TARP: A Trust-Aware Routing Protocol for Sensor-Actuator Networks

Abdelmounaam Rezgui
Dept. of Computer Science
Virginia Tech
Blacksburg, VA 24061, USA
rezgui@vt.edu

Mohamed Eltoweissy
The Bradley Dept. of Electrical and Computer Engineering
Virginia Tech
Arlington, VA 22203, USA
eltoweissy@vt.edu

Abstract

Most routing protocols for sensor-actuator networks (SANETs) are built under the assumption that nodes normally cooperate in forwarding each other's messages. In practice, this assumption is not realistic; SANETs are environments where nodes may or may not cooperate. For several reasons, a node may fail to operate as planned at deployment time. As a result, when actually deployed, protocols and applications may not be as efficient as expected. In this paper, we present TARP (Trust-Aware Routing Protocol), a routing protocol for sensor-actuator networks that exploits past nodes' routing behavior and links' quality to determine efficient paths. We implemented TARP in a TinyOS-based SANET and conducted several experiments to evaluate its performance. The obtained results confirmed that TARP achieves substantial improvements in terms of energy consumption and scalability.

1 Introduction

Sensor-actuator networks (SANETs) are inherently collaborative environments. While single nodes may be able to individually carry out some simple tasks such as sampling an attribute or broadcasting a message, the overall purpose of the network may not be attained without the collaboration of several or all of the nodes of the network. Tasks that require the collaboration of nodes include application tasks (e.g. calculating the average temperature in a given area), system tasks (e.g. synchronization), and networking tasks (e.g routing). SANET protocols and applications often implement these tasks with the assumption of a cooperative environment where nodes interact with each other with no malicious behavior. This, however, is not a valid assumption in practice. For several reasons (e.g. tampering by adversaries, malfunction, selfish behavior), individual nodes and entire networks may cease to behave as expected by their providers and users. This highlights the need to enable nodes to establish trust levels for other nodes prior to interacting with those nodes.

Interacting with untrustworthy nodes may result in several forms of resource waste, e.g. time, energy, bandwidth, etc. It is therefore important to be able to properly determine node trustworthiness, i.e. the likelihood that the node will behave as expected. The need for trust mechanisms in sensor-actuator networks is the result of four factors: (i) application requirements, (ii) system requirements, and (iii) the limitation of traditional security mechanisms, and (iv) across-network interoperability:

- **Application Requirements**

  Many sensor applications have requirements that may not be satisfied if nodes are not able to determine, a priori, to which extent they may entrust other nodes with a given task. Trust is necessary for both dependability requirements and quality of service requirements:

  - **Dependability Requirements:** Consider a SANET that has nodes with temperature sensors and a cooling mechanism controlled by a number of actuators. In an application running on this SANET, sensor nodes periodically send temperatures to the actuators. The actuators trigger the cooling mechanism when any nearby sensor detects a temperature reading of more than a given threshold. Since, sensor nodes may or may not be able to communicate directly with actuators, an end-to-end "trust chain" must exist between each sensor and all the actuators in its neighborhood for this application to be available and reliable. Consider a node n_i that detects a temperature above the specified threshold. n_i must report the temperature value to one or more actuator. Assume that a node n_j routes messages between n_i and one of those actuators. If n_j fails to collaborate, the actuator may never determine that it must activate the cooling mechanism. In

1-4244-1455-5/07/$25.00

this example, there are several failure forms for node n_j. For example, node n_j may not forward n_i's message to the actuator or forward a modified version of n_i's message, e.g. indicating a different temperature value or a different location for the sampled temperature.

Trust is necessary not only to achieve the two dependability properties of *availability* and *reliability*. It may also be necessary to achieve the *safety* in some SANET applications. Consider the previous example. The safety of the temperature monitoring application may translate into strict real-time requirements, e.g. the cooling mechanism must be triggered at most 60 seconds after a sensor node detects a temperature that exceeds the threshold. If a node n_j routing temperature values from a sensor node n_i to an actuator excessively delays n_i's messages, the system may fail to support the application's real-time constraint.

– **Quality of Service Requirements:** An application may have QoS requirements that may not be met if nodes fail to operate as expected and if their failure is not detected by their neighbors. Consider an application that has the following accuracy requirement: "the average temperature reported by a group of sensors deployed in a given area A must not differ from the actual temperature by more than 2°C". Consider a subset of nodes in the region A that report bogus temperature values. Obviously, if it is not possible to isolate these nodes, it may not be possible to satisfy the application's accuracy requirement.

- **System Requirements**

  Trust is also crucial to meet system requirements. The efficiency of many system functionalities on a node depend directly on the behavior of its neighbors and other nodes in the network. The way a node carries out certain tasks may significantly impact other nodes. For example, if a node fails to forward packets (to preserve its energy), its neighbors may have to spend some of their energy in finding other routes and resending undelivered packets. If a node can determine the likelihood that a given neighbor would cooperate in routing its packets, it may make better decisions when establishing routing paths for its packets. Also, many SANET protocols assume that nodes accurately report their energy level. If nodes do not report their energy level accurately, such protocols would obviously not be feasible. Similarly, consider several nodes that transmit according to a given TDMA schedule. If one or more nodes transmitted outside their assigned time slots, collisions would increase reducing the very advantage of TDMA over other MAC medium sharing schemes.

- **Limitation of Traditional Security Mechanisms**

  Thus far, most of the research has formulated the problem of trustworthy sensor systems as a security problem. The focus has been largely on securing the wireless communication between sensor nodes. More specifically, the objective has been to satisfy security requirements through energy-aware, lightweight security mechanisms that provide different types of authentication and confidentiality guarantees. Security mechanisms alone, however, do not guarantee application and system trust requirements. They do not provide pre-interaction reliable predictions regarding the *future* behavior of nodes. Their prime purpose is to secure the interaction between two parties. They do not provide any guarantees about the potential consequences of interacting with a given party. For example, consider an in-network aggregation scenario where a node n_i receives a message m from a node n_j carrying the value v for an attribute t. An authentication mechanism may assure n_i that the actual sender of m is n_j. Also, the two nodes may some form of encryption to ensure confidentiality. This, however, does not guarantee n_i that v is an accurate value for t. Security mechanisms are therefore not sufficient to establish trust.

- **Across-Network Interoperability**

  The need for trust-based interaction will become even greater as SANETs proliferate and become interoperable. For example, a potential future scenario would be of an application that requires the cooperation of nodes from different SANETs deployed by different providers. Nodes, in this context, would be autonomous and driven, primarily, by their self-interest, e.g. minimizing their energy consumption. Without trust-based interaction between nodes from different networks, interoperability between sensor networks would be virtually impossible. In fact, interoperability would make obsolete most of current design approaches for sensor networks. Consider, for example, key management schemes. Current approaches may be able to secure the communication between nodes within a single network. By design, these schemes would not be able to provide trust-based interaction between nodes of different networks. Mechanisms for establishing trust are therefore a pre-requisite to enable tomorrow's interoperable sensor-actuator networks.

Protocols and applications built under the assumption of cooperative SANETs are viable only when this assumption holds. In practice, this assumption is not realistic.

SANETs are environments where a node may not determine to which extent other nodes may be entrusted in carrying out a given task. The challenge is to make SANETs environments where nodes become able to accurately predict, at least with high probability, the actions of other nodes. This, in turn, would make it possible for a node to determine the consequences of interacting with other nodes.

In this paper, we propose TARP (Trust-Aware Routing Protocol), a routing protocol for sensor-actuator networks. Trust-based routing keeps track of nodes' routing behavior and links' quality to determine efficient paths from a SANET's nodes to its base station. In the next section, we describe the theory behind TARP and how TARP operates. We describe our implementation in Section 3. In Section 4, we discuss the results of our experimental evaluation. Section 6 concludes the paper.

2 Trust-Aware Routing Protocol

TARP is responsible for routing messages from the different nodes to the base station. TARP is a trust-based routing scheme. Trust refers to the confidence that a node has in a neighbor's cooperation. A node's cooperation, in this context, is the likelihood that it forwards its neighbors' messages. TARP is based on the basic idea of avoiding to route through non cooperative nodes. The intuition is that sending packets to nodes that are not likely to cooperate in routing messages to their neighbors would probably waste energy with no payoff. TARP captures the concept of cooperation in terms of "routing reputation" or, simply, *reputation*. Informally, reputation is a perception that a node has regarding another node's cooperation. TARP consists of two concurrent phases: (i) *reputation assessment* and (ii) *path reliability evaluation*:

2.1 Reputation Model and Assessment

The reputation value that a node n_i assigns to a neighbor n_j is derived from two types of reputation: direct reputation and indirect reputation:

2.1.1 Direct Reputation

The *direct* reputation of a node n_j as perceived by a node n_i is the reputation that n_i associates with n_j as a result of its first-hand, i.e. direct, interaction with it. We note $Dr_i(n_j)$ node n_j's reputation as perceived by node n_i. We define $Dr_i(n_j)$ as the probability, as evaluated by n_i, that n_j would participate in routing n_i's messages further in the network.

Consider a message m that n_i transmits to n_j and expects n_j to forward it to other nodes in its neighborhood. For n_i to learn that n_j has forwarded its message, three conditions must be satisfied: First, m must reach n_j. Second, n_j must act on m as expected by n_i, i.e. n_j must forward m to its neighbors. Third, n_j must hear its message forwarded by n_j. The first and third conditions clearly depend on the quality of the links (n_i,n_j) and (n_i,n_j) respectively. The second condition depends on node n_j's routing reputation. To derive an expression of routing reputation, we first define the concepts of echo ratio and link quality:

- **Echo Ratio:** We define the *echo ratio* between a node n_i and a node n_j (according to n_i) as the ratio of n_i's messages that n_i overhears forwarded by n_j to the total number of messages that n_i broadcasts. Let ech_{ij} be the value of the echo ratio between n_i and n_j.
- **Link Quality:** Let n_i and n_j be two nodes. We define the link quality between n_i and n_j as the probability that packets sent by n_i are correctly received by n_j. We note the link quality between node n_i and n_j: lq_{ij}.

We now derive the expression of node reputation in terms of link quality and echo ratio. Consider a node n_i that broadcasts a message m to its neighbors and consider the following events:

- B: the link (n_i,n_j) is good enough to let n_j receive m as it is sent by n_i,
- C: n_j forwards m after it receives it from n_i,
- D: the link (n_j,n_i) is good enough to let n_i receive its own message m as it is forwarded by n_j,
- A: n_i overhears m when it is forwarded by n_j

It is easy to see that:

$$A = B \cap C \cap D$$

The events B, C, and D are obviously independent. According to the theorem giving the joint probability of independent events:

$$P(A) = P(B).P(C).P(D) \quad (1)$$

By definition of direct reputation, echo ratio and link quality, the following equalities hold:

$$P(A) = ech_{ij} \quad (2)$$
$$P(B) = lq_{ij} \quad (3)$$
$$P(C) = Dr_i(n_j) \quad (4)$$
$$P(D) = lq_{ji} \quad (5)$$

From equations 1, 2, 3, 4, and 5, we can derive node n_j's routing reputation as perceived by node n_i as:

$$Dr_i(n_j) = \frac{ech_{ij}}{lq_{ij}.lq_{ji}} \quad (6)$$

A node n_i is therefore able to evaluate a neighbor n_j's direct reputation if it can evaluate ech_{ij}, lq_{ij}, and lq_{ji}. For this, each node n_i maintains a *communication state*, noted CS_i that consists of the following fields:

- nbc: the number of messages that n_i has broadcast. nbc is initialized to 0 and incremented before n_i broadcasts a message
- for each neighbor n_j, an entry in CS_i, noted $CS_i(n_j)$, that consists of:
  - ns: the number of unicast messages that n_i has sent to n_j. It is initialized to 0 and incremented before n_i sends a message to n_j
  - nr: the number of unicast messages that n_i has received from n_j. It is initialized to 0 and incremented each time n_i receives a unicast message from n_j
  - nbc: the number of broadcast messages that n_i has received from n_j. It is initialized to 0 and incremented each time n_i receives a broadcast message from n_j
  - rns: the number of unicast messages that n_j (has reported it) has sent to n_i
  - rnr: the number of unicast messages that n_j (has reported it) has received from n_i
  - $rnbc$: the number of broadcast messages that n_j (has reported it) has sent
  - $rrnbc$: the number of n_i's broadcast messages that n_j (has reported it) has received
  - fw: the number of n_i's own messages that it has overheard forwarded from n_j.

We will use the notations $CS_i.nbc$ for the field nbc of CS_i and $CS_i(n_j).f$ for field f of $CS_i(n_j)$.

A message that a node n_j sends may be of one of three types: (i) a unicast message sent to a specific other node n_i, (ii) a broadcast message that n_j sends to all of its neighbors, or (iii) a broadcast message that n_j receives from another node n_i and forwards further in the network. We therefore distinguish three types of communication actions that a node may take: sending a unicast messages, broadcasting a message, and forwarding a message:

- *Unicast Messages:* to send a unicast message to node n_i, n_j increments $CS_j(n_i).ns$ and appends $CS_j(n_i).ns$, $CS_j(n_i).nr$ and $CS_j(n_i).nbc$ to m. It then sends m to n_i.
- *Broadcast Messages:* to broadcast a message m, node n_j increments its broadcasting counter, i.e. $CS_j.nbc$, appends it to m, and then broadcasts m. We will note $m.nbc$ the values of $CS_j.nbc$ carried by m.
- *Forwarded Messages:* When n_j receives a broadcast message m from a neighbor n_i, and if it is cooperative, it increments its broadcasting counter $CS_j.nbc$, sets m's counter to $CS_j.nbc$ and then broadcasts m. The identity of the original sender of m, i.e. n_i, is *kept* in the message m. We will note $m.orig$ the original sender of message m.

We now describe the updates that a node makes on its communication state when it receives a message m. We will use the notation $m.f$ for a field f carried in m. Consider a node n_i that receives a message m from n_j. m may be a unicast or a broadcast message:

- *Unicast Messages:* When n_i receives a unicast message $m(ns, nr, ..)$ from n_j, it updates its communication state as follows:

$$CS_i(n_j).nr = CS_i(n_j).nr + 1$$
$$CS_i(n_j).rns := m.ns$$
$$CS_i(n_j).rnr := m.nr$$
$$CS_i(n_j).rrnbc : m.nbc$$

- *Broadcast Messages:* When n_i receives a broadcast message $m(nbc, ..)$ from n_j, it updates its communication state as follows:

$$CS_i(n_j).nbc = CS_i(n_j).nbc + 1$$
$$CS_i(n_j).rnbc = m.nbc$$

If, in addition, $i = m.orig$:

$$CS_i(n_j).fw = CS_i(n_j).fw + 1$$

Initially, each node n_i assigns the value 1 to $ech_i(n_j)$, lq_{ij} and lq_{ji} for each new neighbor n_j. From equation 6, the initial reputation $Dr_i(n_j)$ is also 1. To track nodes' reputation, we introduce the idea of *reputation checkpoints* (RC). A reputation checkpoint is a point in time at which a node updates the reputation of its neighbors. At this time, nodes also save their current communication state in their *reputation records*. Thus, at any time, the reputation record of a node n_i, noted RR_i, contains a copy of the communication state of n_i at the time of its last reputation checkpoint. We will use the notations $RR_i.nbc$ for the field nbc of RR_i and $RR_i(n_j).f$ for field f of $RR_i(n_j)$.

At the time of a reputation checkpoint, a node n_i first estimates the echo ratio of each neighbor n_j as well as the link quality, in both directions, with n_j since the last reputation checkpoint. n_i then updates the value of its neighbors' reputation according to these values. Let $ech_i^{'}(n_j)$, $lq_{ij}^{'}$, and

$lq_{ji}^{'}$ be, respectively, the values of the echo ratio of n_j and the link quality of the links (n_i,n_j) and (n_j,n_i) since the last reputation checkpoint. n_i derives these values as follows:

$$ech_i^{'}(n_j) := \frac{CS_i(n_j).fw - RR_i(n_j).fw}{CS_i.nbc - RR_i.nbc} \quad (7)$$

$$lq_{ij}^{'} := \frac{CS_i(n_j).rnr + CS_i(n_j).rrnbc - RR_i.(n_j).rnr - RR_i(n_j).rrnbc}{CS_i(n_j).ns + CS_i.nbc - RR_i.(n_j).ns - RR_i.nbc} \quad (8)$$

$$lq_{ji}^{'} := \frac{CS_i(n_j).nr + CS_i(n_j).nbc - RR_i(n_j).nr - RR_i(n_j).nbc}{CS_i(n_j).rns + CS_i(n_j).rnbc - RR_i(n_j).rns - RR_i(n_j).rnbc} \quad (9)$$

To update the values of $ech_i(n_j)$, lq_{ij}, and lq_{ji}, we use a recurrent model that both captures the "effect of history" and takes into consideration the more recent changes. Specifically, Let RC_t and RC_{t+1} be two consecutive reputation checkpoints. Let $ech_i^t(n_j)$, lq_{ij}^t, and lq_{ji}^t be, respectively, the values of $ech_i(n_j)$, lq_{ij}, and lq_{ji} at the time of a reputation checkpoint RC_t. n_i calculates $ech_i^{t+1}(n_j)$, lq_{ij}^{t+1}, and lq_{ji}^{t+1} as follows:

$$ech_i^{t+1}(n_j) := (1-f).ech_i^t(n_j) + f.ech_i^{'}(n_j) \quad (10)$$

$$lq_{ij}^{t+1} := (1-f).lq_{ij}^t + f.lq_{ij}^{'} \quad (11)$$

$$lq_{ji}^{t+1} := (1-f).lq_{ji}^t + f.lq_{ji}^{'} \quad (12)$$

where f, called the *fading* factor, is such that: $0 \leq f \leq 1$. n_i then calculates the new value of n_j's direct reputation using equation (6).

2.1.2 Indirect Reputation

The indirect reputation of a node n_j as perceived by a node n_i reflects n_j's routing behavior as reported to n_i by its neighbors. We note this reputation $Ir_i(n_j)$. Initially, n_i sets the $Ir_i(n_j)$ to a default value ϵ, $0 \leq \epsilon \leq 1$. To receive new feedbacks about n_j from its neighbors, n_i broadcasts a message `ReputationRequest(`n_j`)`. Each node n_k that wants to answer n_i's request broadcasts a message `ReputationReport(`n_j`,` $Dr_k(n_j)$`)` that contains n_j's direct reputation as perceived by n_k. Typically, this message is received by several nodes (not only n_i, the sender of the `ReputationRequest` message). When a node n_l receives n_k's message, it updates its indirect reputation as follows:

$$Ir_l(n_j) = (1-f).Ir_l(n_j) + f.Dr_k(n_j)$$

Nodes may also send reputation reports without requests. When a node n_i detects that a neighbor node n_j's direct reputation $Dr_i(n_j)$ has gone under a threshold θ, it broadcasts a message `ReputationReport(`n_j`,` $Dr_i(n_j)$`)` to its neighbors.

2.1.3 Aggregate Reputation

The aggregate reputation or, simply, the reputation of a node n_j as perceived by a node n_i captures n_j's behavior from the perspective of both n_i and their common neighbors. Node n_j's reputation according to n_i, noted $R_i(n_j)$, is derived as an aggregate of n_j's reputation as calculated directly by n_i and n_j's reputation as calculated by their common neighbors. When n_i updates either of $Dr_i(n_j)$ or $Ir_i(n_j)$, it also updates $R_i(n_j)$ as follows:

$$R_i(n_j) := \alpha.R_i(n_j) + \beta.Dr_i(n_j) + \gamma.Ir_i(n_j)$$

where: $\alpha + \beta + \gamma = 1$

2.2 Path Reliability Evaluation

The second phase in TARP enables each node to determine an approximate value of the reliability of the paths to the sink starting at each of its neighbors. Each node may then determine the next hop of the most reliable path to the sink. For each node n_i, we will call n_i's *parent* n_i's neighbor that is at the other end of the first hop of the most reliable path to the sink. When a node has to send or route a message to the base station, it simply sends it to its parent. Routing in TARP then translates into determining path reliability.

To evaluate path reliability, we first derive an expression for path reliability in terms of node cooperation and link quality. Let (n_i,n_j) be a pair of nodes and let $\mathcal{P}(n_i, n_j)$ be the set of possible paths from n_i to n_j and let $\phi \equiv n_i, n_{p_0}, n_{p_1}, ..., n_{p_m}, n_j$ be a path in $\mathcal{P}(n_i, n_j)$. We call node n_i and node n_j, respectively, path ϕ's *start node* and *end node*. The link (n_i, n_{p_0}) is called path ϕ's *initial hop*. We define the *reliability* of the path ϕ, noted $\rho(\phi)$, as the probability that messages routed from n_i through ϕ be delivered to n_j. We will note the path(s) with the highest reliability from node n_i to node n_j: $\mathcal{P}_\rho(n_i, n_j)$. The purpose of TARP is to enable each node n_i to determine the initial hop of $\mathcal{P}_\rho(n_i, n_{bs})$, the most reliable path that starts at n_i and ends at the base station.

To quantify path reliability, we first define the concept of *node cooperation*:

Node Cooperation: The cooperation of a node n_i is defined as the probability that n_i forwards messages on behalf of its neighbors. We note n_i's cooperation $CP(n_i)$.

Assuming event independence, $\rho(\phi)$ may be written as follows:

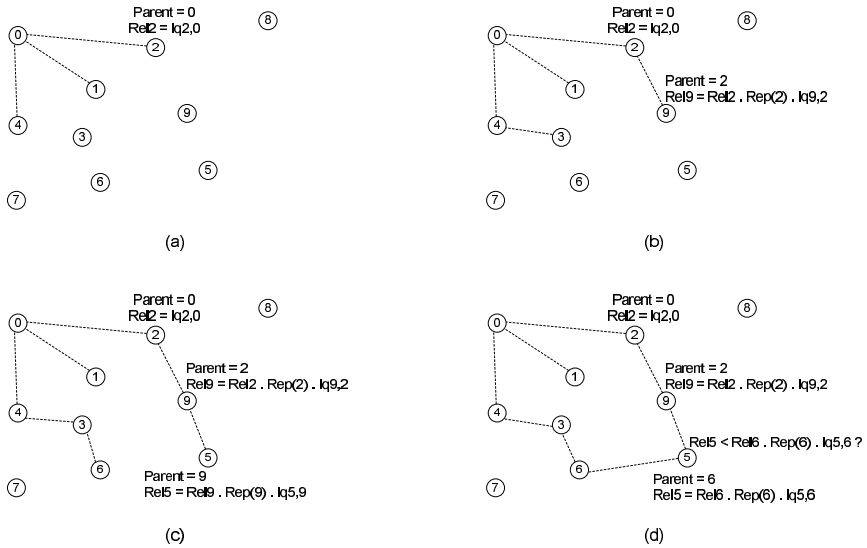


Figure 1. An Example of trust-based routing

$$\rho(\phi) = (\prod_{k=p_0}^{k=p_m} CP(n_k)).lq_{n_i n_{p_0}}.(\prod_{k=0}^{k=m-1} lq_{n_{p_k} n_{p_{k+1}}}).lq_{n_{p_m} n_j} \quad (13)$$

Ideally, if node n_i is able to evaluate $\rho(\phi)$, it will be able to make perfect routing decisions. This, however, is obviously not feasible in the context of SANETs. First, a node may not be able to assess distant nodes' cooperation and links' quality. Second, a node n_i may not be able to determine the cooperation value even for a neighbor n_j since this value depends on traffic between other nodes and n_j that n_i may not even hear. To enable nodes to estimate the reliability of the paths leading to the base station, we approximate node cooperation in terms of their routing reputation as perceived by individual nodes. Equation 13 may now be rewritten as follows:

$$\rho(\phi) = lq_{n_i n_{p_0}}.R_i(n_{p_0}).(\prod_{k=0}^{k=m-1} lq_{n_{p_k} n_{p_{k+1}}}.R_{p_k}(n_{p_{k+1}})).lq_{n_{p_m} n_j} \quad (14)$$

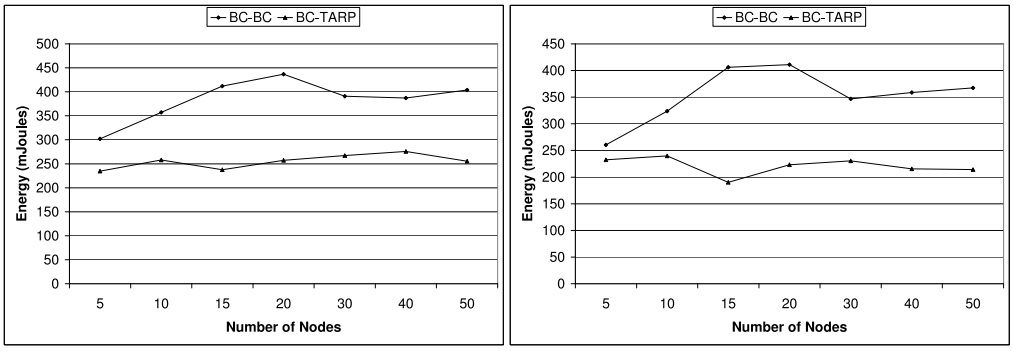
TARP maintains two values at each node n_i: *Rel* and *parent*. *Rel* is the reliability of the most reliable path to the sink. We will note this value $n_i.Rel$. *parent* is the identity of the neighbor that is at the end of the first hop in the most reliable path to the sink. We will note this value $n_i.parent$ Nodes cooperate to derive path reliability backward from the sink as follows:

- Each node initializes *parent* to the identity of its first neighbor and *Rel* to a default reliability value.
- Each node n_i appends the value $n_i.Rel$ to each outgoing message m. We will note this value $m.Rel$.
- A node n_i that has the base station as an immediate neighbor directly evaluates, lq_{i0} the quality of the link (n_i,n_0) between n_i and the sink. n_i then sets the value of $n_i.Rel$ to lq_{i0}.
- When a node n_j receives from a neighbor n_i a message m, it updates its variable Rel as follows:

$r := lq_{ji}.R_j(n_i).(m.Rel);$
if $r > Rel$
{
$parent := n_i;$
$Rel := r;$
}

Figure 1 illustrates the operation of TARP. In Figure 1.a, nodes 1, 2 and 4 have determined that the sink (node 0) is their neighbor. Node 2 assigns the value $lq_{2,0}$ to Rel and considers 0 its parent. In Figure 1.b, node 2 sends a message to node 9. Node 9 takes node 2 as its parent and sets its Rel to $Rel_2.R_9(n_2).lq_{9,2}$. In Figure 1.c, node 9 sends a message to node 5 which then takes node 9 as its parent and sets its Rel to $Rel_9.R_5(n_9).lq_{5,9}$. finally, in Figure 1.d, node 5 hears from node 6 and determines that $Rel < Rel_6.R_5(n_6).lq_{5,6}$. It then takes node 6 as its new parent and sets variable Rel to $Rel_6.R_5(n_6).lq_{5,6}$.

3 Implementation



(a) Radio Energy Consumption

(b) CPU Energy Consumption

Figure 2. Energy Consumption in TARP

```
command result_t send(TOS_MsgPtr m);
event result_t msgReceived(TOS_MsgPtr m);
```

Figure 3. TARP's nesC Interface

We implemented TARP as a routing layer on top of TinyOS 1.1.15 [14, 15]. TARP's nesC interface has one command (`send`) and one event `msgReceived` (Figure 3). A node calls TARP's `send` command to send (or forward) a message to the base station. When TARP receives a message from lower layers, it forwards it to the local node's parent and signals the event `msgReceived` so that the local node is notified of the event.

Communication in TARP is implemented using TinyOS's two interfaces `BareSendMsg` and `ReceiveMsg`. TARP is notified of the reception of messages via `ReceiveMsg`'s event `receive`. When TARP receives a message, it first reads the message's CRC to check whether it is correct. If the message is corrupt (CRC = 0), TARP simply discards the message. If the message is valid, TARP sends it to the local node's parent.

TinyOS's basic communication (using `BareSendMsg` and `ReceiveMsg`) is not reliable. Messages that are not successfully sent are not retransmitted. We therefore implemented a reliable communication layer that uses a FIFO queue `MsgQueue` and a timer `TransmitTimer` that triggers the (re)sending of the message at `MsgQueue`'s head. If the sender successfully sends the message, the message is removed from `MsgQueue`. If not, another attempt to send the message is made the next time `TransmitTimer` is fired.

TARP reduces energy consumption by using a simple power control mechanism. Immediately after TARP successfully sends a message, it makes the two following calls:

```
call CC1000Control.SetRFPower(0x00);
call CC1000StdControl.stop();
```

The first call ensures that power leak is minimized. The second call powers down the radio. To further reduce power consumption, TARP makes the following call immediately after it processes each incoming message:

```
call PowerState.cpuState(POWER_SAVE);
```

This call puts the node's CPU in TinyOS's power save mode.

4 Evaluation

In this section, we present our experimental evaluation of TARP. We will focus on two aspects: energy consumption and scalability. To evaluate TARP, we developed an evaluation benchmark that uses the PowerTOSSIM simulator [13] integrated in the TinyOS package. The benchmark enables a wide spectrum of simulation scenarios.

We considered a typical scenario where the base station submits a query requesting that all nodes that may sample an attribute t send t's value to the base station. The nodes that are able to sample the attribute t are randomly selected. The query message is broadcast to all the nodes in the network. we considered two alternatives to get the results to the base station. In the first case, nodes simply sample the attribute t and send the result message to the base station through broadcasts. Intermediary nodes simply forward the result message through broadcasts until the result message reaches the base station. In the second case, TARP is used to route the results back to the base station. The experiment is halted after a constant delay from the time the base station submits the query. Figure 2 shows the average energy consumption for all nodes in the two scenarios. The scenario labeled "BC-BC" corresponds to the first case where both queries and results are broadcast. The scenario labeled "BC-TARP" corresponds to the second case where queries are broadcast and results are sent back to the base station

using TARP. The figure shows that, in terms of radio energy consumption, BC-TARP outpeforms BC-BC by a percentage varying from 23% to 43%. In terms of CPU energy consumption, BC-TARP outpeforms BC-BC by a percentage varying from 7% to 71%.

5 Related Work

While used in some previous research on routing in the context of mobile ad hoc networks (MANs) and P2P networks, the concepts of reputation and trust have not been extensively investigated in the context of sensor networks. In this section, we overview some of the research related to exploiting reputation and trust in the context of MANs and P2P networks.

5.1 Trust in Mobile Ad hoc Networks

An issue in a mobile ad hoc network is that the nodes do not belong to a single authority [8]. This obviously poses the problem of stimulating cooperation, i.e., giving the incentive for nodes to cooperate, e.g., by forwarding each other's messages, for the overall good of the network. In [11], a reputation system is proposed where each node monitors its neighbor to check whether it actually forwards others' traffic. If it does not, it is considered as an uncooperative node and this reputation is propagated throughout the network. The authors propose two tools: a watchdog that identifies uncooperative nodes, and a pathrater that selects routes that avoid uncooperative nodes. In [2], the authors present a roubust reputation system for MANs. The system aims at detecting misbehaving nodes that are source of false positive and negative ratings. The system The basic idea in this system is that each node maintains a reputation rating and a trust rating about every other node of interest. From time to time, first-hand reputation information is exchanged with others. A node accepts second-hand reputation information only if it is not incompatible with the current reputation rating. In [8], the authors focus on the formal analysis of the properties that mobile nodes use to update and agree on the reputation of other mobile nodes. They also study the correlation between the speed of reputation propagation and the convergence speed of reputation agreement. In [6], the authors propose a reputation-based incentive scheme, called *SORI*, that encourages packet forwarding and disciplines selfish nodes. The scheme is based on three components: neighbor monitoring, reputation propagation, and punishment. Each node maintains the number of packets that each neighbor forwards and requests to be forwarded. Through this monitoring, a node is able to locally determine its selfish neighbors. This local reputation of being selfish is then propagated by the monitoring node so that the selfish node becomes punsihed by all of its neighbors.

5.2 Trust in Peer-to-Peer Networks

A peer-to-peer (P2P) network is a network in which any node (i.e., peer) may interact *directly* with any other node to achieve a given goal. In a typical interaction, a peer a *requests* a resource r from another peer b that *provides* the resource r to peer a. The basic use of reputation in P2P networks is to identify reputable nodes so that non-reputable nodes may be prevented from affecting the system. The basic idea is to deploy reputation systems that provide reliable reputation information about peers. A peer can then use this information in decision making, e.g., who to download a file from [5, 3, 7]. Examples of applications based on this concept include P2P anonymity systems (e.g., anonymous remailers [4]) and P2P resource sharing networks. In [3], the authors propose an approach to security in P2P information sharing environments where peers keep track of other peers' reputation and may share their reputation information with other peers. A distributed polling algorithm is proposed for reputation sharing. The algorithm has two phases. In the polling phase, a peer p polls its peers to inquire about the reputation of selected providers that offer the requested content. This is achieved by broadcasting a `Poll` message requesting each peer's opinion about the providers. All peers may answer p's request. The poller p then uses the opinions received from *voters* to make its decision about where the source to use to download the requested content. In [7], the authors present a system, called *EigenTrust*, that computes and publishes a global reputation rating for each node in a network using an algorithm similar to Google's *PageRank* [12]. In [16], the authors propose a Bayesian network-based trust model and a method for building reputation based on recommendations in P2P networks. The evaluation of the model using a simulation of a P2P file sharing system shows that the system where peers communicate their experiences (recommendations) outperforms the system where peers do not share recommendations. In [10], the authors study the advantages and disadvantages of resource selection techniques based on peer reputation. They analyze the effect of limited reputation information sharing on the efficiency and load distribution of a P2P system. A noticeable result of the study is that limited reputation sharing can reduce the number of failed transactions by a factor of 20. In [17], the authors present *PeerTrust*, a system that uses reputation to estimate the trustworthiness of peers. *PeerTrust* uses a general trust metric that combines a set of basic trust parameters and adaptive factors in computing trustworthiness of peers, namely, the feedback a peer receives from other peers, the total number of transactions a peer performs, the credibility of the feedback sources, transaction context factor, and the community context factor. In [5], the authors propose a reputation system for decentralized unstructured P2P content sharing networks like Gnutella. The purpose is to help well-

reputed peers make decisions about who to serve content to and who to request content from. In [1], the authors address the issue of scalability in reputation-based trust management in P2P systems. They propose to use a scalable access method, called P-Grid, to store trust information that peers have about their past interactions with each other. Another issue in P2P systems is the tension between anonymity and the support for reputation. In [9], the authors compare two identity infrastructures for P2P resource-sharing environments. The first uses a centralized login server that ties nodes' network pseudo-identities to their real-world identities. In the second approach, each node generates its own identity. The authors show that the first approach provides better support for reputation management by preventing nodes from changing identities. They also show that the second approach provides a higher level of anonymity. In [4], the authors report on their experience in designing reputation mechanisms for anonymous remailing and anonymous publishing systems. In anonymous publishing, for example, a set of servers form a P2P network where they may anonymously publish documents. The rule is that any server that publishes must also provide some disk space to store other servers' material for a certain time. The authors use a reputation system that prevents servers from cheating by dropping data early.

6 Conclusion

We presented TARP, a trust-aware routing protocol for sensor-actuator networks. We implemented TARP on top of TinyOS. Empirical results show that TARP achieves significant improvements in terms of energy consumption and scalability. Our future work aims at developing a new context-awareness approach that we plan to integrate in TARP. The new context-aware TARP would further exploit data semantics and enable routing schemes particularly suitable to prioritized traffic. Specifically, the new protocol would enable applications where urgent information has to be routed to the base station before other, less important, traffic.

References

[1] K. Aberer and Z. Despotovic. Managing Trust in a Peer-2-Peer Information System. In *Proc. of the ACM Conference on Information and Knowledge Management (CIKM)*, pages 310–317, Atlanta, Georgia, 2001.

[2] S. Buchegg and J. Y. L. Boudec. A Robust Reputation System for Peer-to-Peer and Mobile Ad-hoc Networks. In *Proc. of the 2nd Workshop on the Economics of Peer-to-Peer Systems*, Cambridge, Massachusetts, June 2004.

[3] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. Managing and Sharing Servents' Reputations in P2P Systems. *IEEE Trans. on Knowledge and Data Engineering*, 15(4), July/August 2003.

[4] R. Dingledine, N. Mathewson, and P. Syverson. Reputation in P2P Anonymity Systems. In *Proc. of the Workshop on Economics of Peer-to-Peer Systems*, Berkeley, California, June 2003.

[5] M. Gupta, P. Judge, and M. Ammar. A reputation system for peer-to-peer networks. In *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video (NOSSDAV)*, pages 144–152, New York, NY, USA, 2003. ACM Press.

[6] Q. He, D. Wu, , and P. Khosla. SORI: A Secure and Objective Reputation-based Incentive Scheme for Ad-hoc Networks. In *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC)*, Atlanta, GA, USA, March 21-25 2004.

[7] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust Algorithm for Reputation Management in P2P Networks. In *Proc. of the 12th International World Wide Web Conference (WWW)*, pages 640–651, Budapest, Hungary, 2003.

[8] Y. Liu and Y. R. Yang. Reputation Propagation and Agreement in Mobile Ad-Hoc Networks. In *Proc. of the IEEE Wireless Communication and Networks Conference (WCNC)*, New Orleans, LA, USA, March 2003.

[9] S. Marti and H. Garcia-Molina. Identity Crisis: Anonymity vs. Reputation in P2P Systems. In *Proc. of the 3rd International Conference on Peer-to-Peer Computing*, pages 134–141, Linkping, Sweden, 2003.

[10] S. Marti and H. Garcia-Molina. Limited Reputation Sharing in P2P Systems. In *Proc. of the 5th ACM Conference on Electronic Commerce*, pages 91–101, New York, NY, USA, May 2004.

[11] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad hoc Networks. In *Proc. of the 6th International Conference on Mobile Computing and Networking (MobiCom)*, pages 255–265, Boston, Massachusetts, USA, 2000.

[12] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[13] V. Shnayder, M. Hempstead, B. rong Chen, G. W. Allen, and M. Welsh. Simulating the power consumption of large-scale sensor network applications. In J. A. Stankovic, A. Arora, and R. Govindan, editors, *SenSys*, pages 188–200. ACM, 2004.

[14] TinyOS. `http://www.tinyos.net`.

[15] TinyOS Tutorial. `http://www.tinyos.net/tinyos-1.x/doc/tutorial`.

[16] Y. Wang and J. Vassileva. Trust and Reputation Model in Peer-to-Peer Networks. In *Proc. of the 3rd IEEE International Conference on Peer-to-Peer Computing*, pages 150–157, Linkoeping, Sweden, September 2003.

[17] L. Xiong and L. Liu. PeerTrust: Supporting Reputation-based Trust for Peer-to-Peer Electronic Communities. *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, 16(7):843–857, July 2004.