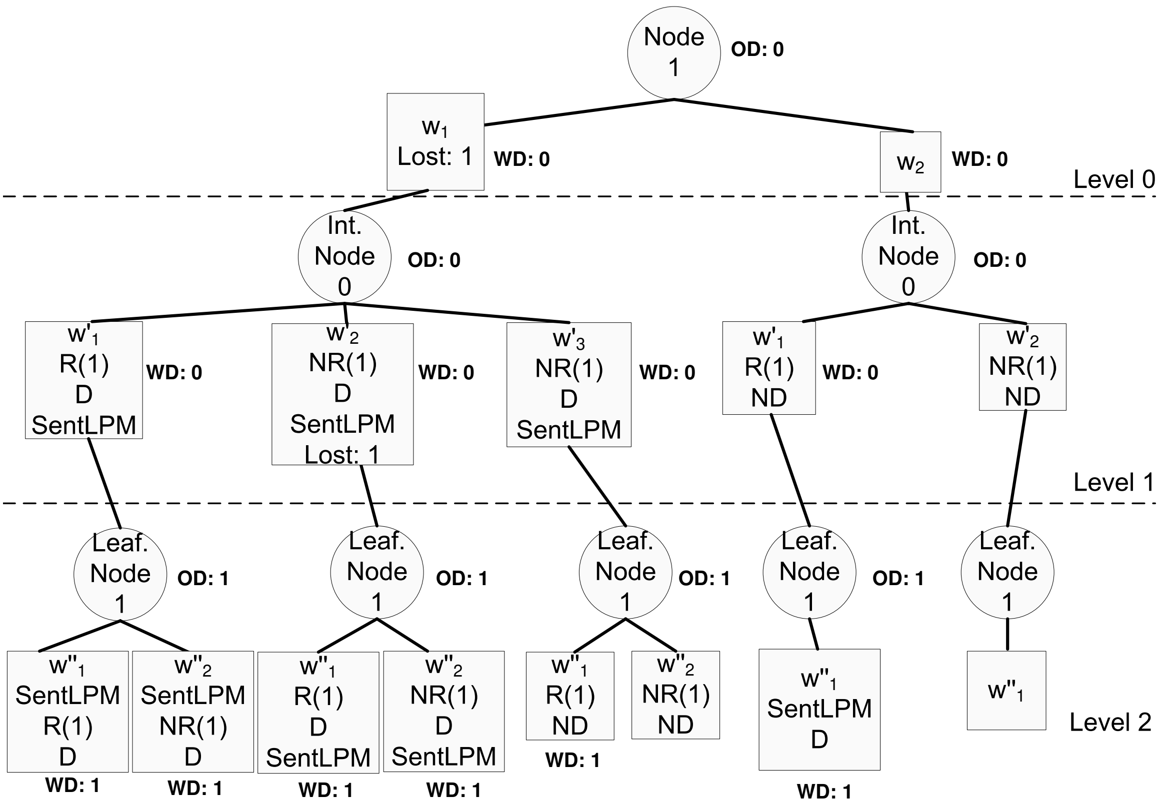

Figure 6: Decision making.

wrong and right respectively. Similarly, let $P(ODW)$ and $P(ODR)$ be the probabilities of an overall decision being wrong and right.[5] The possible worlds of an intermediate node are updated in the following manner:

- If the overall decision made by the intermediate node n_j was itself, then for each level 1 world w where the world decision was also n_j, a new world w_c is created. In w_c it is assumed that the world decision was wrong, and subsequently the overall decision must also be wrong. The probability by which the intermediate node is not the sender in w_c is $P(w_c) = P(w)P(D_wW|ODW)$. This reflects the probability of the real world being w, but the decision made in w being wrong, given that the overall decision is also wrong. The probability of the original world, in which it is assumed that the decisions are right is similarly calculated to be $P(w)P(D_wR|ODR)$.

- If the overall decision of n_j was not itself, new worlds are still created for each world w where the world decision was n_j. In the new

[5] Details about how these probabilities are omitted because they are outside of the scope of this report.

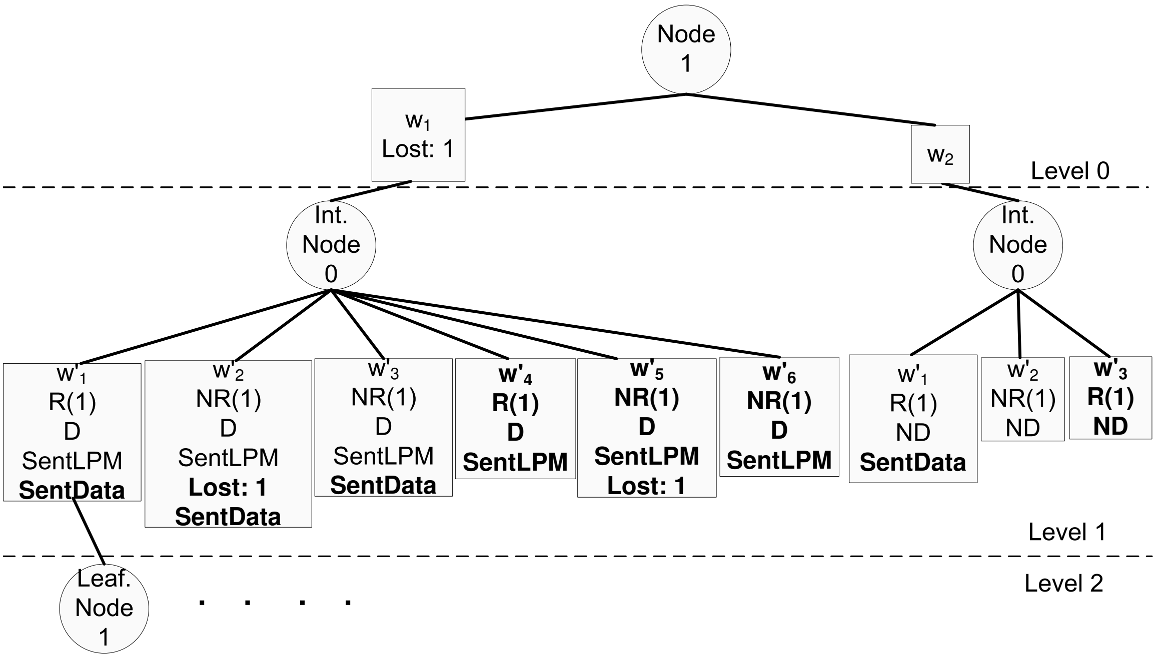

Figure 7: World creation for decision uncertainty.

worlds that are created it is again assumed that the world decision was wrong. However, this time it is implied that the overall decision must be correct and thus n_j indeed is not the sender. In the original worlds where the world decision is assumed right, it is implied that the overall decision must be wrong. The new world w_c therefore in which n_j is not the sender, has probability of $P(w_c) = P(w)P(D_wW|ODR)$. With the same reasoning, the probability of the original world w is $P(w) = P(w)P(D_wR|ODW)$.[6]

This world creation process is shown in Fig.6 and 7. Continuing the example with nodes 0 and 1, Fig.6 shows the world and overall decisions that would have been made. Leaf nodes always decide on themselves since they are assumed to have no knowledge about other nodes in the group. Also, notice that in worlds where nodes are not aware of the event, meaning they neither detected the event nor received a LPM, no decisions are made. Since it is assumed that both nodes have the same battery, the one with the least ID, node 0, is eventually chosen to be the sender. Fig. 7 shows the creation of worlds for intermediate node 0, which reflect decision uncertainty. Also, the figure does not show level 2 since there are no changes made in that level. As it is shown, a copy has been created for all level 1 worlds where the world decision was node 0. In the initial worlds it is assumed that the decision is

[6]If n_j could not make an overall decision at all, then $P(w) = P(w)P(D_wR)$ and $P(w_c) = P(w)P(D_wW)$.

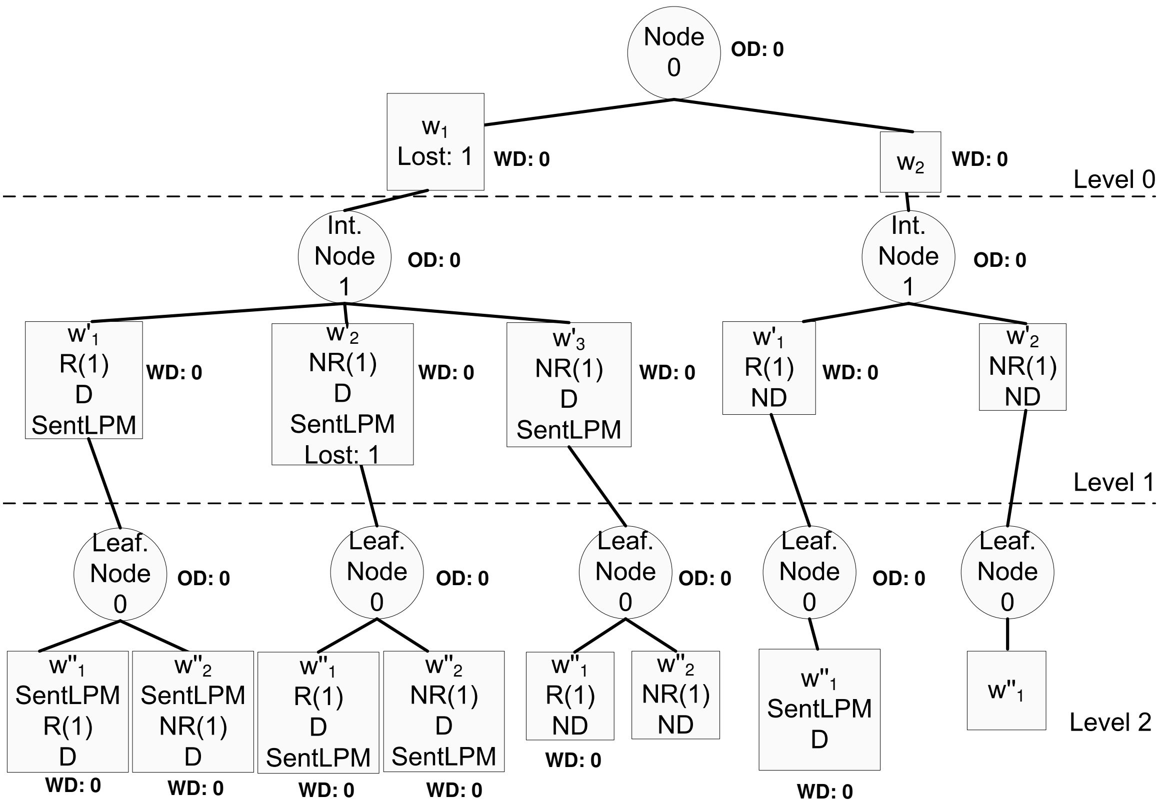

Figure 8: Decision making.

correct and node 0 sent a data message, whereas in the copies it is presumed that the decision was wrong and thus no data message was sent.

Since leaf nodes always decide themselves, if the above mentioned algorithm is used decision uncertainty leaf node worlds would be created in each round of the KBP, which would not be very efficient. Therefore, decision uncertainty leaf node worlds are created only when a leaf n_j exists in a level 1 world w, such that the world decision of its parent in w was n_j. This is illustrated in figures 8 and 9 which show the model for node 0, reversing the roles of the nodes: it is assumed that node 0 detected the event and no LPM was received from node 1.

Fig. 8 again shows the world and overall decisions. In Fig. 9 however, decision uncertainty level 2 worlds are created only where the world decision of intermediate node 1 was for node 0. Some structures are omitted from the model, but they are updated in a similar manner as it is shown in the figure.

World elimination

When a node receives information that proves certain worlds to be impos-

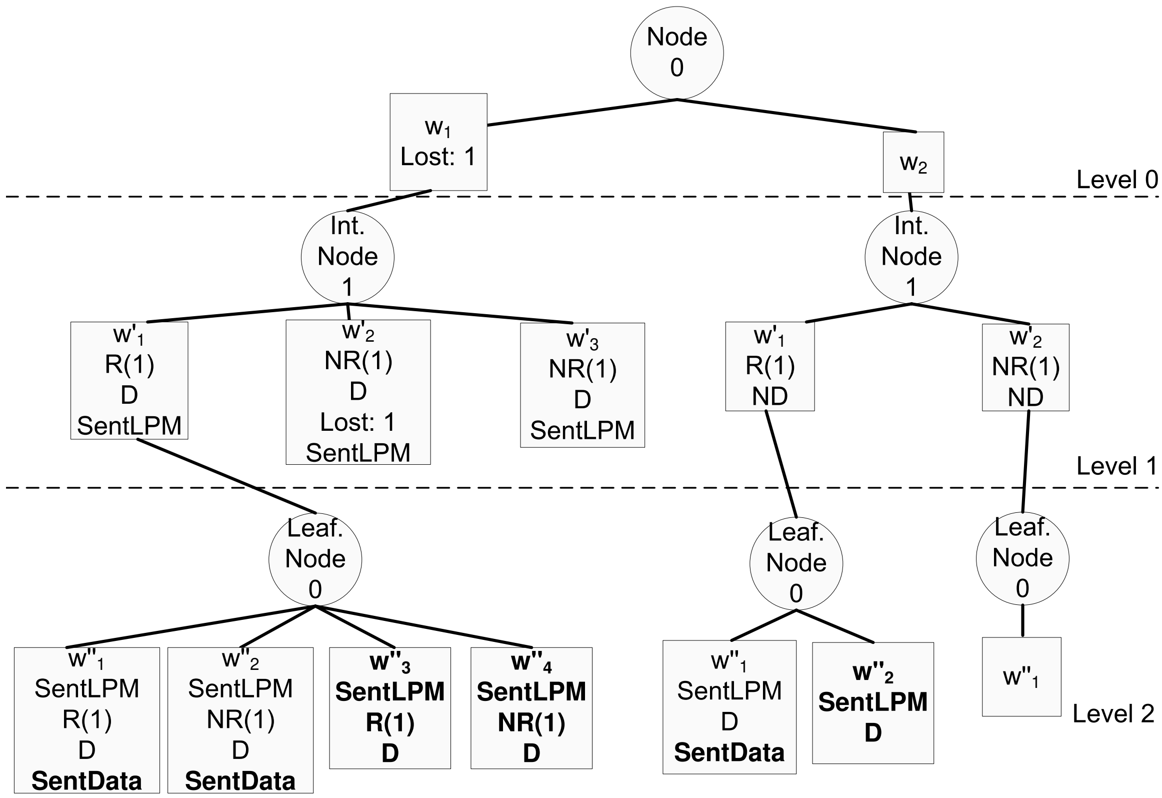

Figure 9: Level 2 world creation for decision uncertainty.

sible, those worlds are eliminated. A world is considered impossible when a LPM received indicates that the perception for the sender node in that world is wrong. In each world it is presumed that the sender transmitted a certain number of LPMs, and thus the next LPM expected to be received should have a number m. If the LPM received has a different number, worlds in which m was expected are eliminated. This nevertheless does not eliminate worlds where the LPM sender is assumed to have non-similar battery than the battery the node actually has. Since nodes are assumed to have dissimilar batteries only if they send a different number of data messages, and a LPM also includes a counter for the number of data messages transmitted by the sender of the LPM, then all worlds in which the node is presumed to have transmitted a different number of data messages messages are eliminated.

Assume for instance that in the running example with nodes 0 and 1, node 1 receives a LPM from 0 indicating that it has sent 1 data message. Moreover, let the LPM number be 2. It is thus implied that node 0 did in fact previously detect an event and sent a LPM with number 1 which was lost. It is also clear that the decision of node 0 being the sender was correct. Any worlds which are not compatible with this information are eliminated,

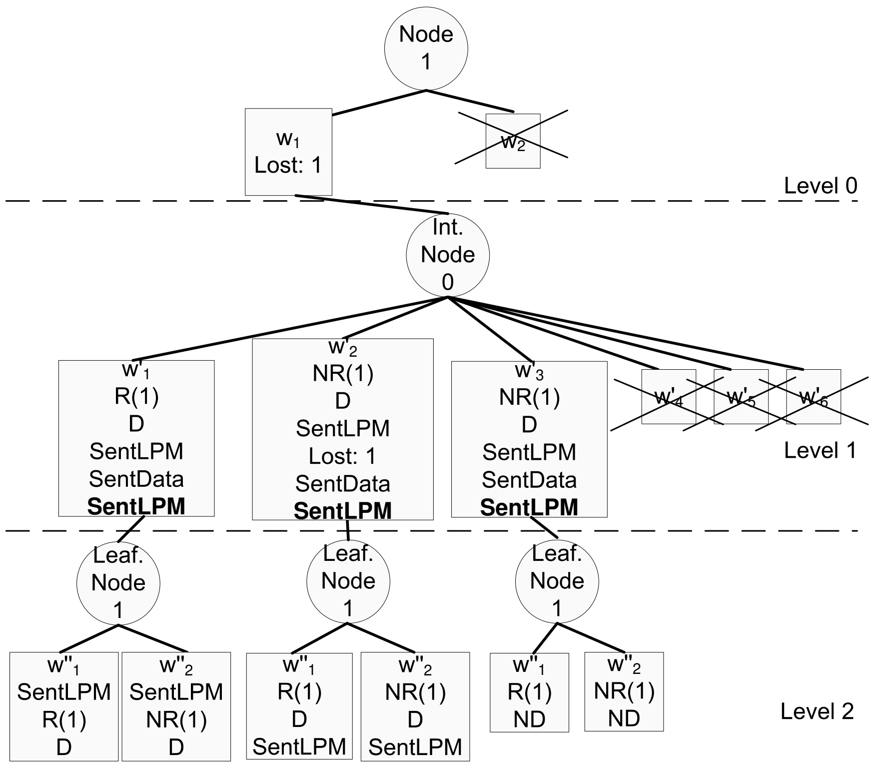

Figure 10: World elimination.

as it is shown in Fig. 10.

If however the model is to be 100% precise and have a world for every possibility for the subgroup, then the model would be very large even with world elimination, since LPMs can be lost. On the other hand, it is necessary to consider all likelihoods for the state of the subgroup to enable nodes to make informed decisions. A solution to this problem is to identify sets of *similar* worlds, and keep one world for each set. In two similar worlds, even though they represented unique realities, the decision the node makes about the next sender is the same.[7]

Two worlds w_1 and w_2 of an actual node at level 0 are considered to be similar if and only if both of the following are true:

1. Respective children simulated nodes in both worlds, have either both detected the *current event* e, or neither detected e.

2. The worlds of respective children simulated nodes are also similar: for each possible world of the simulated node n_j in w_1, there is a similar world for the respective n'_j in w_2 and vice versa.

[7]This however does not mean that if two worlds are *not* similar, then necessarily different decisions are made.

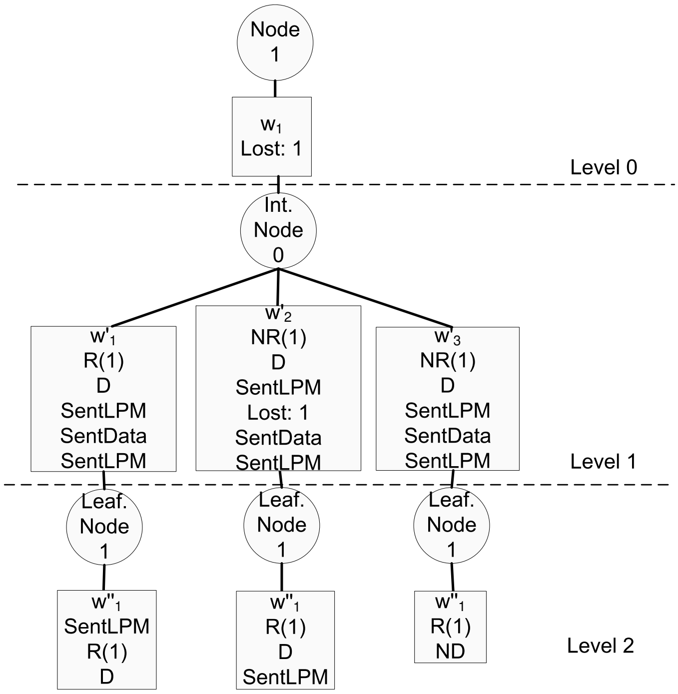

Figure 11: Level 2 similar worlds elimination.

When considering worlds of a simulated node, in order for two worlds w'_1 and w'_2 to be similar, all of the following must be true:

1. The prementioned criteria about the node's children nodes must hold.

2. In both w'_1 and w'_2 the simulated node has similar batteries.

3. In both w'_1 and w'_2 then node is either aware of the current event e, or is not aware of e in neither of the worlds.

Finally, since leaf nodes do not have children in the model, only points 2 and 3 of the conditions for simulated node worlds must hold in order for two leaf node worlds to be similar, meaning the leaf must have similar batteries and be aware of e in both worlds. Checking and eliminating similar worlds takes place before making any decisions.

Continuing the running example, assume for simplicity that the check for similar worlds happens after the elimination shown in Fig.10. Notice that the event detection uncertainty reflected in the model, is for a previous event for which a data message was sent. It thus follows that whether worlds are similar or not, is *not* affected by the event detection uncertainty shown in the model. Starting from level 2, all the worlds of any leaf node 1 are similar between them since node 1 has similar battery values in all of them. The updated model thus far is shown in Fig.11. The level 2 worlds that remain

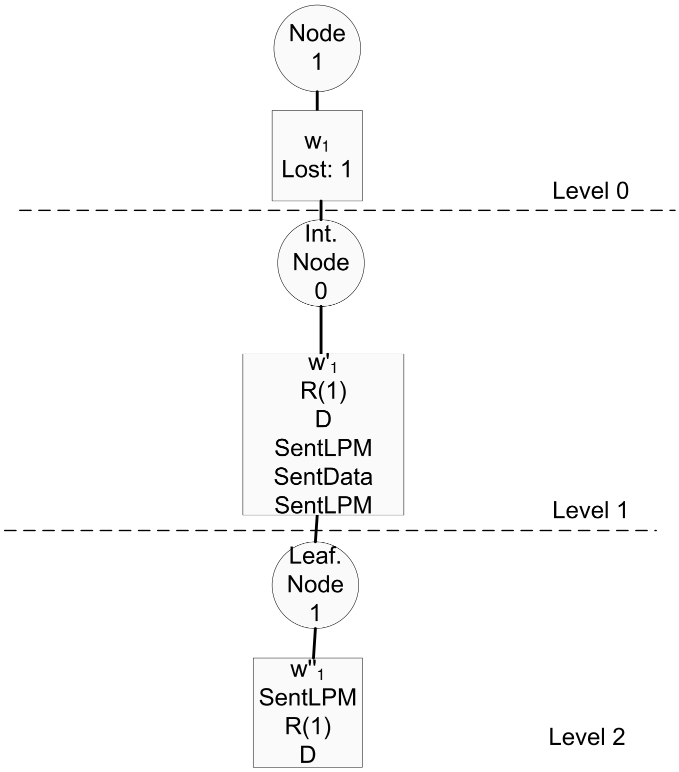

Figure 12: Level 1 similar worlds elimination.

are those with the highest probability. Notice that all level 1 worlds are also similar between them: in all worlds intermediate node 0 has similar battery values, and the respective leaf nodes also have similar worlds. Therefore, as it is shown in Fig.12 only the highest probability world w'_1 remains.

4.3.5 Decision making details

This section provides some more details about the process of decision making, and includes the knowledge-theoretic formulas that are evaluated during this process.

Assuming the formula φ_{n_j} = "node n_j will transmit a data message", a node n_i checks if $K_{n_i}\varphi_{n_j}$ (where n_i and n_j could be the same node) holds with a certain probability p in the knowledge model. This corresponds to n_i making an overall decision that node n_j should be the next sender. An overall decision about a node n_j essentially means that node n_i considers that no matter what the reality is, node n_j is probabilistically the next sender. If a node decides itself ($n_i = n_j$) it transmits a data message. If however the node is not able to decide which is the next sender, it performs the standard test and sends a data message if it is its turn to transmit according to the

ID-based order. In order for node n_i to probabilistically evaluate $K_{n_i}\varphi_{n_j}$, is checks if φ_{n_j} holds in worlds of the model whose probability is over a threshold p_w. When φ_{n_j} holds in a world, corresponds to the node making a world decision that n_j would be the next sender. This captures that node n_i considers that if the reality is the one corresponding to a certain world, then the sender would be node n_j.

In order for φ_{n_j} to hold in a world w, the perception of n_i for n_j when considering w, must be such that n_j satisfies an *eligibility criterion*, which can be anything from node n_j is the nearest to an observed target, to n_j detected an event for the longest time.In this work, the eligibility criterion is *node n_j has the most battery*, thus designing the protocol to maximize the network's useful lifetime. It is therefore necessary for formula $B_{n_j>n_l}$ = "n_j has more battery than n_l" to be evaluated in w. If there is no node with the most battery then the next node to transmit is chosen based on an *equal battery criterion.* In this design, from the nodes with similar battery values (which are more than the battery values of other neighbours) the one with the smallest ID is decided to be the next sender. Thus, evaluation of formulas $B_{n_j=n_l}$ and $ID_{n_j<n_l}$ meaning "n_j and n_l have similar batteries" and " n_j has a smaller ID than n_l" respectively, are also needed to evaluate φ_{n_j}.

Another necessary condition that needs to be satisfied so that φ_{n_j} holds in w is that node n_i can estimate that n_j itself decides with a specific probability that it is the sender. This subsequently means that $K_{n_j}\varphi_{n_j}$ must hold in w. Therefore, the formula φ_{n_j} is also probabilistically evaluated in level 1 possible worlds of intermediate node n_j. Moreover, since it is assumed that nodes can miss events, the formula will only be evaluated in worlds where n_j is presumed to be aware of the current event. Thus φ_{n_j} is evaluated only in worlds where either D_{n_je} ="n_j detected event e" or A_{n_je}="n_j *became* aware of event e" holds.

If n_i itself meets the battery criteria in a world w, then φ_{n_i} holds and the world decision is for itself. A node can also decide itself in w if it believes that no neighbour would transmit, but some other node decided that n_i should be the next sender. Using knowledge-theoretic formulas, this means that there is no n_j for which $K_{n_i}\varphi_{n_j}$ holds, but there is at least one node n_l such that $K_{n_i}K_{n_l}\varphi_{n_i}$.

Putting it all together, the knowledge model is used by a node n_i to probabilistically evaluate the formula $K_{n_i}\varphi_{n_j}$. The formula φ_{n_j} must hold in certain worlds, which requires that $K_{n_i}K_{n_j}\varphi_{n_j}$ and $K_{n_i}B_{n_j>n_l}$, or $K_{n_i}B_{n_j=n_l}$ and $K_{n_i}ID_{n_j<n_l}$, also hold. In order to decide if $K_{n_j}\varphi_{n_j}$ holds, it is necessary to only consider worlds where either $K_{n_i}D_{n_je}$ or $K_{n_i}A_{n_je}$ hold as well. Finally, φ_{n_i} evaluates to true in a world w, if n_i satisfies the sender criteria, or if no node is found such that $K_{n_i}\varphi_{n_j}$ holds, but $K_{n_i}K_{n_l}\varphi_{n_i}$ holds for some node n_l.

5 Protocol evaluation

Evaluation of the KBP was made through experimental simulation and off-line analysis of the protocol resource usage and memory/storage overhead.

5.1 Experimental simulation

The main goal of the experimental simulation is to test whether such a KBP could be implemented, so that nodes make decisions based on a knowledge-model. The implementation thus provides an additional research level and helps to validate the theory behind the KBP. Moreover, the results serve as an indication of the efficiency of the KBP as compared with the two versions of the SP, in terms of resource usage and information communication.

5.1.1 Simulation parameters

The protocols were implemented in Java, for a WSN with one group and one subgroup consisting of 3 nodes. The small number of nodes in the simulation *does not* negatively affect the evaluation: it was desired to test if each node could evaluate a knowledge model and make decisions. Under the KBP nodes behave similarly no matter what N is. This implementation corresponds to simulating a WSN of any size, with any number of groups consisting of subgroups of 3 nodes.

Certain test cases were considered, to demonstrate the KBP's function and prove the operational feasibility of such a knowledge-theoretic framework. In all cases it is assumed that an event e is detected by at least one node, and if a node does not identify e, it receives at least one LPM. The results presented here are obtained from several test cases where events and messages were randomly missed, according to the fault model.

All values of the fault model in the implementation were set to 2, besides m which was set to 1. The following costs were assumed for message communication: the cost to transmit a LPM, l_{tr}, was 1 battery unit, the cost to receive a LPM, l_{rec}, was 0.5 units, the cost to transmit a data message, d_{tr}, was 10 units, and 5 units to receive a data message, d_{rec}. Nodes were assumed to initially have 100 battery units. The simulation would run until all nodes would exhaust their battery. Battery usage was modelled such that energy is consumed only for communication, since the main difference between the protocols was the number of messages communicated. The effects of protocol overhead in terms of processing cost were not studied in the simulation. Nevertheless, some analytic results about the overhead of the KBP are given in a following section.

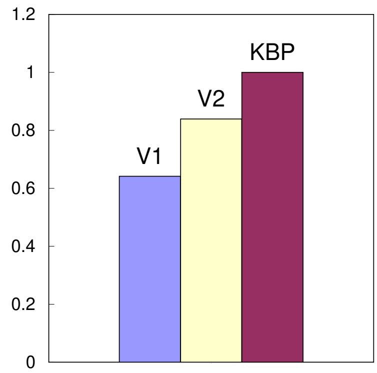

Figure 13: Distinct data messages transmitted to events occurred ratio.

5.1.2 Results

The simulation results showed that nodes successfully maintained the knowledge model and made decisions based on their knowledge. For all test cases, when a node running the KBP missed an event e, it became aware of e through LPM reception. Hence, if n_i was the sender node according to the sending order it would decide itself, and subsequently transmit a data message. Any other node was able to decide that n_i was the sender and thus just one data message was sent for each event occurred. Under the first version of the SP nodes could not identify if they, or a neighbour, missed an event and thus the sending order is jeopardized. It was noticed that event non-communication can propagate: the perception of the sending order of each node can change such that some events are never communicated. Therefore, even if nodes actually miss few events, quite many are not communicated. Moreover, even when running the second version of the protocol, several events were not communicated since the node whose turn it was to transmit could miss the event.

Fig.13 shows the average *distinct messages*[8] *transmitted to events occurred* ratio for all three protocols. This ratio measures the effectiveness of the protocol in terms of communicating information for each event that has occurred in the network. As expected, the KBP is more efficient in communicating unique pieces of information about occurring events. The first version of the SP performs the worst, since there is no communication between the nodes and the sending order cannot be re-established.

A drawback of the KBP is that nodes consume more battery than in the first version of the SP, due to LPM communication. Nonetheless, the *total* battery of all nodes in the group reduces gracefully under the KBP, whereas the reduction is unpredictable and under the SP. With the KBP the user can be sure about the relative battery units of the nodes at any point in the

[8]Data messages are distinct if just one node sent a data message. In any round where two or more messages are sent, the number of distinct data messages is one.

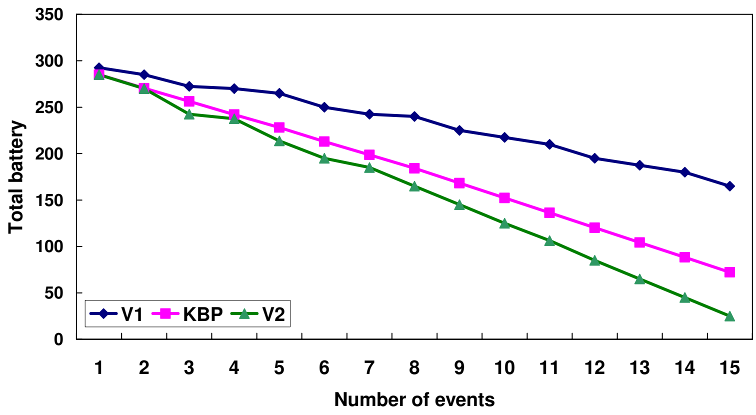

Figure 14: Total battery consumption.

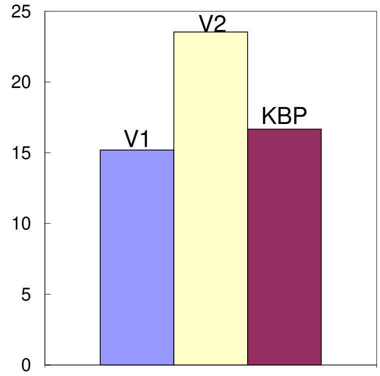

Figure 15: Energy consumed to distinct data messages transmitted ratio.

run. Fig.14 shows the average total battery under the three protocols for 15 occurred events.

Considering moreover the *total energy consumed to distinct data messages sent* ratio shown in Fig.15, it is shown that even though the KBP is more energy consuming, it is similar to the first version of the SP in terms of energy being efficient consumed for sending distinct data messages. The second version of the SP has the worst performance for this metric, since it consumes the most energy to communicate distinct data messages.

5.2 Analytic evaluation

This section includes analytic results about the performance of the KBP, obtained through off-line analysis. Through this analysis it is investigated if in fact the KBP can be verified, the effects of the LPM communication in resource consumption, and the protocol overhead due to the size of the knowledge-model.

5.2.1 Protocol verification

As it was previously mentioned, in all test cases the KBP was verified and exactly one node would sent a data message for each event that occurred. Even though this behaviour was recorded only for the test cases ran, it can in fact be proven that the KBP is verified, under the assumptions that all nodes are aware of all events.

Theorem 1 *Under the KBP, if actual nodes are aware of all events, then whenever a node/intermediate node n_i makes an overall decision that node n_j is the next sender, then according to the sending order it is indeed n_j's turn.*

The proof of Theorem 1 is in the Appendix.

Theorem 2 *Under the KBP, if actual nodes are aware of all events, then for each event only one node transmits a data message.*

Proof Assume any event e and a node n_i whose turn it is to transmit a data message according to the sending sequence. If n_i makes an overall decision, then according to Theorem 1 it will decide itself and transmit a data message for e. If n_i could not make an overall decision, it will perform the standard test. Since it is indeed its turn to send a data message, then it will do so. Consider any other node n_j, and assume that n_j makes an overall decision about event e. According to Theorem 1 ,n_j also decides n_i and thus n_j does not transmit. And finally, assume that n_j could not make an overall decision and performs a standard test. Then, it will not transmit a data message since it is not its turn to send yet. Thus, for any event e, one node transmits a data message. $\square$

The KBP protocol is thus verified only if all nodes become aware of all events. This requirement is not very strict, since it is met if a node that missed an event receives *at least* one LPM. Nevertheless, this might fail in a real WSN. It is therefore interesting and part of future work to examine the probability with which such a KBP would be verified, assuming more realistic message loss models in the network.

5.2.2 Resource usage

One of the drawbacks of the KBP compared with the first version of the SP, is that it nodes consume further energy to communicate LPMs. It is possible nevertheless to guarantee a minimum number of events that are communicated under the KBP, by either selecting an appropriate number of nodes per subgroup, or by adjusting the cost to receive and transmit LPMs.

For this analysis it is assumed that no events or LPMs are missed, since in this case the nodes running the KBP consume the most battery. For each event that has occurred in the network, nodes consume $l_{tr} + (N-1)l_{rec}$ battery units to transmit and receive LPMs. Assume that the events have unique IDs so that the first event has ID 1, and the k_{th} event has ID k. Let node n_l be the node that communicates the last event with ID L_e. Assuming initial B battery units, if the battery consumption of n_l after it communicated L_e, is less or equal to B, it is guaranteed that at least L_e events are communicated. With the KBP therefore,

$$\left\lceil \frac{L_e}{N} \right\rceil d_{tr} + L_e[l_{tr} + (N-1)l_{rec}] \leq B$$

must hold, where $\lceil \frac{L_e}{N} \rceil$ is the total number of data messages transmitted by n_l. For example, if it is required that in the KBP at least 20 events are communicated and keeping $B = 100$, $n = 3$, and $d_{tr} = 10$, then $l_{tr} + 2l_{rec} \leq 1.8$. A bound for the number of nodes per subgroup can also be calculated: if $B = 100$, $d_{tr} = 5$, $l_{tr} = 0.2$, $l_{rec} = 0.1$, and $L_e = 60$ then $5 \leq N \leq 10$. If a subgroup has more that 10 nodes, then further battery is consumed for receiving LPMs. On the other hand, if there are less than 5 nodes in the subgroup, the nodes transmit a data message more often. In both cases, the nodes' batteries will be exhausted before 60 events are communicated.

Equations relating the costs to communicate LPMs and the reception cost of data messages can also be provided, to guarantee that under the KBP nodes will not consume more battery than when running the second version of the SP. In the second version of the SP, for each event occurred in the network nodes consume at least further d_{rec} battery units for receiving the data message transmitted. Therefore, when

$$d_{rec} \geq l_{tr} + (N-1)l_{rec}$$

or when

$$N \leq \frac{d_{rec} - l_{tr} + l_{rec}}{l_{rec}}$$

it can be guaranteed that for each event detected, nodes running the KBP cannot consume more battery that when running the second version of the SP.

The choice of the subgroup size and the radio model are thus critical for the network resource usage of the KBP. An appropriate choice can significantly prolong the nodes' lifetime, and subsequently network usefulness. It is thus possible to investigate the optimal combination of number of the subgroup members and maximum LPM communication cost possible, that will result to the maximum network lifetime.

5.2.3 Protocol overhead: knowledge-model size

The size of the knowledge model is key for the implementation and analysis of the KBP. It is crucial to ensure that nodes have sufficient memory to

Table 1: Maximum world bounds when $N = 3$

Parameters			World Bounds		
			Level 0	Level 1	Level 2
$v = 1$	$m = 1$	$m' = 2$	2	12	4
		$m' = 4$	8	30	6
$v = 2$	$m = 0$	$m' = 2$	1	3	4
		$m' = 4$	1	4	6
	$m = 1$	$m' = 2$	6	16	4
		$m' = 4$	54	40	6

store and process the model used. Furthermore, by obtaining a bound on the number of worlds, different models can be compared to identify the minimum size of the model for which the nodes come to know the same facts, and can effectively make the same decisions. By eliminating impossible and similar worlds the number of worlds in the model can be bounded. It has been observed that after certain LPMs have been received, the model *stabilizes* to a specific number of worlds, meaning that all worlds considered for all nodes stabilize to a *minimum number*. When LPMs are missed the number of worlds increase, but are re-stabilized again after LPM receptions. It is nonetheless more interesting to provide the bounds for the *maximum number* of worlds in each level of the knowledge model, as calculated for various fault models. Detailed equations used to derive the minimum and maximum world bounds are not be provided in this report, but are available.

Worlds are maximized when nodes have the most uncertainties for both event detection and for nodes' batteries, subsequently maximizing the number of non-similar worlds that cannot be eliminated. The maximum worlds at level 0 are created when the node continuously detects events, but does not receive any LPMs. The worlds keep increasing until the round where m' LPMs are not received, since according to the fault model no more than m' consecutive LPMs may not be received by a node, from any neighbour.

The following tables summarize the maximum bounds for worlds considered for one node, in the three levels of the knowledge model for various parameters of the fault model. The parameters were chosen to reflect combinations of the following: *at least one node* must detect an event ($v = N - 1$), *at least half the nodes* detect an event ($v = \lceil \frac{N}{2} \rceil$), *half of the maximum number of LPMs* that can be transmitted for an event must be received ($m = N - 1 - \lceil \frac{N-1}{2} \rceil$), and *the minimum number of LPMs* that can be sent must be received ($m = N - 1 - v$). The values for m' were chosen to correspond to networks with low and high error rates. The calculations were made for a subgroup of 3 nodes corresponding to the experimental simulation, and a subgroup of 10 nodes which reflects a more realistic number of nodes.

Table 2: Maximum world bounds when $N = 10$

Parameters			**World Bounds**		
			Level 0	Level 1	Level 2
$v = 5$	$m = 4$	$m' = 2$	126	1464	4
		$m' = 4$	126^3	5856	4
$v = 9$	$m = 0$	$m' = 2$	1	3	4
		$m' = 4$	1	4	4
	$m = 4$	$m' = 2$	131×256	1984	4
		$m' = 4$	131×256^3	7936	4

It is generally observed that the number of worlds in each level either increases with the number of nodes in each subgroup, or remains the same according to the rest of the fault model parameters. The only time where the number of worlds decreases with a higher N is when calculating level 2 worlds for $m' = 4$. This is justified considering that leaf node worlds increase when the intermediate parent nodes make world decisions about the leafs. For $N = 10$, at least 10 rounds must pass for the leaf node worlds to further increase and this is not the case when $m' = 4$. As it is moreover expected, the stricter the fault model, meaning as v and m decrease, worlds are reduced. However, different combinations of v and m provide different results. For example, when $N = 3$, $v = 2$ and $m = 0$, there are fewer level 0 worlds rather than when $v = 1$ and $m = 1$. This is explained considering that even if fewer worlds exist due to event detection uncertainty, when $m = 0$ only the world where no messages were lost remains.

It is crucial to note that the calculation of the world bounds is quite pessimistic which is one reason why in some cases the bound is quite a large number. Future work can concentrate on deriving and proving tighter bounds for the size of the model. In this work for instance, the effect of not allowing more than v' consecutive events not to be detected by any node, has not been fully investigated in terms of how the model size is affected. It can also be investigated whether the knowledge model, or the conditions under which worlds are eliminated ,can change so that the model's size can be further reduced. For instance, it could be sufficient to use a smaller model that will enable nodes to decide that someone else will send a data message, and not necessarily who will send. Also, latest research suggests that if the model is changed so that decision uncertainty worlds are not created, and the fault model incorporates that all nodes become aware of all events then it is possible to have a constant number of maximum worlds for all rounds.

Nevertheless, it appears that it might not be such a good design policy for the nodes to maintain and evaluate a full-scale knowledge model at run time. An attractive alternative would be to perform protocol analysis off-line using a knowledge-theoretic model, and use the results to enhance the protocol. The nodes would not need to use a knowledge model at run time and thus save precious space and memory. Another solution would be to define light-weight knowledge-models that would store less information but be efficient enough for nodes to make knowledge-based decisions at run time.

5.2.4 Evaluation discussion

Comparing the results of the evaluation with the goals set in section 4, the following are observed. Through the use of knowledge theoretic methodologies, it is possible to develop a KBP based on a SP, where nodes evaluate a knowledge model at run time. The KBP can improve the performance of the network in terms of information transfer but with increased energy consumption. For all test cases the KBP was verified and the condition met, under the considered assumptions of event detection. The assumptions though required for protocol verification might not hold in a real WSN, and it would be interesting to consider probabilistic verification. Moreover, quite a large overhead is associated with the model used in the KBP. More efficient models for knowledge manipulation would thus be a possible area of future work.

6 Contributions and future work

This report discusses a possible new knowledge-theoretic design methodology for WSN protocols. Inspired by principles of epistemic logic an example KBP for WSNs was developed, as a means to explore and investigate the advantages and potential drawbacks of this approach. It is shown that even though the KBP can improve the performance of the network, a serious criticism of the protocol is the potential large size of the knowledge-model used, since WSN nodes are severely memory and storage constrained. This demonstrates a possible disadvantage of KBPs for WSNs.

Ongoing research focuses on further examination on the size of the knowledge model and the number of worlds. Tighter world bounds are currently being investigated, as well as the physical cost to store one world structure. Alternative light-weight models for representing knowledge are also being considered, as well as means to further decrease the size of the knowledge model but still enable nodes to correctly evaluate formulas and gain knowledge of certain facts.

Another potential alternative would be to perform off-line knowledge-theoretic analysis of a protocol and use the results to enhance protocol performance. This eliminates the need to maintain a possibly large knowledge-model during run time. Future work thus concentrates on performing such off-line analysis for existing and more complex WSN protocols. Part of that research will also examine whether knowledge required for the protocol to be verified can be achieved. Eventually, a general knowledge-theoretic framework is envisioned that will serve as an alternative, or a complement, to other means of systematic WSN protocol design. Such a framework will enable designers to methodically develop WSN protocols, and explore tradeoffs between protocol performance, resource consumption, and knowledge related overhead.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.

[2] C. C. Enz, A. El-Hoiydi, J. Decotignie, and V. Peiris. WiseNET: An ultralow-power wireless sensor network solution. *Computer*, 37(8):62–70.

[3] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. The MIT Press, 1995.

[4] J. Y. Halpern and L. D. Zuck. A Little Knowledge Goes a Long Way: Simple knowledge-based derivations and correctness proofs for a family of protocols. *Proceedings of the* 6^{th} *annual ACM Symposium on Principles of distributed computing (PODC)*, 1987.

[5] J. Hintikka. *Knowledge and Belief*. Cornell University Press: Ithaca, NY, 1962.

[6] R. Koo and S. Toueg. Effects of Message Loss on the Termination of Distributed Protocols. *Information Processing Letters*, 27(4):181–188, 1988.

[7] S. Kripke. Semantical analysis of modal logic. *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.

[8] R. Kurki-Suonio. Towards programming with knowledge expressions. *Proceedings of the ACM Symposium on Principles of Programming Languages*, 1986.

[9] C. Schurgens, V. Tsiatsis, S. Ganeriwal, and M.B. Srivastava. Topology management for sensor networks: Exploiting latency and density. *Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2002.

[10] E. Shih, P. Bahl, and M. J. Sinclair. Wake on wireless: an event driven energy saving strategy for battery operated devices. *Proceedings of the*

8^{th} *annual international conference on Mobile computing and networking (MobiCom)*, 2002.

[11] J. A. Stankovic, I. Lee, A. Monk, and R. Rajkumar. Opportunities and obligations for physical computing systems. *IEEE Computer Journal*, 38(11):23–31, 2005.

[12] F. Stulp and R. Verbrugge. A Knowledge-Based Algorithm for the Internet Transmission Control Protocol (TCP). *Bulletin of Economic Research*, 54(1):69–94, 2002.

A Proof of Theorem 1

Theorem 1 *Under the KBP, if actual nodes are aware of all events, then whenever a node/intermediate node n_i makes an overall decision that node n_j is the next sender, then according to the sending order it is indeed n_j's turn.*

Proof Theorem 1 is proven with induction on the number d of overall decisions n_i makes.

Base case: $d = 1$. Since n_i made a decision, the world decisions in all high probability worlds (HPWs) were for the same node. Because $d = 1$, no node is assumed to have sent a data message, and all nodes in the KT have similar batteries. Thus, all world and overall decisions are for node n_j with the least ID. Since the actual node is aware of all events in the network, this decision process is initiated for the *first* event that occurred and hence according to the sending order it is indeed n_j's turn to transmit.

Inductive Step: assuming the claim holds for the d_{th} decision, prove the claim for the $(d+1)_{th}$ decision. At the d_{th} overall decision about node $n_{j'}$ for event e', world decisions in all HPWs were for $n_{j'}$. Assume those worlds are in a set S. (i) Let the $d+1$ decision be for the event e occurring in the network after e'. In all current HPWs including those in S, the same world decision is made. It is thus trivially true that the next decision will be for the node n_j, next in the sending order after $n_{j'}$. According to the Induction Hypothesis, $n_{j'}$ which was chosen during the d_{th} decision was indeed the node that sent a data message for event e', and the node that should transmit for e is the next in the sending order, which is n_j. (ii) If between the d_{th} and the $(d+1)_{th}$ decision there are events for which n_i could not make an overall decision, in all worlds in S the world decisions for each event after e' would still be for the nodes following the sending order after $n_{j'}$. According to the Induction Hypothesis, $n_{j'}$ was indeed the sender node for e', and since the decision process is initiated for all events in the network, in the worlds in S the correct world decision is made. Since an overall decision was made for e, then either HPWs with inconsistent world decisions were eliminated, or their world decisions changed. Hence, in all HPWs that remain the world decisions are for the same node. Since it is certain that the worlds in S which reflect the correct battery status of the group remain, and that they have high probabilities, all world decisions of the HPWs are the same as the world decisions made in worlds in S, which will be for node n_j next in the sending order. Subsequently, the overall decision will be for n_j.