J. Parallel Distrib. Comput. 72 (2012) 475–488

Contents lists available at SciVerse ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Fine grained load balancing in multi-hop wireless networks

Alessandro Mei, Natascia Piroso*, Bruno Vavala

Department of Computer Science, Sapienza University of Rome, via Salaria 113, 00125 Rome, Italy

ARTICLE INFO

Article history:
Received 18 August 2010
Received in revised form
11 December 2011
Accepted 23 January 2012
Available online 28 January 2012

Keywords:
Multi-hop wireless networks
Sensor networks
Routing
Load balancing
Security
Energy efficient

ABSTRACT

In this paper we address the problem of local balancing in multi-hop wireless networks. We introduce the notion of proactive routing: after a short pre-processing phase in which nodes build their routing tables by exchanging messages with neighbors, we require that nodes decide the relay of each message without any further interaction with other nodes. Besides delivering very low communication overhead, proactive routing protocols are robust against some well known active attacks to network routing. In this framework, we develop a proactive routing protocol that is able to balance the local load. Experiments show that our protocol improves network lifetime up to 98% and that it delivers a network that is more robust against attacks that have the goal of getting control over a large part of the network traffic.

1. Introduction

The interest on multi-hop ad hoc wireless networks has been largely growing in the last decade. In large multi-hop wireless networks nodes communicate from source to destination using other nodes as relays. Important examples of these kinds of networks are sensor networks—networks made of low-end tiny devices that are equipped with a battery and a radio transmitter. Sensors are usually deployed in large numbers within a geographic area in order to perform some common task such as monitoring environment properties, like temperature, humidity, presence of people, etc. The scientific literature on multi-hop wireless networks is vast. A large part of it is devoted to the problem of routing—how to forward packets hop by hop from source to destination in an efficient way.

Typically, routing protocols try to route packets through the shortest path from source to destination so as to obtain small delays and short traveled distances [28]. However, shortest-path routing is not always the appropriate option as it can often cause the appearance of large congested areas or the occurrence of local hot-spots. The obvious consequence is that some nodes become significantly more loaded that others, thus quickly discharging their battery. This is a critical concern, especially in wireless sensor networks. When a node dies, its neighbors experiment an immediate increase in traffic volume due to the necessity of handling routes that were previously covered by the deceased node. The congestion can then spread to a large area thus limiting the network's lifetime and efficiency. To avoid such an undesired effect, researchers have designed a number of routing protocols that attempt to distribute traffic evenly among nodes. Load balancing protocols can assure longer operational functionality and a more graceful decay of the network.

Moreover, congestion also raises security related issues, that are important concerns in both wireless sensor networks and multi-hop wireless networks. For instance, the disrupting effects of jamming become more severe when the attack is directed to an overloaded region of the network. In addition, if traffic is concentrated in small areas, the attacker is further favored, as jamming can disturb a large number of communications at a limited cost [16].

In this paper, we introduce the notion of pro-active routing and apply it to load balancing. A proactive routing protocol performs all the decisions regarding how to route packets in the network in a short pre-processing phase at the start of the network operations. Then, routing tables and routing mechanisms are frozen, and every node decides how to select the next relay only looking at the destination of the packet and at its internal information, without any further interaction with other nodes. The rationale of proactive routing is to have a routing protocol that is both efficient and robust against some security attacks to the network. We assume that the adversary is inactive for a short period of time at the start of the network operations. This is a common assumption (see [27] for an example among many others)—for a short period of time we are able to use expensive security techniques, including physical surveillance, that avoid the presence of malicious activity.

* Corresponding author.
E-mail addresses: mei@di.uniroma1.it (A. Mei), natascia.piroso@gmail.com, piroso@di.uniroma1.it (N. Piroso), vavala@di.fc.ul.pt (B. Vavala).

0743-7315/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2012.01.010

Afterwards, we expect the network to be robust against the adversary.

In the literature, load balancing is usually obtained by using reactive routing protocols. There is a very large number of such protocols—we review only a few of them in Section 2. These are protocols that dynamically use local information, like the remaining energy of neighboring nodes, to repeatedly choose, time after time, an appropriate relay for the goal of getting an even load balance. However, this information can be easily spoofed, altered, or replayed. In [11], these attacks are described and are shown to be hard to stop. In particular, by altering routing information it is usually easy to perform the *sinkhole attack* [11], where the adversary's goal is to attract a large amount of traffic in a particular area through a compromised node. It is enough that the nodes under the control of the adversary claim to have a high level of remaining energy. Pro-active routing techniques, like the one we introduce in this paper, is a simple and clean way to protect the network against these attacks since no routing information is exchanged, except in the very first pre-processing phase. A positive side-effect of pro-active routing is that it is extremely efficient, since the overhead due to control information is reduced to virtually zero.

In this framework, our target is to obtain a fairly distributed traffic on a strictly local basis. The problem is far from trivial since we cannot rely on any dynamic information to achieve our goal. The solution we propose in this paper takes advantage of pro-active routing and allows an extraordinary result on network lifetime. As our experiments will show, we are able to deliver 98% more messages when comparing our protocol with standard, state of the art shortest path routing protocols.

The rest of the article is organized as follows. In Section 2, we briefly survey the related work. In Section 3, we discuss the main general advantages achievable by routing pro-actively. At the beginning of Section 4, we show how pro-active and reactive routing specifically act on nodal load. We then distinguish two levels at which load unbalance is generated, the macroscopic and the microscopic level. We discuss the opportunity of separately designing load balance at the two levels and introduce, in Section 4.1, a tool that allows to isolate and cut out the macroscopic unbalance. In Section 5, we address the microscopic load balance problem through the notion of competence and use this to propose, in Section 6, a complete algorithm for competence balancing. Finally, in Section 7, simulation results show that competence balancing induces load balancing and enables improvements in terms of load variance, network lifetime and protection against adversaries.

2. Related work

In the last decade wireless ad-hoc networks have drawn the attention of the research community. The capability of these networks of scaling gracefully has permitted a wide range of applications. In particular, there is now a great interest in sensor networks [1], as the limited capacity of these devices raises a number of challenging issues. One of the most relevant obstacles to the exploitation of sensor networks is the limited battery supply so that many efforts have been made in order to design energy aware protocols [2]. In this paper we focus on the achievement of a fair work load distribution among nodes, as it has been broadly confirmed that load balancing has a considerable impact on network lifetime [10].

The definition of appropriate routing schemes is obviously one of the key solutions to energy saving. Many authors propose greedy routing schemes as an approximation of shortest path routing, often coupled with alternate awake-sleep states to a given duty cycle [3,23,28]. In [4,24,25] the problem of load unbalance has been tackled with the use of *reactive routing* protocols. Routes are determined on-demand by forwarding messages to neighbors with higher residual energy, thus smoothing the energy level among neighbors. Some authors have also used multiple paths for load balancing [19]. However, it is shown in [5] that a good load distribution can be obtained only by using a large number of paths, otherwise results remain much similar to those obtained with single path shortest routing.

Recently, it has observed that shortest path routing leads to several problems in terms of load induced by the relay traffic. In [13], it is shown that straight-line routing in a network deployed in a convex surface generates an irregular distribution of load among nodes. As a matter of fact, centrally located nodes handle many more packets than nodes situated in peripheral areas of the network, and nodes with a large Voronoi set handle many more packets than nodes with a smaller one.

The analysis of theoretical results reported in [13] also reveals that load unbalance has two independent components: one at a *macroscopic*, global level (distance of nodes from the center of the network), the other at a *microscopic*, local level (size of Voronoi sets, that is local distribution of nodes). The authors first give an estimate of nodal load (defined as the number of packets served at a node) in the case of straight line routing and then show that nodal load is a function of the traffic pattern, of the nodes' Voronoi cells and of the location of nodes in the network.

The most efficient protocols addressing load unbalance at the macroscopic level are pro-active in nature. Given a source–destination couple, those solutions envisage the use of longer routing paths. The idea is to replace shortest paths with alternative routes, computed in such a way to improve load balance at the global level. This task cannot be accomplished only using locally available data, so that reactive procedures appear to be ineffective to this end.

In [14,20] wireless nodes on a disk are projected on a sphere and routing decisions are made according to great circle distances. Shortest paths defined on the sphere define paths on the given disk and those paths avoid the central congested area, redistributing load among nodes. A similar approach is adopted in [16]. Nodes in a square network surface are mapped to a torus by the use of some randomness and of an easily computable function. Using a simple metric defined on the virtual toric space, each node chooses as next hop relay the neighboring node closer to the destination. Authors of [10] also address the crowded center problem and present a general framework for analyzing traffic load and routing. They also propose an heuristic solution were routing on a disk is carried out by switching from the circle traversing the source to the circle traversing the destination.

3. Pro-active vs. reactive routing and security implications

All of the above-mentioned works focus on the problem of load balancing in multi-hop wireless networks. They define an appropriate routing strategy in order to evenly distribute work load among all nodes, so that overall energy endurance can be preserved. As suggested, these routing protocols fall under two distinguished paradigms: routes are either defined reactively or pro-actively.

In the first case, nodes periodically inquire their neighbors about their energy remainder, and load balance among nodes is approached by routing to neighbors proving high power supply. The network's response to unexpected topology changes is swift, but blind to the inner cause of the appearance of congested areas, such as the position of node in a centrally located area. A considerable drawback of this method is the high communication overhead.

On the other hand, when routing is defined in a pro-active fashion, the information that a node requires to decide which neighbor to rely on does not change during a network's activity—each node is only supposed to know what is the shape of the surface that carries the network and its position within such surface. This data is then employed in the computation of an *a priori* known function which finally returns a convenient relay node, without further interaction with neighboring nodes.

The search for energy efficiency can possibly backfire on the safety of the system, if the network's security is not taken into account at routing design time [11]. Reactive routing protocols are vulnerable to those attacks that abuse the protocol itself. When energy preservation is a crucial demand, it is not surprising that an attacker can do serious damage by forging energy related information. An attacker may be able to identify itself as a network node and therefore communicate with other nodes in the network. Once a node is captured, the adversary can make it look attractive to neighbors by declaring a very high battery supply. All neighboring nodes will then preferably choose the malicious node as next-hop relay. The result is that most of the messages handled by network nodes located within the communication range of the attacker will be absorbed by the attacker itself, who may decide to drop certain messages thus invalidating the overall accuracy of data. These attacks are particularly vicious since the attacker can make a lot of damage by controlling one or few nodes.

Pro-active routing proves to be more robust against this sort of attack. The pre-processing stage is designed in order to prevent an unfair use of energy among nodes by preventively establishing how nodes have to be used in the routing phase. After pre-processing, nodes no longer inquire their neighbors about their energy remainder, therefore ignoring any malicious neighbor advertising high battery levels. During pre-processing we assume that major security measures are adopted, including physical surveillance, if necessary. It is only in this stage that the network is vulnerable to attacks based on energy scarcity. When pre-processing ends, all nodes stop exchanging energy-related information, thus making unfeasible for an attacker to intervene on the construction of routing tables. Of course, pro-active protocols cannot do much if the adversary can physically stop a large part of the network nodes. These attacks are not the focus of this work.

There is a number of attacks that can be prevented, if nodes are relieved from investigating their neighbors energy state or their position [9,11,18]. We list a few of them.

Sinkhole attack A malicious node declares to have full or high energy in order to drain all the traffic coming from its neighbors. Damage to the network can be caused by subsequently dropping all (*blackhole behavior*) or part (*selective forwarding*) of the received packets. Data can be also altered before retransmission.

Wormhole attack A captured node owns a private and direct link with another distant captured node. Packets arriving at one of the two malicious nodes can be dysfunctionally diverted to the other malicious node, thus altering the expected communication system.

Sybil attack A captured node forges multiple identities thus resulting in different places simultaneously. In this case, the malicious node promotes himself by increasing its chance of being chosen, as it appears to be very often on the direction of the packet's destination.

Both wormhole and sinkhole attacks can be made more severe by adopting the sinkhole strategy. Pro-active routing is an effective counter-measure that can be used to stop the sinkhole attack (nodes cannot attract more messages since routing tables are frozen and the injection of malicious information regarding energy and positioning does not influence routing decisions). In this way, we also mitigate the possible damage caused by the wormhole attack and the sybil attack, whose severity also depends on the possibility of launching a sinkhole attack.

A limitation to the robustness of a network under a pro-active routing protocol could be the necessity of joining new nodes to the existing ones in order to replace exhausted nodes. In this case, the pre-processing procedure must be repeated and the adversary can then actively participate, therefore having the opportunity to heavily influence routing decisions. However, if the initial pre-processing procedure is well designed, then nodes end their battery charge almost simultaneously. In this case, there is no sense in linking the old nodes to the new ones. It is more convenient to initiate a new network with a new set of cryptographic keys.

The above observations suggests what has been the direction of our effort. This work introduces a novel approach to the load balancing problem and proposes a pro-active new protocol to resolve it, in order to take advantage of the benefits in terms of network security and network performance that this technique offers.

4. Macroscopic and microscopic level

A common assumption, when focusing on the dynamics ruling traffic distribution, is to consider nodes uniformly distributed over the surface of a convex polygon, usually a circle or a square. We follow this assumption, but also anticipate that our balancing procedure is not merely constructed for a uniform deployment of nodes, as it can be applied unchanged to any distribution of nodes.

We also assume that each pair of nodes has the same chance of being the sender and the final receiver of a communication and we call this a *uniform traffic communication model*. This type of traffic distribution has been widely adopted in network simulations, for instance in the analysis of the capacity of a wireless network, in the study of routing optimality and for testing security properties [7,8,26]. Moreover, there exist many applications that generate a uniform traffic pattern, such as [17], a protocol for the selection of witnesses for a node's location in sensor networks, that uses geometric probability to detect replicated nodes. Another example of uniform traffic generation is given by the use of mobile sinks in sensor networks [6,15]. Sensors typically send data to a limited number of common sink collectors, so that nodes in the proximity of a sink undergo heavy forwarding. If a sink is able to move among several anchor points (for instance around the network periphery), the role of bottleneck nodes is then distributed over time, so that the distribution of nodal load can approximate the uniform distribution.

The use of a uniform communication pattern, together with shortest path routing, causes centrally located nodes to be much more congested than others, as most of the routes will traverse the central area of the network. From this point of view, it is clear that the amount of traffic traversing a certain node strongly depends on its position inside the network. We call this level of sight the *macroscopic level* and underline that load unbalance observable at this scale is a consequence of the global network geometry. This problem has been investigated and a few efficient solutions have been proposed recently [14,16,20]. Nevertheless, *local unbalance remains*. With some sort of magnifying glass, we can imagine to zoom in on different parts of the network. Most likely, the resulting observation frames are going to look very different from one another. Mutual location of nodes, density and surface coverage are some of the varying general aspects resulting in hot-spots scattered over the network. We will show, later in this paper, that local irregularities have a conspicuous impact on load balance and therefore on network lifetime.

Opposite to the macroscopic level, we then define the *microscopic scale*, corresponding to a single node and its immediate neighbors. When we talk about load unbalance produced at the microscopic level, we are therefore looking at load differences among single nodes. At this scale, the problem is often addressed through reactive protocols that define routes dynamically on demand according to remaining energy at each node, however, we already discussed the security problems of this choice.

In this paper, we attack the problem of designing a pro-active routing protocol that guarantees load balancing of relay traffic at the microscopic level. As far as we are concerned, this is the first work proposing a pro-active procedure preventing congestion on a local basis. In this way, we deliver a low overhead, load-balancing routing protocol that is robust against some well-known attacks to routing security.

4.1. *Virtually routing*

Motivated by the observations made so far, we propose a completely distributed pro-active procedure for local load balancing. We assume that a pro-active procedure acting at the macroscopic level adjusts unbalance due to the global geometry of the network. Our target is then to tackle the residual problem of local load unbalance. We will show that our procedure shows beneficial effects on both energy endurance and security implications.

As far as we are concerned, local load unbalance has been addressed exclusively through reactive routing protocols. The drawbacks of reactive routing in terms of energy consumption and network security have been already discussed. In our procedure, nodes do not need to keep track of the residual energy of their neighbors. They only need to gather local information on their neighborhood before messages start to be sent. Each node uses the gathered data to set the forwarding schedule it will subsequently follow. Once the routes are set, they never change again, unless changes in the network topology occur. Moreover, schedules are constructed in such way that nodal load remains equally distributed during all network activity. As a matter of fact, known reactive protocols act on load unbalance only after its occurrence and the adopted countermeasures need some time to produce their effect.

The parallel use of certain pro-active protocols dealing with congestion produced at the macroscopic level allows us to isolate unbalance rising at the microscopic level. Hereunder we make this statement more precise.

Let $\mathcal{N} = (\mathcal{V}, \mathcal{E})$ be an undirected graph. $\mathcal{N}$ represents a multi-hop wireless network with the following features. $\mathcal{N}$ is deployed within a plane convex polygon $\mathcal{S}$ and two nodes $u, v \in \mathcal{V}$ can directly communicate if and only if $\delta(u, v) \leq r$ where δ is the Euclidean distance and r is the communication range, common to all nodes, so that $\mathcal{E} = \{(u, v) \in \mathcal{V} \times \mathcal{V} \mid u \neq v \text{ and } \delta(u, v) \leq r\}$.

Among all routing protocols that are commonly employed for load balancing control, *virtual routing* models have carved out a space for themselves. In those models each node receives, through an appropriate mapping function, new space coordinates which take the name of *virtual coordinates*. Virtual coordinates can be points in the original space, in our case $\mathcal{S}$, or in another convenient space. Whichever is the destination space, we call this *virtual space* and denote it by Σ. Packet shipping then takes place as in geographic routing protocols, except that nodes act as if they were located at their virtual positions. Once routing in the virtual space Σ is carried out, routing in the original space $\mathcal{S}$ is immediately given. It is sufficient to trace routes defined on Σ back into the original space S.

Suppose that a message has to be sent from $s \in \mathcal{V}$ to $d \in \mathcal{V}$ and that geographic routing is the shortest path strategy fixed for routing in the virtual space. Let $f : \mathcal{S} \to \Sigma$ be some easily computable function used to obtain virtual coordinates from the real position of nodes in the network. Finally, let us denote the real position of a generic node u with P_u. Then, s and d are virtually located at $f(P_s)$ and $f(P_d)$ respectively. Let $N(u)$ be the set of nodes adjacent to u. Starting from $u = s$, each relay u decides the next hop to destination selecting the neighbor v such that $\delta_\Sigma(f(P_v), f(P_d)) = \min_{w \in B \subseteq N(u)} \delta_\Sigma(f(P_w), f(P_d))$, where δ_Σ is some fixed distance defined on Σ. Then relays thus determined are the same relays that will route the message in the real network. In other words, routes determined on Σ are traced back in the original space S.

Note that we have imposed the choice of the next hop within some subset B of $N(u)$. In fact, relays selected by the routing protocol in the virtual space must be actual neighbors in the original network, otherwise virtual paths will not correspond to existing paths in the real network.

Various examples of virtual routing can be found in literature. For instance, in [14,20] routing is virtually made over a sphere, while in [16] paths are determined using a torus as virtual space.

Why is virtual routing interesting for us? If the mapping function and the virtual space Σ are carefully chosen, the following properties are ensured:

1. uniform distribution of (virtual) nodes over Σ;
2. absence of hot spots in Σ referable to its particular shape;
3. a uniform communication pattern on S results in a uniform communication pattern on Σ.

Our idea is to directly consider the network to be deployed on a virtual space producing the listed features. Obviously, we are not saying that the network is actually located over some odd virtual space! Instead we assume that an appropriate pro-active virtual protocol is adopted, so that we can perform local load balancing when nodes are mapped into the virtual space, before any route is fixed. This expedient enables us to focus exclusively on unbalance due to local topology difference among neighborhoods, thus leaving out unbalance due to the global geometry of the surface of deployment. For our convenience the choice for the virtual space has fallen on a toric surface, as shortest paths and distances can be easily visualized on it. However, the same reasonings hold for other constructions.

An example of a mapping function f from the square $S = [0, 1] \times [0, 1]$ to the torus $\Sigma = [0, 2] \times [0, 2]$ is the following. Let $P = (x, y) \in S$ and let p_x and p_y be two independent variables assuming value 0 or 1 as the result of fair coin toss. Then the new virtual coordinates are:

$$f_x(P) = p_x \cdot x + (1 - p_x)(2 - x)$$
$$f_y(P) = p_y \cdot y + (1 - p_y)(2 - y).$$

Under this mapping, Properties 1–3 hold, as proved in [16].

Routing over a torus

Let $T = [0, 1] \times [0, 1]$ be a two-dimensional torus constructed from a unit square by pasting the opposite edges together with no twist. An *unfolded* torus then appears exactly as a square, but points on the vertical edges with same y coordinate coincide as well as points on the horizontal edges with same x coordinate. Given $P = (x_P, y_P)$, $Q = (x_Q, y_Q) \in T$ we define on T the distance δ_T as follows

$$\delta_T(P, Q) = \sqrt{\delta_x^2 + \delta_y^2}$$

where

$$\delta_x = \min\{|x_Q - x_P|, 1 - |x_Q - x_P|\}$$
$$\delta_y = \min\{|y_Q - y_P|, 1 - |y_Q - y_P|\}.$$

Note that if the network deployed on T presents a grid topology, then it is obvious that shortest path routing among all possible pairs of nodes results in a perfect balance in nodal load.

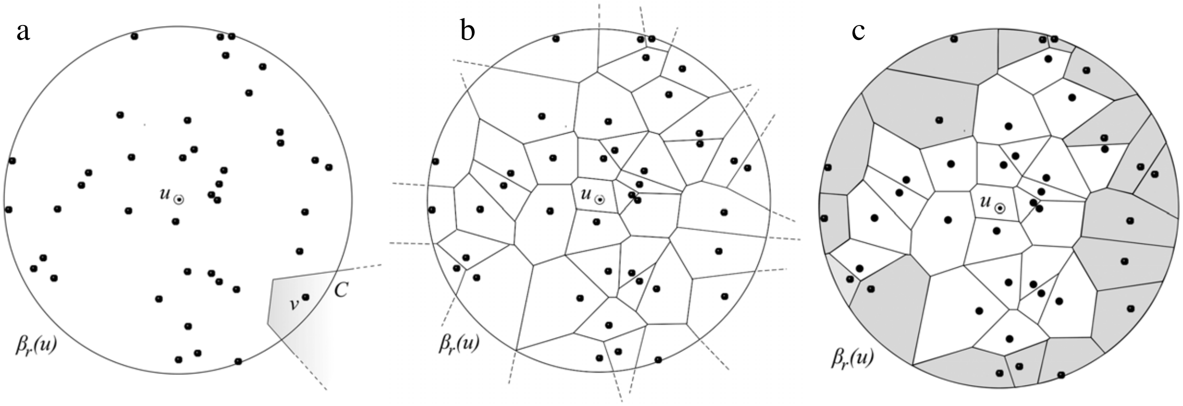


Fig. 1. *Example of active-neighborhood construction*. (a) Black dots represent nodes within the transmitting range of the central node u. The gray area is the locus of points that are closer to node v than to any other neighbor of u. In other terms, it is the Voronoi cell of node v in the Voronoi diagram obtained by using neighbors of u as sites. — (b) Outline of the Voronoi cells of nodes in $N(u)$. — (c) Neighbors whose Voronoi cell is cut by the transmitting bound $\beta_r(u)$, here highlighted in gray, are taken as u-active neighbors. Such nodes result to be closer to some point in the network surface T outside $\beta_r(u)$ and can be appointed by u as relays in a compass routing procedure.

5. Competence balancing

In the following, we briefly describe the compass routing method and then introduce the notion of *competence*. We use competences to give an approximate estimate of nodal workload and claim that an even distribution of competences among nodes leads to a fair workload distribution. In other words, the compass routing scheme is optimized through the notion of competence, in order to achieve a uniform traffic load distribution. The detailed procedure is described in Section 6 and the foundation of our intuition has been empirically tested. Simulation results can be found in Section 7.

5.1. Shortest path routing

A similar alternative to geographic routing is given by *compass routing* [12]: a node u that has to transmit a packet to a destination d chooses the next hop to destination by picking among its neighbors the node v that minimizes the angle formed by v, u and d. If we compare paths between source and destination obtained by routing geographically, with those defined by compass routing, the latter appear to be closer to the source–destination connecting straight line. This achievement is obtained at the cost of a higher number of hops, as the selected node v can be very close to u. To avoid brief hops we restrict the choice of relay v to a subset of the set $N(u)$ of neighbors of u. We call such subset $N_A(u)$ and refer to elements in it as *u-active nodes*, as these are the only neighbors that can be appointed by u as relays for any possible packet shipping. If $\beta_r(u)$ is the circle of radius r centered at u that defines the transmitting range of u, then it is obvious that u-active nodes should be selected among those neighbors of u that are located in the proximity of $\beta_r(u)$. In the following we describe our choice for the active set $N_A(u)$.

Given $u \in \mathcal{V}$ and $v \in N(u)$, consider the locus $\mathcal{C}$ of points in $\beta_r(u)$ that are closer to v than to any other neighbor of u. If the network is sufficiently dense, $\mathcal{C}$ is either the empty set or a continuous line, precisely an arc of $\beta_r(u)$. As shown in Fig. 1(a), by using nodes in $N(u)$ as sites to construct a Voronoi diagram, arc $\mathcal{C}$ can also be viewed as the intersection of $\beta_r(u)$ with the Voronoi cell of v. Referring to this formalism, we can define our u-active set as follows.

Definition 1 (*Active Neighborhood*). Given $u \in \mathcal{V}$, consider the *Voronoi diagram* constructed using nodes in $N(u)$ as *Voronoi sites*. A node $v \in \mathcal{V}$ is u-active if

(i) v is a neighbor of u;
(ii) the intersection of v's *Voronoi cell* and $\beta_r(u)$ is non-empty.

Fig. 1 shows how to identify a node's active-neighborhood using the previous definition. Note that the set of neighbors that u selects as relays when geographic routing is adopted is a subset of the set of u-active nodes above defined and eventually agrees with it.

5.2. Estimating nodal work-load

Now we have fixed compass routing as routing strategy for packet delivery, we are ready to introduce a tool to locally estimate the potential load of a node in order to balance energy use among nodes basing decisions exclusively on neighborhood evaluation. For every node $v \in N(u)$ let $r_u(v)$ be the segment-radius of $\beta_r(u)$ passing through v. If u is located in (x_u, y_u), the coordinates of a point $v \in N_A(u)$ can be expressed in the form $(x_u + r\cos\alpha_v, y_u + r\sin\alpha_v)$, with $\alpha_v \in [0, 2\pi)$ and $r \in [0, 1)$. We use the α_v angle to order u-active nodes *counterclockwise*.

Definition 2 (*Angle of Competence*). Given $u, v \in \mathcal{V}$, if $v \in N_A(u)$, let $v_{\text{prev}}, v_{\text{next}} \in N_A(u)$ be the u-active nodes that precede and follow v in the given counterclockwise order. The angle of competence (or simply *competence*) of v with respect to u is then defined as

$$\theta_u(v) := \begin{cases} \dfrac{\phi}{2} + \dfrac{\psi}{2} & \text{if } v \in N_A(u) \\ 0 & \text{otherwise} \end{cases}$$

where ϕ is the angle formed by $r_u(v_{\text{prev}})$ and $r_u(v)$, and ψ the angle located between $r_u(v)$ and $r_u(v_{\text{next}})$.

The above definition is described graphically in Fig. 2. Intuitively the angle of competence of a node v with respect to u expresses a measure of the quantity of packets arriving at u and that u will forward to v. In other words, the number of times that u will choose v as a relay. In fact, if the line that connects u to the destination of a certain packet passes through the competence of v and we follow the compass routing procedure, then the packet has to be handled by v. We have extended the definition of angle of competence with respect to a node u also to those nodes v that are not adjacent to u and to neighbors of u that are not u-active by setting $\theta_u(v) = 0$.

For simplicity, from now on we will confuse the concept of angle with the concept of magnitude of an angle each time the disambiguation can be clearly established from the context. The following is a key definition for our balancing procedure.

Definition 3 (*Total Competence*). The total competence of a node $v \in \mathcal{V}$ is the sum

$$L(v) = \sum_{u \in \mathcal{V}} \theta_u(v)$$

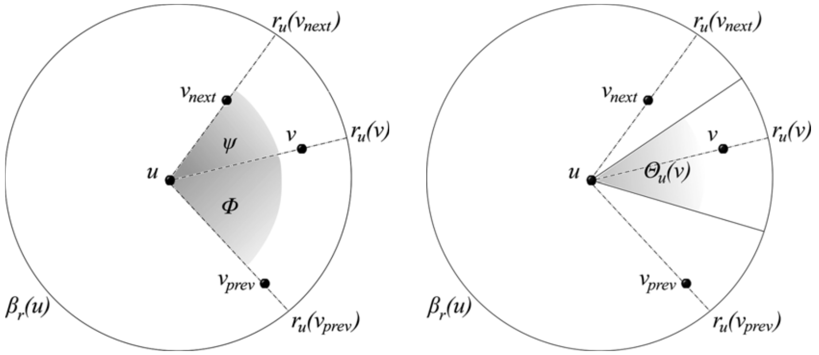


Fig. 2. *Angle of competence assignment.* Node v belongs to the active neighborhood of node u. u-active nodes are ordered counterclockwise: the angle of competence $\theta_u(v)$ of v with respect to u depends on the position of the two u-active nodes, v_{prev} and v_{next}, preceding and following v in the fixed ordering. Angles ψ and ϕ are enclosed between the radius-segments passing through v_{prev}, v and v_{next} (left figure). $\theta_u(v)$ is then obtained merging together the lower half of ψ and the upper half of ϕ (right figure).

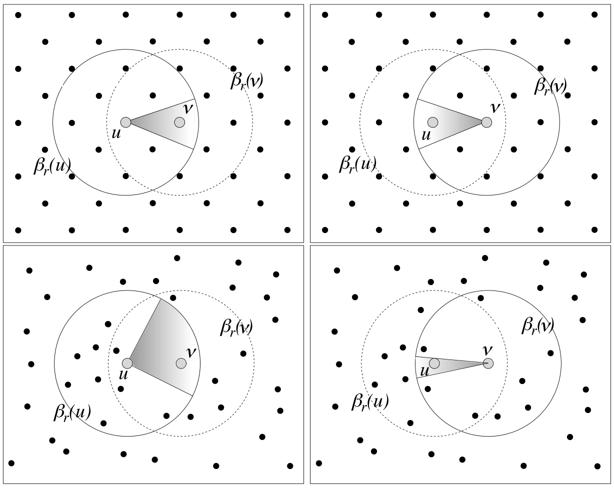


Fig. 3. *Mutual competence assignment between node couples.* In a grid network topology (top figures), nodes are symmetrically placed so that $\theta_u(v) = \theta_v(u)$ for all $u, v \in \mathcal{V}$. When nodes are uniformly distributed (bottom figures), there is no such correspondence and the total competence of a node may largely vary around the round angle.

of all angles of competence assigned to v by other nodes in the network.

$L(v)$ then expresses an estimate of how frequently v is used as a relay by its neighbors.

5.3. Towards local load balancing

Let us suppose for a moment that the given network presents a grid topology. Then work-load is perfectly distributed among nodes and the following identity holds.

Lemma 1. *If $\mathcal{N}$ is a grid then*

$$L(v) = \sum_{u \in \mathcal{V}} \theta_u(v) = \sum_{u \in \mathcal{V}} \theta_v(u) = 2\pi \quad \forall v \in \mathcal{V}.$$

Proof. The first equality is the definition of total competence of a vertex. Because of the typical symmetries of a grid, it follows that a node u makes a node v responsible for packets going into a certain direction if and only if v makes u responsible for packets going into the opposite direction (see Fig. 3, top). Therefore we have that $\theta_u(v) = \theta_v(u)$ for all $u, v \in \mathcal{V}$. Finally, for every network node, each possible shipping direction must be covered so that the angles of competence assigned from a node to its active neighbors sum up to the round angle. □

Roughly speaking this means that each node is assigned angles of competence whose sum is equal to 2π, which is also the total angle assigned from a node to its neighbors.

As shown in Fig. 3 (bottom, left and right), when nodes are uniformly distributed at random in the network space, the regularity expressed by Lemma 1 is no longer assured. As a matter of fact, the volume of packets traversing a node from opposite incoming or outgoing directions does not agree in general; actually, it can differ greatly is spite of the assumption of a uniform traffic pattern. This is due to the local irregular distribution of neighbors as well as on the impact of irregular distribution of source–destination pairs. Unfortunately, we cannot intervene locally on the latter aspect without expensive searches in the network and our interest remains that of constructing a light and completely distributed procedure for load balancing.

Not being able to relocate nodes, we tried to artificially recreate a fair distribution of packets among different directions through the achievement of the equi-distribution of the angles of competence. In particular, our idea is to approach as closely as possible the main feature observed in grid-networks: the equality, stated in Lemma 1, of the full circle to the sum of competences received by a node from its neighbors. We will show that the approximate optimization of this factor allows to obtain excellent results in the balancing of nodal load.

6. An approximate procedure for competence balancing

As we have seen, when nodes are distributed at random on a continuous surface, many local topological irregularities occur. Compass routing, as well as geographic routing, generates on a torus a balanced load at a macroscopic level, while *punctual load* analysis, i.e., load per single node, shows a considerable variance of values.

Our aim is to intervene on punctual load by reducing the variance around the mean load. To do so we modify for each node u the set of nodes delegated by u to act as relays and therefore modify the proportion of arc in $\beta_r(u)$ that those neighbors serve. In compass routing the set of nodes that are capable to receive packets from u are the u-active nodes. We recall that in compass routing u sends a packet to the u-active neighbor v that minimizes the angle between $r_u(v)$ and the line connecting u to the destination. This line intersects circle $\beta_r(u)$, so each active node is responsible for an arc of circumference. We act on the span of the arc of circumference for which an active node is responsible by reducing or enlarging it and we eventually modify the active set in order to gain variance reduction in load distribution. The key to variance load control is actually competence balancing. As a matter of fact, simulation results will show that the attempt to reduce the variance of quantity $L(\mathcal{V}) := \{L(v) \mid v \in \mathcal{V}\}$ leads to a reduction of load variance.

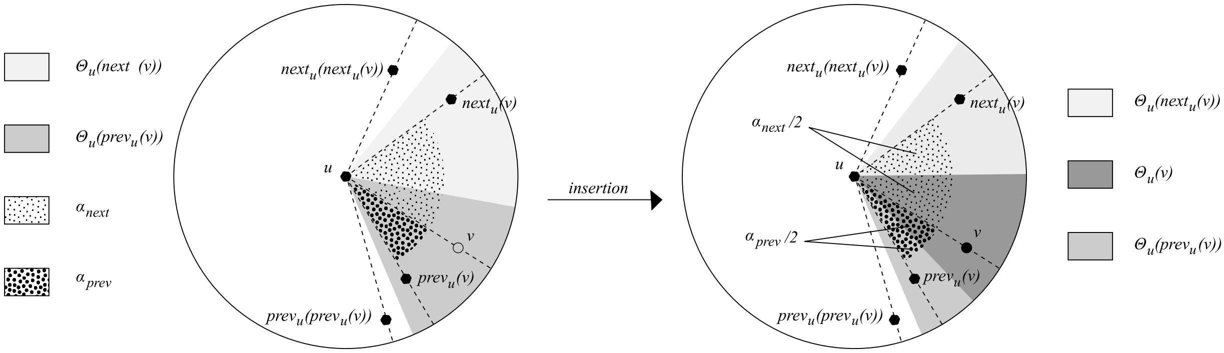


Fig. 4. *Insertion operation*. Node v is u-eligible but not u-active. Subsequent u-active nodes near v in the fixed counterclockwise order are drawn with a full black dot, while competences for u-active neighbors are highlighted in gray. If v is made u-active, half of the angle formed between the radius-segments passing through v and $\text{prev}_u(v)$ is subtracted from.

Proposition 1. *The mean value of $L(\mathcal{V})$ is 2π whether nodes are uniformly distributed or they form a grid topology.*

Proof. For a grid network the result follows immediately from Lemma 1. However, in both case, every node assigns all of its round angle so that globally all nodes will have assigned an angle of $2\pi \cdot n$. □

Moreover, Lemma 1 proves that $L(\mathcal{V})$ is exactly equal to 2π for a grid configuration, so that it follows from the above proposition that the variance of $L(\mathcal{V})$ is zero in this case. Fig. 3 shows instead that a uniform distribution of nodes does not necessarily lead to a zero mean value for $L(\mathcal{V})$. As argued at the end of Section 5.3, our intent is to artificially *make* the given network $\mathcal{N}$ *look like* a grid, imposing the equality between the total competence of a node and the round angle and therefore reducing the variance of the total competence of a node. Simulation results in Section 7 will show that this strategy actually leads to a reduction of load variance, as expected.

6.1. Neighborhood manipulation: insertion and removal operations

Our load balancing procedure is essentially constructed over two basic operations, *node removal* and *node insertion*. Both operations have effect on the active set of nodes. Loosely speaking their behavior can be outlined as follows.

1. By node insertion, a non-active node is selected and appointed as active if its insertion in $N_A(u)$ reduces the variance of $L(\mathcal{V})$.
2. Node removal is carried out to eliminate a node from $N_A(u)$ in the case that the variance of $L(\mathcal{V})$ decreases after such removal.

As earlier mentioned, when using the compass routing procedure it is advisable to restrict the choice of a next hop relay to a subset of the involved neighborhood, or hops might be too short. Therefore, as we will operate on the set of active nodes by the insertion and removal of nodes, it is necessary to fix a reference set of the nodes that can be added to the current active set. To this extent, let us denote with $N_A^{(0)}$ the initial and largest possible set of active nodes and let $N_A^{(0)}$ be constructed as specified in Definition 1. $N_A(u)$, instead denotes the current u-active set. Any node to be inserted in $N_A(u)$ will be chosen among nodes in $N_A^{(0)}(u)$: any other node in the neighborhood of u is to be considered too close to the departure node u. A node in $N_A^{(0)}(u)$ will be also called *u-eligible*. A u-eligible node is also u-active if and only if it belongs to $N_A(u)$. To summarize,

$N_A^{(0)}(u)$ is the reference set of eligible nodes – nodes that u can appoint as relays – and is fixed once and for all as in Definition 1;

Operation 1 – Insertion

```
Require: $u \in \mathcal{V}, v \in N_A^{(0)}(u) \setminus N_A(u)$
1: $\alpha_{prev}$ = angle between $r_{prev_u(v)}$ and $r_v$
2: $\alpha_{next}$ = angle between $r_{next_u(v)}$ and $r_v$
3: $L'(prev_u(v)) = L(prev_u(v)) - \frac{\alpha_{next}}{2}$
4: $L'(next_u(v)) = L(next_u(v)) - \frac{\alpha_{prev}}{2}$
5: $L'(v) = L(v) + \frac{\alpha_{prev}+\alpha_{next}}{2}$
6: $Var' = Var - \frac{(L(prev_u(v))-2\pi)^2+(L(next_u(v))-2\pi)^2+(L(v)-2\pi)^2}{n}$
7: $Var' += \frac{(L'(prev_u(v))-2\pi)^2+(L'(next_u(v))-2\pi)^2+(L'(v)-2\pi)^2}{n}$
8: if $Var' < Var$ then
9:     $v \rightarrow N_A(u)$
10:    update $L(prev_u(v)), L(next_u(v)), L(v), Var$
11: return
```

Fig. 5. *Operation* 1. The procedure evaluates and eventually executes the insertion of an eligible node in a current active set.

$N_A(u)$ is the set of current active nodes for u and must be a subset of $N_A^{(0)}(u)$.

As usual, nodes in $N_A^{(0)}(u)$ are considered numbered according to the counterclockwise order specified at the beginning of Section 5.2. Given a node $v \in N_A^{(0)}(u)$, let $\text{prev}_u(v)$ and $\text{next}_u(v)$ be the u-active nodes immediately preceding and following v according to the fixed progressive numbering and let us use symbol $\prec$ to precise the order relation between two eligible nodes.

Suppose that one has to check if the insertion in $N_A(u)$ of an eligible non-u-active node turns out to be convenient and let Var denote the current variance for competences. The operation is then carried out as described by the algorithm in Fig. 5 and by the drawing in Fig. 4. The insertion of v in $N_A(u)$ has effect on the values of $L(v)$, $L(\text{prev}_u(v))$ and $L(\text{next}_u(v))$, enlarging the total competence of v and reducing the total competence of $\text{prev}_u(v)$ and $\text{next}_u(v)$. In particular, v becomes responsible for a portion of packets passing through u. Lines 3, 4 and 5 account for this task. The term 2π that appears from line 6 is the mean value of $L(\mathcal{V})$. Notice, at lines 6 and 7, that updating Var, a global variable, only requires the knowledge of local variables that node u can trace from its neighbors. It is sufficient to remove the addends that involve the total competence of $\text{prev}_u(v)$ and $\text{next}_u(v)$ and to add the terms that are related to the inclusion of v in $N_A(u)$. At lines 8–10, the inclusion of v in $N_A(u)$ is made final when competence variance is reduced.

It might be worth to explain the assignments at lines 3 and 4. If node v is not u-active, the angle enclosed between $r_{\text{prev}_u(v)}$ and $r_{\text{next}_u(v)}$ measures $\alpha_{\text{prev}} + \alpha_{\text{next}}$. Let us call γ the angle between $r_{\text{prev}(\text{prev}_u(v))}$ and $r_{\text{prev}_u(v)}$, and σ the angle between $r_{\text{next}_u(v)}$ and

```
Operation 2 – Removal
Require: $u \in \mathcal{V}, v \in N_A(u)$
$\alpha_{prev}$ = angle formed by $r_{prev_u(v)}$ and $r_v$
$\alpha_{next}$ = angle formed by $r_{next_u(v)}$ and $r_v$
$L'(prev_u(v)) = L(prev_u(v)) + \frac{\alpha_{next}}{2}$
$L'(next_u(v)) = L(next_u(v)) + \frac{\alpha_{prev}}{2}$
$L'(v) = L(v) - \frac{\alpha_{prev}+\alpha_{next}}{2}$
$Var' = Var - \frac{(L(prev_u(v))-2\pi)^2+(L(next_u v)-2\pi)^2+(L(v)-2\pi)^2}{n}$
$Var' += \frac{(L'(prev_u(v))-2\pi)^2+(L'(next_u(v))-2\pi)^2+(L'(v)-2\pi)^2}{n}$
if $Var' < Var$ then
    remove $v$ from $N_A(u)$
    update $L(prev_u(v)), L(next_u(v)), L(v), Var$
return
```

Fig. 6. *Operation* 2. The procedure evaluates and eventually executes the removal of a node from a current active set.

$r_{\text{next}(\text{next}_u(v))}$. Then we can compute the competence of $\text{prev}_u(v)$ and $\text{next}_u(v)$:

$$\theta_u(\text{prev}_u(v)) = \frac{\gamma}{2} + \frac{\alpha_{\text{prev}} + \alpha_{\text{next}}}{2}$$
$$\theta_u(\text{next}_u(v)) = \frac{\alpha_{\text{prev}} + \alpha_{\text{next}}}{2} + \frac{\sigma}{2}.$$

If v is inserted in $N_A(u)$ competences are updated as follows:

$$\theta'_u(v) = \frac{\alpha_{\text{prev}}}{2} + \frac{\alpha_{\text{next}}}{2}$$
$$\theta'_u(\text{prev}_u(v)) = \frac{\gamma}{2} + \frac{\alpha_{\text{prev}}}{2} = \theta_u(\text{prev}_u(v)) - \frac{\alpha_{\text{next}}}{2}$$
$$\theta'_u(\text{next}_u(v)) = \frac{\alpha_{\text{next}}}{2} + \frac{\sigma}{2} = \theta_u(\text{next}_u(v)) - \frac{\alpha_{\text{prev}}}{2}.$$

In a similar way we check for node removal. We suppose that $\text{prev}_u(v)$, $\text{next}_u(v)$ and v are u-active nodes that are responsible for three adjacent arcs of $\beta_r(u)$. The procedure described in Fig. 6 decides whether it is convenient to remove v from $N_A(u)$ and eventually redistributes $\theta_u(v)$ to $\text{prev}_u(v)$ and $\text{next}_u(v)$.

6.2. Operations scheduling

Operations are scheduled in a distributed manner, but for the sake of simplicity in Fig. 7 we give the pseudo-code description of the procedure in a sequential mode.

All nodes must define the angle of competence to be assigned to their neighbors. We preferred not to adopt a fixed order of assignment of competences to a neighborhood in order to answer for a fair assignment flow. Consider a node u and the initial assignment of competences defined by $N_A^{(0)}(u)$. u associates to each node $v \in N_A^{(0)}(u)$ a state variable e such that

$$e_u(v) = \begin{cases} 0 & \text{if } u \text{ needs to evaluate the convenience} \\ & \text{ of an insertion or removal operation on } v \\ 1 & \text{otherwise.} \end{cases}$$

Initially $e_u(v) = 0$ for all $v \in N_A^{(0)}(u)$ and for all $u \in \mathcal{V}$. u selects at random a node v marked with $e_u(v) = 0$. If v is currently u-active, by performing Operation 2, u decides whether to maintain or erase v from $N_A(u)$. Clearly v is actually removed if and only if its removal decreases competence variance. Similarly, if v is non-u-active, u checks for the convenience of restoring v in $N_A(u)$, throughout Operation 1. Once the operation is executed, $e_u(v)$ is set to 1.

```
Competence Balancing Algorithm
Require: $\mathcal{N} = (\mathcal{V}, \mathcal{E})$
 1: for all $u \in \mathcal{V}$ do
 2:     $N_A(u) = N_A^{(0)}(u)$
 3:     $f(u) = 0$
 4:     for all $v \in N_A(u)$ do
 5:         $e_u(v) = 0$
 6:         compute $\theta_u(v)$
 7: for all $v \in \mathcal{V}$ do
 8:     compute $L(v)$
 9: $Var = Var(L)$
10: for all $u \in \mathcal{V} \mid f(u) = 0$ ( pick $u$ at random) do
11:     for all $v \in N_A^{(0)} \mid e_u(v) = 0$ ( pick $v$ at random) do
12:         if $v \in N_A^{(0)}(u) \setminus N_A(u)$ then
13:             carry out Operation 1
14:         else
15:             carry out Operation 2
16:         if $\theta_u(v)$ has been modified then
17:             for all $u' \mid v$ or $prev_u(v)$ or $next_u(v) \in N_A^{(0)}(u')$
                do
18:                 $f(u') = 0$
19:                 for all $w \in N_A^{(0)}(u')$ do
20:                     $e_{u'}(w) = 0$
21:         $e_u(v) = 1$
22:     $f(u) = 1$
23: return
```

Fig. 7. *Basic competence balancing procedure.*

An additional state variable is used to define the state of the evaluation tasks of each node. To this extent we set

$$f(u) = \begin{cases} 0 & \text{if } \exists\, v \in N_A^{(0)}(u) \mid e_u(v) = 0 \\ 1 & \text{otherwise.} \end{cases}$$

We initialize $f(u) = 0$ for all nodes in the network. Once all nodes have assigned the convenient angle of competence to their neighbors, we have $f(u) = 1$ for every $v \in \mathcal{V}$ and the procedure is complete.

When a node v receives a new competence assignment $\theta_u(v)$ from a neighbor u, the consequent change of $L(v)$ might create new opportunities of improvement in competence distribution, i.e., the opportunity to further decrease competence variance. Our competence balancing procedure is thoroughly described in Fig. 7 where the above mentioned additional checks are included in lines 16–20. The procedure selects at each round a node u such that $f(u) = 0$ at random and carries out the above described routines.

In the Competence Balancing algorithm a naïf strategy is adopted. As described in lines 16–20, each time the assignment of a node v changes, all nodes u' that contain v in $N_A^{(0)}(u')$ set the elements in $N_A^{(0)}(u')$ as to be evaluated, through the appropriate setting of the variable state e. This occurs whether or not the evaluation and assignment of the competence of these neighbors have been previously executed. Node u obviously knows that some neighbors are to be checked and sets $f(u) = 0$. This sub-routine may be optimized by reducing the number of neighbors set for checking as in the following proposition.

Proposition 2. *Suppose that the Competence Balancing algorithm has just modified the value of $\theta_u(v)$ by an insertion or removal operation and let u' be such that $v \in N_A^{(0)}(u')$. Then*

(i) *the only nodes $w \in N_A^{(0)}(u)$ that require $e_u(w)$ to be set to zero are such that*

$$\text{prev}_u(\text{prev}_u(v)) \preceq w \preceq \text{next}_u(\text{next}_u(v)).$$

(ii) *If v is u'-active, it is sufficient to set for checking only nodes $w \in N_A^{(0)}(u')$ such that*

$$\text{prev}_{u'}(v) = w \quad \text{or} \quad \text{next}_{u'}(v) = w.$$

If v is non-u'-active, no node in $w \in N_A^{(0)}(u')$ requires $e_{u'}(w)$ to be set to zero.

Proof. Suppose that a node v has been inserted back or removed from $N_A(u)$ for some $u \in \mathcal{V}$. Each time Operation 1 or Operation 2 are carried out on a node $v \in N_A^{(0)}(u)$, it is necessary by definition to modify the value of the competence of the two u-active nodes preceding and following v in the counterclockwise order established by u over its neighbors in $N_A^{(0)}(u)$. Those nodes are $\text{prev}_u(v)$ and $\text{next}_u(v)$. Therefore it makes sense to check only those nodes w for which either $\text{prev}_u(w)$ or $\text{next}_u(w)$ has changed as a consequence of the insertion of v. Then it is not difficult to see that such nodes are namely the nodes included between $\text{prev}_u(\text{prev}_u(v))$ and $\text{next}_u(\text{next}_u(v))$ in the counterclockwise order set by u. Checks on any other node in $N_A^{(0)}(u)$ are unnecessary and this proves (i).

On the other hand if u' is such that $v \in N_A^{(0)}(u')$ and $u' \neq u$, even less checks have to be imposed. Precisely, if v is non-u'-active, $\text{prev}_{u'}(v)$ and $\text{next}_{u'}(v)$ are unchanged for every $w \in N_A^{(0)}(u')$, so no additional checks are necessary.

On the contrary, if v is u'-active then it is necessary to mark $w \in N_A^{(0)}(u')$ for checking only if $\text{prev}_{u'}(w)$ or $\text{next}_{u'}(w)$ are equal to v, as v is the only u'-active node that has changed its competence. And this proves (ii). □

6.3. Transition to a distributed procedure: holdbacks, checks and termination

The complete transposition of the Competence Balancing Algorithm into a completely distributed procedure is almost immediate. This is due to the fact that each single node u maintains its own state variable e and the set of f variables related to its neighbors. When checking a node v marked with $e_u(v) = 0$ for eventual insertion or removal from its active set, u only needs to know the value of the current angle of competence of three of its neighbors, information that u can obtain with a simple 1-hop request message. Yet, it may be of use to point out the possibility of deadlocks. In order to avoid the corruption of data regarding the value of competences, it is necessary to impose a lock on the neighbors that receive a competence request. When a node u asks for the value of the angle of competence of a neighbor v for the execution of Operations 1 and 2, it also wants to be sure that no other node will change such value while he is evaluating. Therefore u will lock v to prevent it from giving information over his total competence to another inquiring node until he has sent an unlock message to v, eventually changing its total competence by a different assignment of $\theta_u(v)$. This procedure can obviously lead to deadlocks with nodes indefinitely waiting for information. Procedures against the occurrence of deadlocks, such as ordered labeling of nodes, are well known and able to avoid any possibility of deadlock in our case.

An issue that regards the distributed version of our algorithm as well as the centralized procedure, is *termination*. It is immediate to prove that the network always reaches a configuration such that no further insertion or removal operations lead to variance decrease. Should the number of operation be infinite, then there would exist two distinguished time units t_1 and t_2, $t_1 < t_2$, such that the same assignment of competences occurs. As the Competence Balancing Algorithm envisages the necessity of carrying out a new operation only if the variance of $L(\mathcal{V})$ decreases, then variance at time t_2 must be strictly smaller than variance at time t_1. But this a contradiction because at time t_1 the network presents the same configuration of time t_2.

6.4. Dealing with node failures

A node can fail for software or hardware faults, or as a consequence of malicious activity. In both cases, re-starting pre-processing may be considered and countermeasures for the acquisition of competence values should be adopted.

The communication of competence values is requested only for pre-processing. If a competence value is not available as a consequence of node failure, the failing neighbor is simply considered as non-existing and its competence is split between the two surviving adjacent active nodes. More precisely, on the basis of the insertion and removal operations, a node u containing the failing node v in its active neighborhood knows the angle of competence personally assigned to v and contributing to the definition of the total competence of v. Node u needs only to split this part of v's total competence among its residual active neighbors; the rest of the failing node's competence is redistributed from the nodes that have assigned it. Once the competence of failing node is re-assigned, pre-processing carries on regularly.

When node failure occurs after preprocessing, unless pre-processing is restarted, optimal competence balancing is disturbed. A node can detect no longer operating neighbors by requiring periodical hello messages (or simply ack messages) and eventually starts a re-distribution of its competences. However, in most of the network competence values are equally distributed, so that only very local countermeasures are sufficient in order to contain load inequalities and avoid an excessive surcharge of nodes adjacent to the failing one. In Section 7 we will see that the pre-processing stage only requires a number of operations per-node that is at most linear in the number of neighbors. Moreover, we believe that the cost of pre-processing is largely compensated by the suppression of any communication overhead for the exchange of information on energy reminders.

In our experiments we assumed that nodes do not fail, although our protocol works in case of failures, too. We stress that pre-processing may be restarted if needed, but it is crucial in this phase to protect the network with all the necessary security measures against external attacks in order to avoid the corruption of energy-related data.

7. Experimental results

The simulator. In this section we show the results obtained by the simulation of our local load balancing procedure by means of a high level simulator implemented in C++. We do not model communication interferences, but we do look at packet losses. The reason behind this choice is due to our interest in analyzing the algorithm behavior in order to confirm our intuition over a high bond between nodal load and the notion of angle of competence.

We model our network as a sensor network with limited capacity, in terms of battery supply, using data given in [22]: 324 000 mJ of total available node battery, 15.104 mJ for packet transmission and 7.168 mJ to receive a packet. Nodes are deployed on a unit square network with opposite sides glued together. Obviously no real network is deployed on such a surface, but this expedient allows us to analyze network congestion spots due exclusively to local topological features, thus ignoring hot spots caused by the well known and studied phenomenon of central congestion in convex plane surfaces for which many solutions have been already proposed.

Nodes are located uniformly at random over the network surface and they all have the same radial communication range, that is a radius of 0.1. We consider networks with 1000 and 2000 of nodes, hence nodes have on average approximatively 30 neighbors for experiments with 1000 nodes and approximatively 60 neighbors for experiments with 2000 nodes. In all analyzed data there is a steady improvement from the case of 1000 nodes and that of 2000 nodes and it is reasonable to expect improvements along with the increase of nodal density.

The communication system used for simulation changes according to the data information we are mining. We use either an *all pairs communication system* (Section 7.1), were each node sends a message/packet to every other node in the network, or a *uniform communication system* (Sections 7.2 and 7.3), were the source–destination couples are chosen uniformly at random. Note that the former merely represents the ideal convergence point of a uniform communication routine.

By our pre-processing procedure each node u defines the portion of communication range served by its neighbors. If the line connecting u to the destination passes through a certain arc, the node serving this arc will be chosen as relay. We then inject traffic into the network and follow the defined schedule for packet shipping.

Cost of routing. As a result of pre-processing, each node stores a small table containing only two numeric values for each of its neighbors. The two values identify the beginning and the end of the angle of competence of a neighbor. Nodes are already aware of their neighbors and their relative position, so that we are only storing a small additional information for each (active) neighbor. Moreover, the size of the additional information required from our protocol is comparable to the size of additional information required by reactive protocols, where instead it is the level of residual energy to be associated to each neighbor. On the contrary, after pre-processing the neighbor to be chosen for each direction is already known and never changes, while it should be decided from time to time when a reactive protocol is adopted. The small amount of data stored during pre-processing and the absence of routing decision as the networks begins to be operational make our protocol well suited for a context of resource scarcity.

Cost of pre-processing. More precise experiments have shown that the number of u-active neighbors for a node u is much smaller than the total number of neighbors of u. In a network with 2000 nodes, each node has on average 62.59 neighbors with 7.06 of standard deviation. The average size of the initial u-active neighborhood is 23.14, while at the end of the pre-processing phase the average size of the u-active neighborhood drops down to 12.03 with a standard deviation from the expected value of 4.81.

Pre-processing consists in the insertion and removal of appropriate nodes in the active set. The total number of insertion and removal operations is on average 14.7 per-node with a standard deviation from the expected value of 5.99. Each insertion or removal operation involves the selection of the node to insert or remove ($O(d)$, where d is the degree of the node), the computation of the new variance value (constant time), and the communication of the change in competence assignment to the node that has been possibly removed or inserted in the active set. This means that, in general, the total number of operations carried out by a node to make competences final is below the size of the initial u-active set N_A^0.

Terms of comparison. In all our experiments we compare the response of the *pre-processed* network with the case of a non-pre-processed identical network that undergoes the same traffic pattern. The scheduling of relays follows in this case the compass routing strategy. We chose this technique because it makes use of angles to determine the next hop to destination, as well as our balancing procedure does. Note that, as well as geographic routing, compass routing is a shortest path protocol as it greedily attempts to approximate the straight line connecting the two ends of a communication. We have also produced statistics with the use of geographic routing, but we omit them, as the difference with those obtained via compass routing is negligible. This is also confirmed in [21], where simulation results reveal that nodes select in the two schemes the same forwarding neighbor in over 99% of cases.

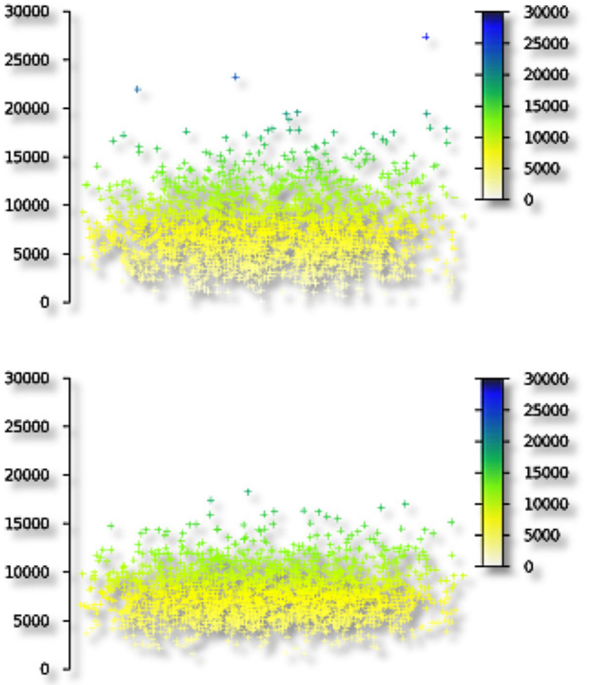


Fig. 8. *Achievement obtainable by the competence balancing procedure.* Nodes are plotted on the xy plane while the z axis carries the information over nodal load. The graphic is pictured facing the yz plane in order to show how the cloud of points obtained without pre-processing (top) gets compressed by competence balancing (bottom).

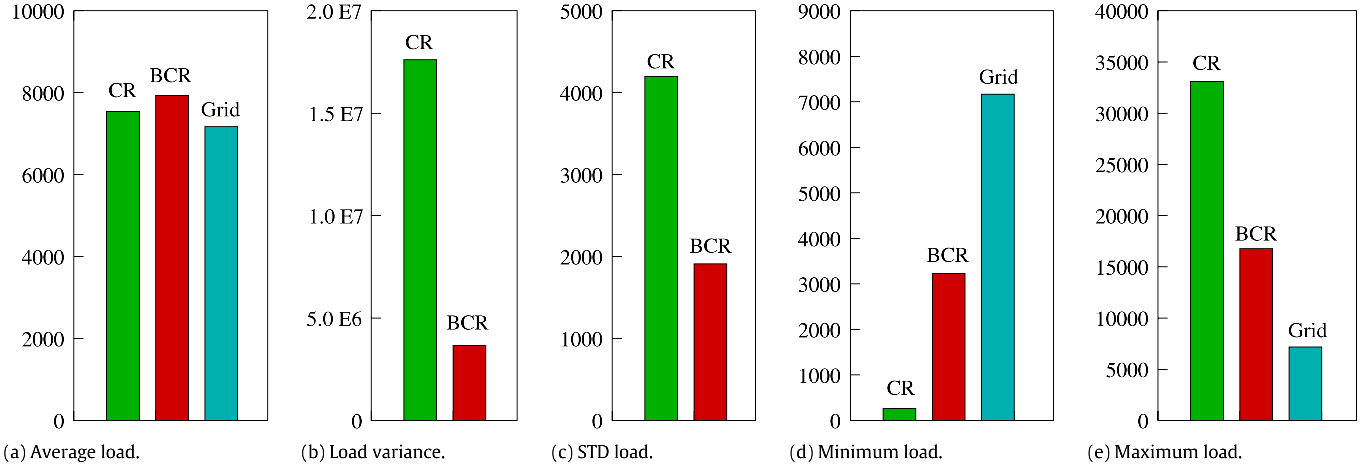
The other key comparison is made with an ideal configuration in terms of nodal load distribution: we take a regular grid network, whose nodes adopt a greedy strategy to forward packets, and inject traffic into it. When the all pairs communication system is adopted each node handles exactly the same number of packets, while with the uniform communication system we have near enough the same number of packets going through each node. Results over a regular grid serve as ideal bounds for the maximum increment achievable in load balancing. In the following we name the above described simulation scenarios as

CR	—for the simple shortest path strategy;
BCR	—for competence balancing pre-processing;
Grid	—for the ideal grid configuration.

The reported results refer to random networks with 2000 nodes and grids made up of 44 nodes on each row and column.

7.1. Statistics over punctual nodal load

The *load* of a node is the number of times that a node receives, receives and retransmits or simply transmits a packet. Even by eliminating the crowded center effect of shortest path routing typical of convex surfaces, the pure greedy routing strategy causes some nodes to be considerably more loaded than others. This is shown clearly in Fig. 8 – top – where isolated nodes are traversed



(a) Average load. (b) Load variance. (c) STD load. (d) Minimum load. (e) Maximum load.

Fig. 9. Punctual load statistics.

by a number of packets near to zero while other nodes nearly reach the level of 30 000 of handled packets. In the bottom figure, BCR shows instead a tighter concentration of points around the average load level.

This is even clearer in Fig. 9. At the price of a minimum increment of the average load (5%), we achieve a substantial 80% of variance decrement. Fig. 9(d) and (e) show how the nodes that in CR are barely used, see their exploitation increase in BCR; similarly, nodes overloaded by a pure greedy protocol are instead relieved in BCR.

Nodal load statistics also give an idea of the average and maximum stretch of a path. One can easily calculate that on average BCR routing increases the path length by less than 6%. It is extremely important to keep this percentage small as on realistic convex networks our competence balancing technique should be associated to a coarse-grained balancing procedure – to attack unbalance also at the macroscopic level – that alone brings with it a non-negligible path stretch.

7.2. *Network lifetime*

The small increment in average nodal load translates into a small increment in overall energy consumption, as shown in Fig. 10. Nevertheless, by evenly spreading workload among sensors, the network manages to live longer. We measure network lifetime using four different metrics broadly present in literature. We count the number of messages that are sent between random source–destination pairs until an undesired event occurs: the death of the first node, the arise of unsensed areas (i.e., an area outside the transmission/sensing range of any node in the network), the disconnection of the network and finally the reaching of a fixed percentage of undelivered messages. Results shown in Fig. 11 are dramatic: when the first node dies we manage to send over 98% more messages than those sent by carrying out a simple compass routing procedure, almost halving the way to the result achievable with the ideal topology of a grid. Similarly, BCR halves the way to the grid case for the other three metrics (delivery rates are depicted in Fig. 10(d)).

We also report that at the time of the first death of a node no message was lost, while at network disconnection and at the first occurrence of a coverage hole only less than 1% of the total sent messages did not reach the destination.

7.3. *When squares are squares…*

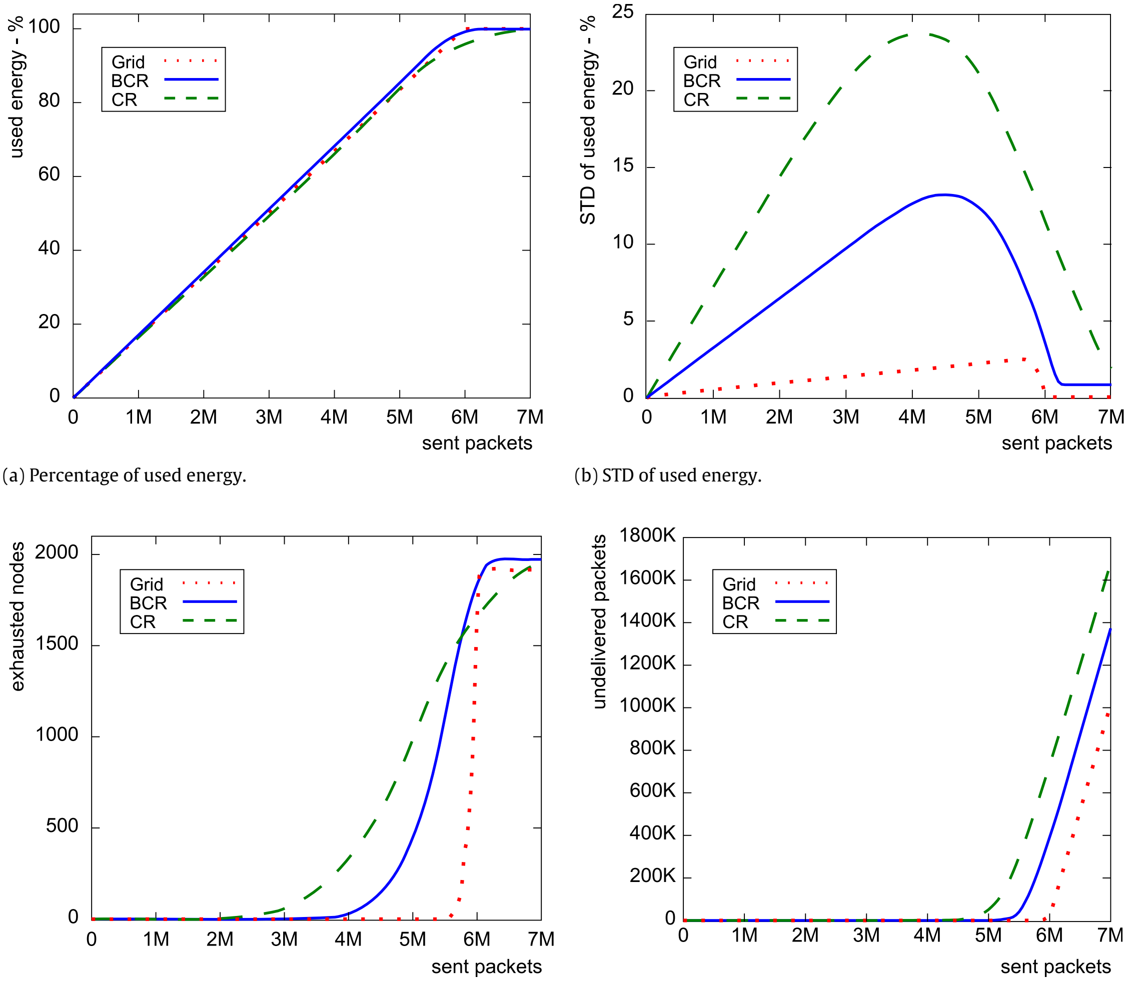
So far we have tested our competence balancing procedure over a torus surface, in order to exclusively address unbalance springing at the microscopic level and we saw that nodes die out at a much slower rate, almost doubling network longevity. We now show how BCR behaves when it is applied to a simple square network—opposite sides of the square are *not* glued together to construct a torus. The best way to use BCR, however, is to couple it with a load balancing procedure acting at the macroscopic level. BCR works in fact on smoothing energy use among neighboring nodes, prolonging their concomitant use as long as possible. This is achieved by virtually re-creating local topological regularity. If a balancing procedure for the macroscopic level is adopted, BCR is able to further prolong its action, as major disconnection and coverage loss due to the global geometry of the network is thus postponed. For time being, we test BCR alone on a square network and leave for future work the coupling with coarse-grained procedures.

The result of our simulation is depicted in Fig. 12. All parameters have been set as earlier described, at the beginning of Section 7. Experiments in [16], establishes that – with geographic routing – network disconnection and coverage loss appear after approximatively 3 million messages have been send. At this time the network can be considered exhausted, as performance starts to become very poor and the decay process rapidly evolves. By running BCR over the same network, we observe that node's death can be slowed down, before 3 million packets are sent (Fig. 12(b)). This confirms the above stated intuitions—if a macroscopic balancing procedure is also adopted, major disconnection can be postponed, while BCR maintains the network in a healthier state up to that moment.

The most remarkable result comes when considering delivery ratio as an efficiency estimator (Fig. 12(c)). Although unbalance due to global geometry parameters remains, the pre-processed network manages to reduce, by more than half, the number of messages not reaching the destination. By assuring fair work-load distribution among neighboring nodes, we believe that the BCR procedure prevents the formation of dead ends, where paths are interrupted before reaching the destination. In Fig. 12(a), we also show that the observed achievements are obtained at a low energy cost increment.

7.4. *Security related data results*

As discussed earlier in this paper, our pro-active routing protocol protects the network against the sinkhole attack.



(a) Percentage of used energy.

(b) STD of used energy.

(c) Number of exhausted nodes.

(d) Number of undelivered messages.

Fig. 10. *Monitoring of network behavior*. We sent 7 million packets and observed the response of the network. Our balanced routing procedure requires a negligible amount of extra energy respect to the greedy routing approach (a) and the use of energy is far more efficient as it is better spread among nodes (b). In fact, nodes die off at much smaller rate (c) and the number of undelivered messages with BCR stays always below the corresponding value for CR.

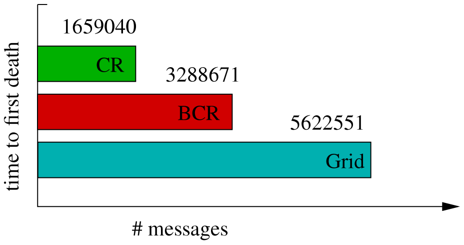


Fig. 11. *Network lifetime*. Number of messages sent at the death of the first node.

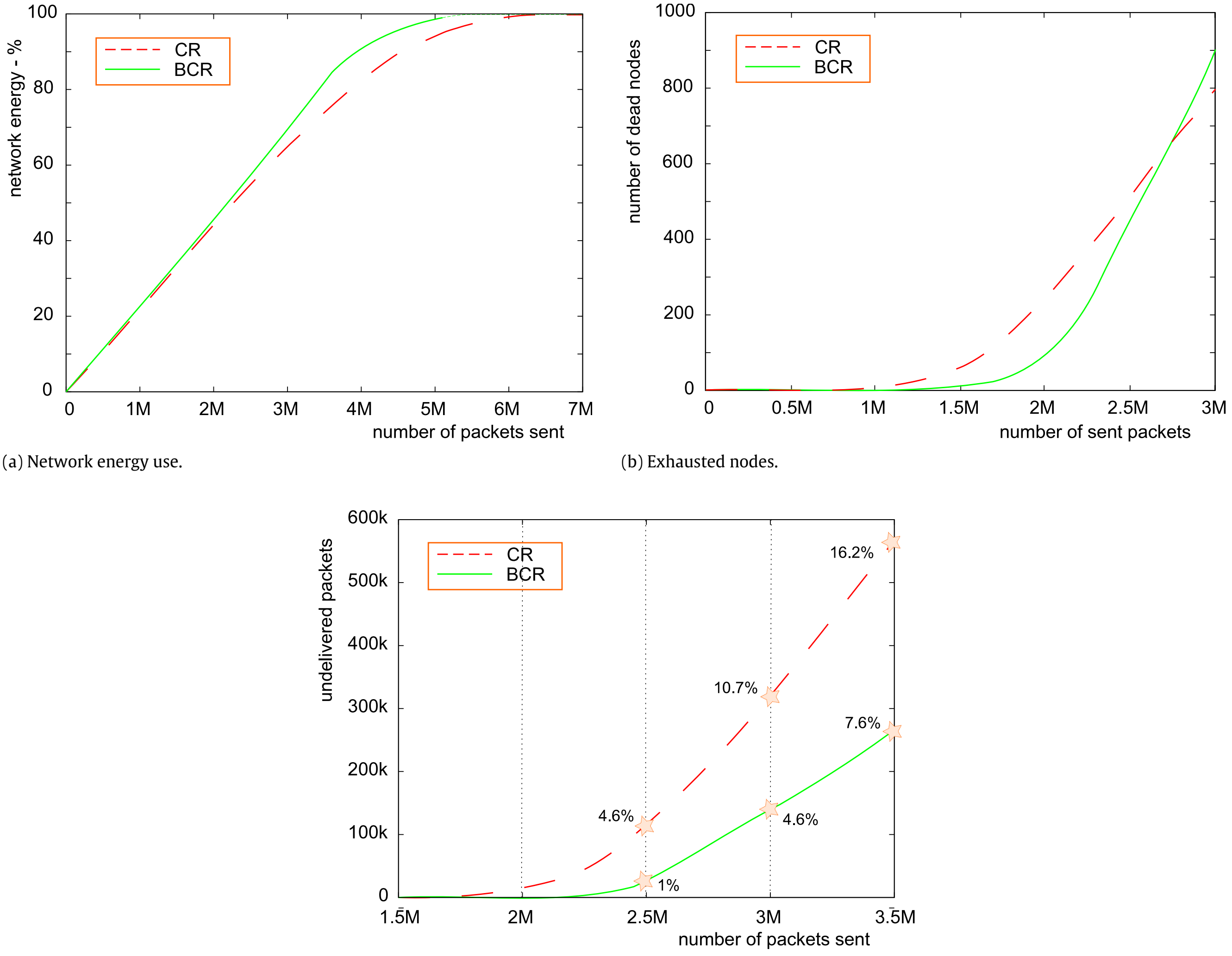
However, homogeneous distribution of work load also makes it harder for an adversary to perform attacks with the goal of taking control over the largest possible number of messages. In fact, an attacker interested in controlling as many messages as possible would certainly start with compromising the most loaded nodes, as those are the ones that witness the major volume of traffic. We assume that a strong adversary is capable of identifying these nodes and tamper with them. To measure the additional protection against this attack obtainable with our protocol we made the following experiment. We considered the set $\mathcal{P}$ of packages that traverse the network. We then select the node u that handles the maximum number of packets and consider the set $\mathcal{P}'$ obtained from $\mathcal{P}$ by deleting packets going through u. At the second step we select a second node u', distinguished from u, choosing the node that handles the maximum number of packages contained in $\mathcal{P}'$ and consider the set $\mathcal{P}''$ obtained from $\mathcal{P}'$ by deleting packets going through the last selected node. We follow this procedure until the number of packets taken out from $\mathcal{P}$ is the 25% or 50% of the total number of packets traversing the network. It thus results that using our protocol, an attacker should take control of approximatively 20% nodes more than the number of nodes that the same attacker should capture if the sole greedy protocol is to be adopted.

8. Conclusion

In this paper we have introduced the first pro-active routing mechanism to balance the relay traffic in multi-hop wireless networks at a local level of sight. Our experiments show improvements on network lifetime up to 98%. As a positive feature of our pro-active strategy, we also deliver a network that is more robust against the presence of nodes that try to divert routing in such a way to attract a large part of the traffic in the network.

Acknowledgment

Alessandro Mei is supported by a Marie Curie Outgoing International Fellowship funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 253461.



(a) Network energy use.

(b) Exhausted nodes.

(c) Undelivered messages.

Fig. 12. *Performance on a square network.* We have sent 7 million packets over a network deployed on a square surface and have compared the performance of our balancing procedure (BCR) with the greedy-compass protocol (CR). (a) shows that the amount of additional energy required by BCR is almost negligible. (b) depicts the distribution of the number of exhausted nodes. We observe that, up to 3 million of sent packets, BCR outperforms CR; afterwards the tendency is inverted. The highlighted turning point corresponds to the approximate time of first appearance of a disconnection and loss of coverage in the network. A network is generally considered inefficient at this point. In (c) network efficiency is measured with the percentage of undelivered messages out of the total number of sent messages. The percentage of undelivered messages with BCR is always well below the corresponding value for CR.

References

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

[2] Giuseppe Anastasi, Marco Conti, Mario Di Francesco, Andrea Passarella, Energy conservation in wireless sensor networks: a survey, Ad Hoc Networks 7 (3) (2009) 537–568.

[3] P. Casari, A. Marcucci, M. Nati, C. Petrioli, M. Zorzi, A detailed simulation study of geographic random forwarding (GeRaF) in wireless sensor networks, 2005. MILCOM 2005, in: Military Communications Conference, vol. 1, IEEE, 2005, pp. 59–68. pp. 17–20.

[4] P. Casari, M. Nati, C. Petrioli, M. Zorzi, ALBA: an adaptive load-balanced algorithm for geographic forwarding in wireless sensor networks, in: Military Communications Conference, 2006. MILCOM 2006, IEEE, 2006, pp. 1–9.

[5] Yashar Ganjali, Abtin Keshavarzian, Load balancing in ad hoc networks: single-path routing vs. multi-path routing, in: INFOCOM 2004. 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 2, IEEE Computer Society, 2004, pp. 1120–1125.

[6] M. Gatzianas, L. Georgiadis, A distributed algorithm for maximum lifetime routing in sensor networks with mobile sink, IEEE Transactions on Wireless Communications 7 (3) (2008) 984–994.

[7] Piyush Gupta, P.R. Kumar, The capacity of wireless networks, IEEE Transactions on Information Theory IT-46 (2) (2000) 388–404.

[8] Xiaoyan Hong, Pu Wang, Jiejun Kong, Qunwei Zheng, Jun Liu, Effective probabilistic approach protecting sensor traffic, in: MILCOM—Military Communications Conference, 2005, pp. 1–7.

[9] Yih-Chun Hu, Adrian Perrig, A survey of secure wireless ad hoc routing, IEEE Security & Privacy 2 (3) (2004) 28–39. Special issue on Making Wireless Work.

[10] Esa Hyytiä, Jorma Virtamo, On traffic load distribution and load balancing in dense wireless multihop networks, EURASIP Journal on Wireless Communications and Networking (2007). Special issue on Novel Techniques for Analysis & Design of Cross-Layer Optimized Wireless Sensor Networks.

[11] Chris Karlof, David Wagner, Secure routing in wireless sensor networks: attacks and countermeasures, in: Elsevier's Ad Hoc Networks Journal, Special Issue on Sensor Network Applications and Protocols, 2003.

[12] Evangelos Kranakis, Harvinder Singh, Jorge Urrutia, Compass routing on geometric networks, in: Proc. 11th Canadian Conference on Computational Geometry, 1999, pp. 51–54.

[13] Sungoh Kwon, Ness B. Shroff, Paradox of shortest path routing for large multi-hop wireless networks, in: INFOCOM, IEEE Computer Society, 2007, pp. 1001–1009.

[14] Fan Li, Yu Wang, Circular sailing routing for wireless networks, in: INFOCOM, IEEE Computer Society, 2008, pp. 1346–1354.

[15] Jun Luo, Jacques Panchard, Michal Piorkowski, Matthias Grossglauser, Jean-Pierre Hubaux, Mobiroute: routing towards a mobile sink for improving lifetime in sensor networks, in: International Conference on Distributed Computing in Sensor Systems, 2006.

[16] Alessandro Mei, Julinda Stefa, Routing in outer space: fair traffic load in multi-hop wireless networks, in: MobiHoc'08: Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing, ACM, 2008, pp. 23–32.

[17] Bryan Parno, Adrian Perrig, Virgil Gligor, Distributed detection of node replication attacks in sensor networks, in: IEEE Symposium on Security and Privacy, 2005.

[18] Adrian Perrig, John Stankovic, David Wagner, Security in wireless sensor networks, Communications of the ACM 47 (6) (2004) 53–57.

[19] Peter Pham, Sylvie Perreau, Performance analysis of reactive shortest single-path and multi-path routing mechanism with load balance, in: INFOCOM, IEEE Computer Society, 2003.

[20] Lucian Popa, Afshin Rostamizadeh, Richard M. Karp, Christos H. Papadimitriou, Ion Stoica, Balancing traffic load in wireless networks with curveball routing, in: Proceedings of the 8th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2007, ACM, 2007, pp. 170–179.

[21] Ivan Stojmenovic, Xu Lin, Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks, IEEE Transactions on Parallel and Distributed Systems 12 (2001) 1023–1032.

[22] Arvinderpal S. Wander, Nils Gura, Hans Eberle, Vipul Gupta, Sheueling Chang Shantz, Energy analysis of public-key cryptography for wireless sensor networks, in: PerCom, IEEE International Conference on Pervasive Computing and Communications, IEEE Computer Society, 2005, pp. 324–328.

[23] Ya Xu, John Heidemann, Deborah Estrin, Geography-informed energy conservation for ad hoc routing, in: ACM MOBICOM, 2001, pp. 70–84.

[24] Yunjung Yi, Taek Jin Kwon, Mario Gerla, A load aware routing (LWR) based on local information, in: PIMRC 2001, 12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, vol. 2, IEEE Computer Society, 2001, pp. G-65–G-69.

[25] Y. Yu, R. Govindan, D. Estrin, Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks, Technical Report UCLA/CSD-TR-01-0023, UCLA Computer Science Department, 2001.

[26] Alexander Zemlianov, Gustavo de Veciana, Capacity of ad hoc networks with infrastructure support, IEEE Journal on Selected Areas in Communications 23 (3) (2005) 657–667.

[27] Sencun Zhu, Sanjeev Setia, Sushil Jajodia, LEAP: efficient security mechanisms for large-scale distributed sensor networks, in: Proceedings of the 10th ACM Conference on Computer and Communications Security, ACM Press, 2003, pp. 62–72.

[28] M. Zorzi, R.R. Rao, Geographic random forwarding (GeRaF) for ad hoc and sensor networks: energy and latency performance, IEEE Transactions on Mobile Computing 2 (4) (2003) 349–365.

Alessandro Mei received the laurea degree in Computer Science summa cum laude from the University of Pisa, Italy, in 1994. He continued his studies as a Ph.D. student at the Department of Mathematics of the University of Trento, Italy, and as a visiting scholar at the Department of EE-Systems of the University of Southern California during 1998 and part of 1999. He received the Ph.D. in Mathematics at the University of Trento in 1999. After a postdoctoral position at the University of Trento, in 2001 he joined the faculty of the Computer Science Department at Sapienza University of Rome, Italy. His main research interests include computer system security and parallel, distributed, and networked systems. He was presented with the Best Paper Award of the 16th IEEE International Parallel and Distributed Processing Symposium in 2002, the EE-Systems Outstanding Research Paper Award of the University of Southern California for 2000, and the Outstanding Paper Award of the Fifth IEEE/ACM International Conference on High Performance Computing in 1998. He is a member of the ACM and the IEEE, a past associate editor of the IEEE Transactions on Computers (2005–2009), and the general chair of IEEE IPDPS 2009, Rome, Italy. Alessandro Mei is a Marie Curie Fellow (at the CSE Department, University of California San Diego, from Aug. 2010 to Aug. 2011, and at the CS Department, Sapienza University of Rome, from Aug. 2011 to Aug. 2012).

Natascia Piroso graduated in Mathematics in 2005 at "Roma Tre" University in Rome, Italy. In 2007, she received her Mathematics Master's degree with highest honors from the same university. She is currently a Ph.D. student in Computer Science at Sapienza University in Rome under the supervision of Prof. Alessandro Mei. Her research interests include algorithms for parallel and distributed systems, network security, fault-tolerant computing, resource management, optimization, computer networks and communication, wireless networks, analysis of social mobile networks and dedicated applications. In spring 2011 she was a graduate visiting scholar at the CS&E department, University of California San Diego.

Bruno Vavala received the B.Sc. and the M.Sc. degree with highest honors in Computer Science from Sapienza University of Rome, Italy, respectively in 2007 and in 2009. At the completion of his studies, the same institution awarded him the Excellence Course Fellowship. In 2009 he joined University of Lisbon, Portugal, where he is currently conducting research, working on national and European projects as a Ph.D. student member of the Large-Scale Informatics Systems Laboratory (LaSIGE). In 2011 he will also join Carnegie Mellon University. His main research interests include distributed computing, fault and intrusion tolerance, security, wireless ad-hoc networks.