An analysis of cooperation in ad hoc networks focusing on energy consumption

Alessandro Urpi

September 2005
Chapter 1

Introduction

Pure altruism is a behavior that does not exist in nature: It is simply not convenient from an evolutionary point of view.

If a population is composed only by altruist and exploiter elements, it is more advantageous to adopt the latter behavior as long as someone keeps bearing the costs and working for the others. On the other hand, a society composed only by exploiters is not possible, since nobody would produce anything.

However, nature is full of examples of cooperation among animals of the same species, or even among animals and plants. For example:

- Birds can not, in general, groom parasites off their own heads. For this reason, the task is usually accomplished by other birds, that expect to receive the same treatment when they finish.

- Suricatas (small savannah mammals) coordinate themselves to guard the area where they live, each looking at a different direction. With this technique, they are able, as a team, to control an area much larger, and from any angle.

- Insect help plants in pollinating, receiving food as incentive.

Note that in all the presented cases, cooperation is mutual and brings to individuals more advantages than costs: In a few words, it increases the survival probability of anyone which adopts it. As pointed out in [30], even postulating that selfishness is the basic behavior of any organism, there may exist particular configurations in which cooperation may naturally arise as the best way to operate.

In this thesis we analyze wireless ad hoc networks, a class of systems naturally based on cooperation, in order to understand under which conditions cooperation is the best strategy for all the involved nodes, and when it needs to be enforced in some way.

In Section 1.1 we introduce ad hoc networks, focusing on the particular networks analyzed throughout the thesis. In Section 1.2 we outline the objectives and contributions of the thesis, presenting its organization in Section 1.3.
1.1 Ad hoc networks and cooperation

This section is devoted to a brief description of ad hoc networks, the system analyzed throughout all the thesis. It is difficult to characterize such a dynamic and evolving world, and for this reason we sketch only its basic properties, stating which sub-class of system can be damaged by the lack of cooperation, and are thus the object of this thesis.

For a deeper introduction to ad hoc networking, many books have been written, like [32], [19] and [7].

1.1.1 Networking in absence of infrastructure

Wireless mobile ad hoc networks (called simply ad hoc networks in the remainder of the thesis) have been proposed in a military environment, with the name of DARPA packet radio networks ([42]), as soon as radio waves were successfully used for packet switched communication.

The target was to build a communication network in problematic scenarios, like battlefields, where communication infrastructures do not exist, or are not reliable, given the presence of an enemy controlling them.

The designers aim was to build networks easy and quick to be deployed, tolerant to nodes mobility and failure, and automatically managed while in operation. These requirements can be met only if nodes composing the network cooperate to keep it connected: if a message can not be delivered directly to destination, it must be forwarded by intermediate nodes.

As reported in [42], during the first years of the eighties networks with up to 138 nodes were effectively tested, with transmission, queueing and routing protocols successfully implemented. This success led to a boost in research in packet radio networks, also looking for commercial applications.

The first natural extension of their usage was in disaster recovery and civil protection scenarios: when communication infrastructure is damaged, it is very important to be able to set up a temporary network to coordinate all the persons working in the area.

In the last years, due to the advances in wireless technologies, and the diffusion of low cost wireless network adapters, it is possible to use ad hoc networks in many additional scenarios. Typical examples range from temporary networks that allow a group of persons to exchange files and documents while meeting, to connectivity of rooms and entire buildings (e.g., waiting rooms in airport or stations, schools, etc), to vehicular networks.

During the evolution of ad hoc networks, many protocols have been proposed in order to optimize different resources. As an example, originally implemented routing was performed by means of tables periodically broadcast by every node, obtaining a global knowledge of the network at every node ([42]). This method is very expensive when not all the possible routes are needed, and when a route setup
can be delayed a few fractions of seconds: For this reason the current trend is to use on-demand routing algorithms, that build routes only when needed, saving in general bandwidth and energy.

The adaptation of ad hoc networks to new scenarios, with the specialization of protocols, led to consider the “ad hoc” as a framework, a sort of meta-system which is only an abstract generalization of several systems sharing a basic set of features, with different needs and issues.

The core vision is that ad hoc networks are composed by nodes potentially mobile, that use a shared medium to communicate, generally by means of omnidirectional wireless interfaces, without the aid of any infrastructure.

As an example, all the following systems fit into the ad hoc definition:

- Vehicular networks [38], whose nodes are cars, mobility is high but predictable.
- Personal area networks [60], whose nodes are small devices (e.g., mobile phones, PDAs, etc) surrounding a person, with little relative mobility.
- A few notebooks in a room that connect to exchange files during a meeting or a conference, with no mobility.
- Sensor networks [3], whose nodes are small devices with strong limitations in power, computational power and storage size, with generally no mobility but high failure rate.

However, sensor and vehicular networks share nothing more than the usage of a shared medium for communications: The former is severely limited in energy and almost static, while the latter has almost no energetic problem and it is very mobile. For this reason, given their peculiarities that led to the introduction of dedicated protocols, it is common to consider such networks as disjoint from ad hoc ones.

On the other hand, networks not fulfilling the presented definition can be considered almost ad hoc, and can adopt protocols specifically designed for the ad hoc world. For example, hybrid cellular networks ([2]) can be treated as an interconnection of ad hoc networks by means of a dedicated infrastructure.

### 1.1.2 Who makes ad hoc network work?

As previously noted, the absence of any infrastructure can be overcome if the nodes cooperate to keep network connected and deliver packets. If a node can not directly deliver packets to a destination, it needs one or more relying nodes (chosen with a routing protocol) to accomplish the task.

Packet delivery can be considered itself as a distributed protocol, which uses routing to select, at each node, the following step: A generic node, at the reception of any packet not directed to it, selects a neighbor node to which to send a copy of the packet.
As in classical distributed computing, cooperation of nodes has not been questioned for a long time: it was assumed as a fact, and the research effort has been directed mainly in finding optimal forwarding and routing protocols.

However, it is possible to observe a discontinuity point in the evolution of ad hoc network presented in Section 1.1.1: From closed systems managed and controlled by a unique authority (like in the military or civil protection scenarios), networks became open systems, with nodes possibly belonging to different domains, and without any centralized control. This mutation was to be expected, in systems lacking any type of infrastructure, opening at the same time interesting opportunities and unconsidered security issues.

For example, well established security techniques like asymmetric cryptography can not be used if it is not possible to trust any node in the network. In fact, during the last years, many research efforts have been directed at secure ad hoc networks, and the work to be done is still much.

Another problem that was pointed out is that of possible lack of cooperation. In fact, noting that helping to operate the network has a non negligible cost in energy and/or in bandwidth, if nodes in an ad hoc network do not have a common goal, and are not controlled by any central authority, why should they waste resources helping others?

Any node could benefit by adopting a non cooperative behavior, namely ignoring the forwarding and routing requests. Clearly, such a behavior causes network dysfunction, which is more critical as the number of misbehaving nodes increases. Note that uncooperative nodes are not malicious, as they do not want to disrupt the network, nor to damage any other node. It is just a self-interested behavior that causes problems as a side effect.

Nodes that do not forward for others are often called selfish nodes, following the definition of selfish gene given in [30] by Dawkins. Throughout this thesis, they will be also called parasite nodes.

The problem of cooperation stimulation in ad hoc network is to study the impact of not cooperative nodes on the network performances, and to eliminate their presence.

1.2 Thesis objectives

This thesis analyzes both sides of the cooperation stimulation problem.

First of all, a realistic estimation of the impact of selfish nodes in ad hoc networks is given. In order to motivate the need for cooperation stimulation mechanisms, many simulative studies have been published during last years ([46], [47]). However, at our best knowledge it has never been noted that selfish nodes must be rational entities that want to maximize their resources while using the network. For this reason, scenarios with many selfish nodes and a non working network are very unlikely to be observed in the real world. Even in absence of any cooperation
stimulation/enforcement mechanism, there must be a selfishness threshold that is not overcome, in order to have a network functioning on average.

It is then worthy to ask whether a cooperation enforcement mechanism is always needed, or if the network will work as the result of an equilibrium between cooperative and selfish nodes, just like many societies do.

On the other hand, even if a small amount of selfish nodes do not have a great impact on the global network functioning, they can be a serious trouble for some of the network nodes, which would be forced to an extremely high energy consumption, or to complete isolation. For this reason, the design of a good cooperation stimulation mechanism is surely important in order to have robust and reliable open ad hoc networks.

As pointed out in Chapter 2, cooperation is usually enforced by means of a punishment system or an incentive scheme.

In the former case, all the nodes in a distributed way monitor their neighbors, identifying selfish nodes with a precision high enough (perfect detection is not possible, for many practical reasons), and punishing them with the (often temporary) exclusion from the network (i.e., their traffic is not forwarded). In the latter cases, some virtual or real currency is used as payment for any message successfully delivered to the destination, making forwarding advantageous.

It is not clear which solution is more effective, given their differences that makes them almost incomparable. We present two formal models, both based on game theory, able to describe different features of any punishment system. The models allow to prove that punishments inflicted to presumed selfish nodes must be carefully tuned in order to avoid an unfair or useless solution. However, this task is extremely difficult when selfish nodes can not be detected with precision.

This analysis suggests that incentive mechanisms are to be preferred, since they do not force any node to do anything, but obtain cooperation by effectively making such a behavior more convenient.

Unfortunately, the bigger problem of incentive systems is that they need to be implemented by means of special hardware that forbids currency creation and steal. Thus, if in a hybrid cellular network [2] this can be done using the existing communication and pricing infrastructure, in general it would imply a re-design of ad hoc networks.

We then conclude the thesis by suggesting a novel approach that can be used as incentive scheme in ad hoc networks where energy is limited (it can not be used, for example, in vehicular networks): Cooperation is rewarded with sleep time, that translates into energy saving. Nodes that do not cooperate are still able to use the network, but in many cases spending more energy than cooperative nodes: Selfishness would be then an inconvenient behavior also from a personal point of view. Due to the complexity of such a mechanism, we do not offer any final solution in the thesis, but we propose an energy saving mechanism that, if combined with a good detection mechanism, could probably be successful and practical to be implemented.
1.3 Organization of the thesis

The thesis is organized as follows.

Chapter 2 presents a survey of the results regarding cooperation stimulation.

In Chapter 3 it is presented the notation used throughout the thesis, and a survey of the main results about topological properties of ad hoc networks. This is useful for the thesis main subject since all the presented models are very general, and use concepts like hop count distribution and communications pattern that changes depending on the network type.

A theoretical evaluation of performances degradation caused by selfish nodes is presented in Chapter 4. The analysis shows how the impact of selfish nodes is related to nodes density and network topology, and partially mitigates the disruptive view presented in many papers related to cooperation enforcement.

Chapter 5 presents two models of punishment based cooperation enforcement mechanisms, that allow to understand general features common to all the mechanisms and in principle to compare them.

In Chapter 6, a distributed energy saving mechanism that can be used in encouraging cooperation, if a good reputation system is built to detect selfish nodes, is described.

Finally, conclusions and future works terminate the thesis.
Chapter 2

State of the art in cooperation enforcement/stimulation

In this chapter, we give a critical survey of the known results on cooperation enforcement, dividing all the proposed solutions in two main classes: punishment systems and incentive mechanisms.

This review is meant to describe in details how cooperation enforcement research was started, and how the proposed solutions can be logically grouped in two classes that have evolved in a parallel way: After such analysis, it will be clear which problems are still open, and what is the contribution of this thesis.

Moreover, we outline a basic model that can be used to study proposed cooperation enforcement mechanisms, and that will be specialized in this thesis (particularly in Chapter 5). It may be useful, to better understand this Chapter, a very basic knowledge of game theory, quickly reviewed in Appendix.

2.1 Cooperation identified as a critical issue

The first description of the terrible effects of the lack of cooperation is, at our best knowledge, in [46]. Authors show, by means of simulations, that even not extremely high percentages of parasite nodes are capable to disrupt communications in an ad hoc network.

However, authors of [46] do not directly address the problem of guarantee that all nodes cooperate, but aim at increase the network goodput even in presence of uncooperative nodes: They propose two mechanisms to detect and avoid selfish nodes. In this way, cooperative nodes are not damaged by uncooperative ones, unless the fraction of selfish nodes is not extremely high.

The first proposed mechanism is called \textbf{watchdog}, and it is used to detect selfish nodes. In practice, all the nodes put their radio in promiscuous mode, i.e., they receive all the packets that arrive to their radio (even not destined to them), and analyze their content. In many cases it is possible to understand if the node
that transmitted the packet is the originator, or a forwarder doing its task. Thus, it is possible to introduce a reputation system that labels neighbors as good or bad.

This mechanism is not perfect: in many cases it is not possible to listen to neighbors’ communications (for interferences, simultaneous incoming packets, etc). Moreover, keeping the radio constantly in promiscuous mode is very expensive in energetic terms, and it should be avoided if possible.

The second mechanism is called pathrater, and it is used to give a reliability index to any route, using the reputation of the nodes composing the route itself. Thus, routing must be modified in order to ask the pathrater before to propose any route, and to discard routes containing misbehaving nodes (looking for alternatives, if existing).

Simulations show a significant increase in goodput, because many selfish nodes could be detected with success.

However, this was the starting point of an unfinished debate: Many researchers argued that letting selfish nodes use the network, while not using them as forwarders, is equivalent to reward selfishness.

Following [46], many solutions have tried to discourage selfishness by means of punishments for selfish nodes or rewards for cooperative ones.

2.2 Punishments and rewards: a formal explanation

Cooperation in ad hoc networks is difficult to model: it is not clear which are the costs and the benefits that it yields to nodes, and it is not well understood if there are cases in which it may spontaneously arise without any control. For this reason there is an increasing interest in using analytical tools borrowed from economics and social sciences, like game theory.

Nodes composing an ad hoc network can be seen as interacting entities that can request or offer a service: a single node can be, and usually is, both a user and a provider. For the sake of clearness we will just consider the packet forwarding functionality. It is possible, at least in principle, to extend all the considerations we will make to a large class of other services that need global cooperation, like routing, even if in practice it can be difficult to deal with all the services at the same level.

In a given moment, one or more nodes composing an ad hoc network ask to some other nodes to forward messages for them. A potential forwarder can accept the task or ignore it. In the latter case, ignored packets are lost, and the sources have to fix the problem in some way. Given the infrastructureless nature of an ad-hoc network, nothing can work if no providers accept relay requests. On the other side, each node is really concerned with energy consumption, and would prefer someone else to carry on the task, since there is no explicit payment for it.

This situation reminds the well known prisoner’s dilemma ([31]): Every node
2.2. PUNISHMENTS AND REWARDS: A FORMAL EXPLANATION

has two possible actions to choose from. Cooperation leads to system optimization, but it is individually not convenient, and any rational node would prefer to leave the task to someone else. On the other hand, if no nodes cooperate, the network guarantees only single hop communications, limiting also the nodes utility. However, every node would prefer this situation to the one in which they cooperate, but their neighbors do not do the same.

Formally, for a two nodes network the payoff matrix is the following, with moves labelled as Acc (accept to forward) and Rej (reject forward requests) and with $a > b > c > d$:

<table>
<thead>
<tr>
<th></th>
<th>Acc</th>
<th>Rej</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc</td>
<td>$b, b$</td>
<td>$d, a$</td>
</tr>
<tr>
<td>Rej</td>
<td>$a, d$</td>
<td>$c, c$</td>
</tr>
</tbody>
</table>

Packet forwarding in a network can be seen as an infinite repetition of such a game, with forwarding requests made at every round, and decisions taken at the beginning in the form of a strategy (e.g. always accept, always reject, accept to forward just in even rounds, do to others what they are doing to you, do to others what you expect they are going to give you, and so on).

The bad news is that, in the short term, non cooperation is the only equilibrium of the game. However, with repetitions it is possible to force different equilibria by means of two tools:

- Punishments: if all the players observe the behavior of other players, under certain assumptions it is possible to enforce the strategy $s$ if all play the strategy $s'$, which basically can be described as “play $s$ until all the players do the same, and if someone deviates then punish her by playing the worst move with respect to the deviant player(s)”.

- Rewards: if nodes are payed to play a certain move, then the payoff will change and a sequence of moves can become more convenient than the sequence of the equilibria of the single round game. If the resulting equilibrium is socially more profitable than any sequence of single round game equilibria, then the payment can derive from this increased social welfare.

Apparently, the strategy beyond the mechanism presented in [46] is “play cooperate, and if some node does not cooperate, do not ask it to forward any other traffic”. Obviously enough, this strategy is not an equilibrium, because the strategy “never cooperate” will bring a higher payoff, since after a turn no nodes will ask to forward any traffic.

For this reason many alternative mechanisms have been presented, trying to enforce a cooperative equilibrium. We will analyze the two philosophies in the next Sections.
2.3 Rewarding the cooperation

Paying nodes to cooperate is an economic approach to resolve the problem of cooperation. The solution is theoretically superior to the counterpart of punishing deviating nodes, because nodes are free to cooperate or not: but if they really look at their own convenience, they will end cooperating at least at a certain degree, in order to store enough money (which can be virtual) to be able to use the network (i.e. to pay other nodes in order to forward their own traffic). However, the solution is at the moment of difficult application: in order to manage money (real or virtual), special hardware must be present in the nodes. Otherwise, stealing or forging can not be avoided. We present the principal solutions using incentives, and in Chapter 6 we will present an alternative solution, in which cooperating nodes are rewarded with sleeping time.

2.3.1 Nuglets: a market model

In [17, 18], Buttyan and Hubaux model networks as markets, where services are exchanged. A virtual economy, based on a virtual currency called nuglet (or bean), is then introduced, forcing nodes to pay to have their packets forwarded, and to being paid when they forward some data. In this way, a selfish node would soon finish its nuglets, and would be forced to cooperate in order to send other packets.

This is one of the very first solutions proposed in literature and it is not based on monitoring, which is the main weak point of all the punishment based mechanisms. Every node has an initial account of $C$ nuglets and a fixed battery capability. Packets are generated at a constant average rate $f_o$, and every node receives packets to forward at rate $f_r$. It is possible then to compute the number of packets to send and forward in order to maximize the throughput while not running out of battery. The point is clearly a fair optimum, since all the nodes are forced to spend almost the same amount of energy to forward and to send their data.

It is clear that nuglets should be managed in a tamper-proof part of the node, ensuring that a malicious user does not change its device in order to steal, forge, or throw away nuglets. Some of these problems have been solved in [18] with the introduction of counters in place of nuglets, always managed in a separate hardware module.

The proposed solution has been enriched by Zhong et al. in [71], where authors propose a solution where no tamper proof hardware is needed, but a centralized server acting as a bank must exist, if not within the ad hoc network, at least in another network reachable by all the nodes.

Four different dynamic strategies for managing nuglets are analyzed, showing that the most generous one (forward all the packets until you reach your limit) is also the best performing one.
2.3.2 A general model of cooperation pricing

In [28] it is presented a general model that considers incentives for nodes in a wireless ad hoc network based mainly on the congestion of the network. Authors model nodes of a network as having a willingness to pay to send their traffic (each node may have a different level), and provide a distributed mechanism to fix prices that, depending from the traffic flowing through nodes (i.e. from the congestion at that node) will be an incentive to communicate. Prices depend on actual network characteristics, e.g. existence of different paths among sources and destinations, energy level of forwarding nodes, etc.

Simulations of the model show that prices quickly stabilize for a static network, with nodes located at the center having the highest prices since they in general serve more flows. Mobility helps by increasing the throughput of the network and by balancing the benefit among nodes: on average, all the nodes will experience the same congestion/revenue during their lifetime.

2.3.3 Auctions and forwarding

The solution presented in [28] is very similar to iPass ([22]), a mechanism to reserve bandwidth and stimulate cooperations based on auctions. Basically, when trying to set up a new flow, every source must acquire enough bandwidth along all the path by participating to an auction at every forwarding node. Bandwidth is assigned following a generalized Vickrey auction scheme ([64]), in order to ensure that every user bids following its real evaluation of service. A reserve price is used, in order to ensure a minimum payoff to all the forwarding nodes, and simulations confirm that the mechanism is able to optimally allocate resources, stimulating cooperation. On the other hand, iPass is a heavy mechanism, specially considering that many ad hoc networks are composed by small devices, and it relies on a security infrastructure that still need to be provided.

2.4 Punishing the selfish

Punishing the nodes that are not cooperating is more practical than rewarding the well behaving nodes: In fact, the main issue is to design a good reputation system, because punishing a node which is in reality not selfish is unacceptable.

We present the most distinguished solutions in this field.

2.4.1 CONFIDANT: An evolutionary model

Buchegger and Le Boudec in [15, 16] start from the possibility of adopting an evolutionary approach: they see nodes as an interacting population, and look for a strategy that yields more benefit than any other strategy that a newcomer node can adopt ([30]). If all the nodes in a network adopt such a mechanism (like if it
was coded in their genes), then a node using a different strategy (for a mutation, of because it arrived from another population where evolution led to other solutions) would not adapt, i.e., it would receive less service than other nodes, and should change its strategy (adopting the official one), or die, being excluded from network use. In [4] it is shown that a strategy can be evolutionary winning only if, among its features, is adaptive. For this reason, nodes have to be equipped with a watchdog unit, to observe the behavior of neighbors, and adapt to their behavior. Moreover, they are also organized in a friendship network to speed up the propagation of alerts. When a node discovers that one of its neighbors is not cooperating, it starts warning all its friends, that mark the misbehaving node as bad. At this point, the selfish node is cut off from network services, because bad nodes are not served. Warning messages are surely the main weak point of the first proposed solution ([15]): they add overhead to the network, and it is possible to spread fake information, having an open door to denial of service attacks. For this reason in [16] authors extended the protocol in order to trust to messages about unknown nodes only when they arrive close in time and from a large number of friends.

2.4.2 CORE: A reputation based model

The solution proposed in [48] and [49] by Michiardi and Molva overcomes the problem of alert messages, and makes explicitly possible to end a punishment when a node starts behaving well. For this reason, every node has a local knowledge of reputation of other nodes, which can modify just in two cases:

- With local observations (again, with a watchdog unit), it can increase or decrease other nodes’ reputation, depending on how are they behaving: in this way, a previously non collaborative node can start helping and be re-integrated in the network, even if very slowly.

- With indirect deductions, that can be just positive: for example, if the identity of every node along a route is known to the sender and the communications are explicitly acknowledged by destinations, it is possible to raise the reputation of all the forwarders after every received ack.

Authors analyze their solution in game theoretic terms, proving that if only half of the nodes adopt it, the remaining nodes have to collaborate in order to use the network. However, authors use ERC theory ([13]) to model a multi-player prisoner’s dilemma with fair share of resources. ERC perfectly explains weird behaviors exhibited by humans, when playing games during research experiments. The observed tendency to depart from Nash equilibria is explained with a major satisfaction deriving not only from monetary payoff, but also from the distance of others’ payoff. Ad-hoc nodes probably are selfish in an absolute way, and it is difficult to compare them with humans, so a classical game theoretic analysis would have been more appropriate.
2.4.3 Using cryptographic certificates

In [68] it is proposed a mechanism to secure at the same time routing and forwarding. Every node must hold a token in order to use the network, and the token can be assigned by any $k - 1$ other nodes with a polynomial secret sharing technique ([58]). Tokens have a limited lifetime and need to be periodically renewed.

Every node must also monitor its neighbors, in order to discover selfish and malicious nodes, and alert all the network in case of a misbehaving node is detected. After an alert, the token is revoked and not renewed for a long time to the guilty node.

This solution is very similar to the CONFIDANT approach ([15]).

2.4.4 GTFT: an energy aware model

Srinivasan et al. in [61], propose a trade-off between the existent solutions. Since energy is the main concern of authors, nodes are seen as partitioned in energy classes (depending on their energy constraints), and communications are arranged in sessions, each with an associated energy class, determined by the “weaker” node in the chain of forwarders. Each node, before to start a session, asks to all the nodes in the route that will be used if they are going to support the session itself. It is possible to compute optimal strategies for every class having complete off-line network information, and authors also propose an algorithm that allow nodes to reach optimal allocations (by knowing them in advance) by simply recording how they are treated by other nodes, and keeping things in balance, substituting nuglets with explicit requests to participate in sessions. The solution does not enforce cooperation in the sense other works did, since a node is never really cut off from network services, and can continue to communicate also without forwarding. However, it is formally proven that if nodes are rational, and eager to receive the best possible service, they will always cooperate at a rate that it is Pareto efficient, i.e., not improvable by any other allocation unless one or more nodes receive less utility.

As already pointed out, every node must know in advance the energy classes of all the other nodes, in order to compute the best strategy to use, and all the nodes must declare their real energy class: Unless these requirements are eliminated in some way, the solution is of difficult usage in practice.

2.4.5 A general model

In [63] (extended in Chapter 5 and used as basis to model the behavior of CORE in [49]) it is not presented any cooperation enforcement mechanism, but it is given a general model in order to understand whether cooperation is really a problem in every case, and if yes, when it is possible to obtain it. Authors of this paper propose
an approach to cooperation based on Bayesian games, looking for the equilibria points of the system and for the class of enforceable cooperative behaviors.

Each node has initially no information about its neighbors, but can observe what they do during network operation. Prior to choosing which neighbor to serve and where to direct their own traffic, nodes can analyze the history they recorded, using a reactive strategy.

It is possible to show that the non cooperation is always an equilibrium (even if it does not necessarily imply that this will be the general behavior), and that in general there is a limit on the traffic a generic node is willing, at maximum, to forward, inducing then a bound on network performances. However, this limit depends on the characteristics of the nodes, and on the reached knowledge: the less a node is trusted, the more in general it should work to have its packets forwarded. Moreover, the amount of cooperation that can be forced depends on the mobility of nodes and on the network size: a punishment, in order to be effective, must persist for a long enough time (in order to make the punishment a real deterrent). If a selfish node has a very high mobility, it could damage its neighbors and then move before that the punishment becomes effective.

See Chapter 5 for more details.

### 2.4.6 Combining graph and game theory

Authors of [36] model an ad hoc network as a graph, and every node as a player of a game against the rest of the network (technically, as an automaton receiving traffic and feedback from the network, deciding how to play, and forwarding traffic following such a decision).

They introduce the dependencies graph, which is a closure of the connection graph, in which there is a directed edge between nodes $n$ and $m$ if $n$ is a forwarder for $m$ in some route.

Authors show that punitive strategy for a node is meaningful only when in the dependencies graph there exists a cycle: in this case, reducing its cooperation a node can damage also nodes that are not cooperating to its own traffic (even in an indirect way). Otherwise, the strategy of not cooperating is optimal.

Authors present a simulative study of the network conditions in which cooperation (with punishments) is the equilibrium, but they restrict the analysis to static networks with toroidal topology.

### 2.5 Other approaches

Selfishness is grouped with various malicious behaviors and with congestion in the interesting proposal presented in [26].

Every node is equipped with a reputation system that label the links to each neighbor, and not the nodes. Every time a packet sent through a neighbor arrives
to destination, the quality of the link directed to that neighbor is increased, while in case of not acknowledged reception, the quality is lowered.

A node chooses then paths with the best quality in their first hop (but if the system is used by all the nodes, this condition will be true for all the hops) in order to guarantee a good quality of communications, and it serves only neighbors with a good link quality, in order to try to dissuade selfishness.

By simulations, the authors show that the system is a good solution to avoid congestion, but there are no results regarding selfishness.

2.6 Conclusions

In this chapter we presented a survey of the literature regarding the cooperation enforcement/stimulation in ad hoc networks.

The main solutions proposed up to now were presented and theoretically justified.

In particular, we pointed out the differences between a punishment system, that tries to exclude selfish nodes from the network, and a rewarding system, which makes cooperation worthwhile.

There is not a solution that definitively solves the problem, and it is not even clear if selfishness needs to be treated as a very peculiar malicious behavior (thus trying to avoid it by means of a security system), or as a pathological feature of ad hoc networks that deserves special care.
CHAPTER 2. STATE OF THE ART IN COOPERATION ENFORCEMENT/STIMULATION
Chapter 3

Probabilistic topological properties

Wireless ad hoc networks are characterized by an intrinsic anarchy, which is one of their strengths and, at the same time, their main weakness.

In fact, ad hoc networks do not depend on a fixed, prone to failures infrastructure, and can be set up quickly and for free (e.g. spontaneous networking, [34]). However, on the other hand, it is difficult to offer guarantees on the quality of service, because performances heavily depend on randomly varying factors, like connectivity, network extension, mobility, etc.

For example, as pointed out in [69], the considered mobility model may drastically change the performance analysis of routing algorithms, and a wrong choice (as it turned to be random waypoint with null minimum velocity) could lead to inconsistent results.

Many existing works assume uniform spatial node distribution, and under such assumptions there are fundamental results relating network connectivity to node density. Other aspects (like average path length) are usually studied for very dense networks (see Section 3.2.4). However, uniform distribution is a good description of static networks (see Section 3.2), but it is not general.

In the analysis presented in Chapter 4, we will show how the impact of selfish nodes in a network depends on nodes density, average path length and number of different routes among nodes. In fact, the presence of parasites can be treated as a reduction in density (nodes do not participate to network protocols), and it is fundamental to understand how network properties change with density. This is specially true for densities approaching critically low values, for which connectivity has not a high probability.

It is then fundamental to define what is a good statistical description of an ad hoc network. Such description must provide essential information, like the average number of neighbors per node, the probability that a message needs more than $h$ hops to be delivered, etc.

In this thesis we describe an ad hoc network with the following building blocks:
Initial node distribution: If the network is set up in a single step, it is important to know how nodes are initially positioned. For example, it is generally assumed that nodes in a sensor network ([3]) are uniformly distributed, since sensors are scattered uniformly at random in the sensing field. If there is not an initial distribution, the arrival/departure process of nodes must be described.

Mobility model: How nodes move during their participation to the network. Examples are Random Waypoint Model ([41, 14]), City Section ([29]), etc (see [20] for a survey).

Communication pattern: How source-destination pairs are formed. Examples are uniform distribution (i.e. any node in the network can be destination for a given source), local communications (message destinations are limited to nodes located a few hops from the source), or communications mainly directed to one or more hotspots (e.g. when in the network is present a gateway to Internet).

Given these descriptions, if the positions of nodes at times $t$ and $t'$ are independently distributed, it is then possible to study the network by focusing on a single snapshot, simplifying the analysis of the desired properties.

This Chapter is mainly intended as a support for the analysis presented in Chapter 4. Its role in the thesis is to offer general definitions useful through all the Chapters (e.g., nodes density, average path length, etc) and to report the main results in statistical description of topological properties of ad hoc networks presented in literature. Moreover, for properties that have not yet been formally described, an intensive work of simulation has been done, and the results are presented and described.

Finally, the following three different scenarios are presented and analyzed:

- Nodes initially uniformly distributed, no mobility or Brownian motion (Section 3.2). This is the model usually assumed for sensor and static networks, with many important theoretical results about it. Destinations are chosen uniformly at random.

- Nodes initially uniformly distributed, Random Waypoint mobility (Section 3.3). This is the simulation scenario of many well established works in ad hoc networking. Destinations are chosen uniformly at random.

- Nodes that arrive at different moments, connecting to already connected nodes, no mobility or very low mobility (Section 3.4). This can be a simplified case of network formation in a realistic case, and it is an original model to our best knowledge. Communications are all directed to a central node.

For every scenario, we will describe network connectivity and hopcount distribution.
3.1 General model

In all the following Sections, we will consider a network composed by $N$ nodes positioned on a square with side $R$.

All the nodes have the same fixed communication range $r$, and we assume that two nodes positioned at an Euclidean distance less than $r$ are connected (free space pathloss propagation model).

This hypothesis is clearly a simplification of a real case, where obstacles and interferences can make two nodes at distance less than $r$ disconnected, or even asymmetrically connected.

We will consider just combinations of initial spatial distributions and mobility models that yield to a steady state in nodes positions: namely, after an initial stabilization time, network at generic time $t$ is statistically indistinguishable from the same network at time $t + t'$ for any $t'$.

Density

There is not a unique, globally adopted notion of nodes density in wireless networks. However, all the measures must take into account the network extension ($R$), the transmission range ($r$) and the number of nodes ($N$). We use the average number of neighbors per node as a density measure (called normalized transmission range in [37, 43] for uniformly distributed nodes).

Let $C(x, y)$ be the area covered by a node placed in the location $(x, y)$ (for nodes not in the border, it will be a circle of radius $r$) and let $p(x, y)$ be the pdf of nodes spatial distribution. Ignoring the border effect (for $r \ll R$), the probability that a node is neighbor of a node in the location $(\bar{x}, \bar{y})$ is

$$\nu(\bar{x}, \bar{y}) = \int_{C(\bar{x}, \bar{y})} p(x, y)dx\,dy.$$  

The average number of neighbors will then be:

$$\Delta = N \int_{x=0}^{R} \int_{y=0}^{R} \nu(x, y)p(x, y)dx\,dy$$

Notation

In this section we quickly summarize the notation used throughout the whole Chapter. We present it as a table with the meaning of every used symbol, in order to enable quick look-ups to the reader.

---

1The unique exception will be the model of nodes connecting to already connected nodes. In that case, in different times the network may exhibit different densities.
### Chapter 3. Probabilistic Topological Properties

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>number of nodes composing the network</td>
</tr>
<tr>
<td>$R$</td>
<td>length of the network area side</td>
</tr>
<tr>
<td>$r$</td>
<td>communication range</td>
</tr>
<tr>
<td>$\rho = \frac{r}{R}$</td>
<td>ratio of communication range over network extension</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>density (average number of neighbors per node)</td>
</tr>
<tr>
<td>$p(x,y)$</td>
<td>spatial density of nodes (probability a node is in the location $(x,y)$)</td>
</tr>
<tr>
<td>$C_N^p(i)$</td>
<td>Probability that at least $i$ paths exist between two nodes</td>
</tr>
<tr>
<td>$H_N^p(h)$</td>
<td>Probability that two nodes are distant $h$ hops (using the shortest path)</td>
</tr>
<tr>
<td>$H_p(\Delta)$</td>
<td>Average number of hops</td>
</tr>
</tbody>
</table>

#### 3.2 Uniformly distributed nodes, no mobility

As already mentioned, in this scenario nodes are initially positioned uniformly at random in the network area, and they do not move. Results in [56] show that this model can also fit the case of Random Waypoint with high pause time (i.e. low mobility) or Brownian mobility model (i.e. totally random nodes movement).

Message destinations are chosen uniformly at random among all the nodes in the network.

Such a model can describe for example sensor networks implementing a data centric storage ([55, 59]), or ad hoc networks with very low mobility.

Under these assumptions, at any given time $t$ nodes can be considered uniformly distributed.

##### 3.2.1 Expected number of neighbors

Let $\rho$ be the ratio of transmission range over network side, i.e. $\rho = \frac{r}{R}$. If $\rho$ is very small, namely when $r \ll R$, the probability to have $n$ nodes in a circle with area $\pi r^2$ follows a Poisson law ([25]) with parameter (and expected value)

$$\Delta = N\pi\rho^2.$$

This value is the number of expected neighbors per node (incremented by one) and it will be used as measure of density in the remainder of the Section.

##### 3.2.2 Connectivity

Connectivity of uniformly distributed wireless networks was intensively studied in the packet radio networks context, and it is currently being investigated in the (similar) context of ad hoc networks.

In analogy with graphs, a fixed size network is said connected if there exist a path between any pair of nodes. Since the spatial distribution follows a random law, connectivity in probability is commonly used. A network is connected if the probability to have unreachable nodes is low, and tends to 0 as the number of nodes
increases. In practice, a network is connected if it is composed by a giant connected component, and by some nodes not connected to that component, whose number is finite as total number of nodes goes to infinity.

Thus, the problem of the network connectivity is to study which values of $r$ causes a network to be connected with high probability. Smaller values of $r$ lead to less energy consumption and less interferences, making the problem very interesting from a practical point of view.

In [44] it was shown that, using the ALOHA protocol, if $r$ is set such that every node have at least six neighbors, then the probability to have a connected network tends to 1. For this reason, six was called the magic number.

In [62] the magic number was changed to 8 in the case of ALOHA, and to a number between 5 and 8 for a large number of other MAC protocols.

Subsequent works tended to confute the magic number hypothesis: in [53] it was shown that for any constant number of neighbors, it exists a large enough network disconnected with high probability.

In [39] it is shown that if nodes sets their transmission range $r$ such that

$$\pi r^2 = \frac{\log(N) + c(N)}{N}$$

then the network is connected with probability 1 if and only if $\lim_{N \to \infty} c(N) = +\infty$.

In [67], it is shown that if and only if the average number of neighbors of every node grows like $\Theta(\log(N))$ then the network is connected. Thus, no magic number exists, because for any fixed average number of neighbors, there exists an $N$ big enough to have a disconnected network.

Finally, in [33] it was proven that if the transmission power is set in order to have $r = c \sqrt{\frac{\log N}{N}}$, then the network is connected with high probability only if $c > 1$. If $c < 1$, moreover, the size of the component not connected to the giant graph is proportional to $N^{1-c}$.

### 3.2.3 Existence of multiple paths

Another interesting property is the probability to find different paths among two randomly chosen nodes. The usage of multiple paths among nodes may improve reliability and throughput. Moreover, if selfish nodes are present in the network, it may be necessary to drop a non working path and to chose a new one.

Given the example made in Chapter 4, we consider fully disjoint paths, i.e. paths composed by different intermediate nodes. The probability to find two paths that can have some nodes in common is obviously greater. See Chapter 4 for the motivation of this choice.

We will denote $C_N^\rho(l)$ the probability to find at least $l$ disjoint paths in a network with parameters $\rho$ and $N$.

We run a set of simulations to estimate such a probability for low values of $l$, and in two network scenarios.
CHAPTER 3. PROBABILISTIC TOPOLOGICAL PROPERTIES

0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
6
8
10
12
14
16
18
20
22
24
26
Frequency
D
1 path
2 paths
3 paths
4 paths
(a)
(b)
Figure 3.1: Probability to find multiple disjoint paths.

In figure 3.1(a) the results for $C_N^{0.05}$ with varying $N$ are presented, while in Figure 3.1(b) are presented the results for $C_N^{0.03}$.

3.2.4 Hopcount distribution

The problem to determine the probability that the shortest path between two randomly chosen nodes is long $h$ hops is, at our best knowledge, still open.

When studying the distribution of route lengths, the assumption commonly made is that the used routing always finds the shortest path in the connection graph.

In [43], investigating the relation between node density and capacity of ad hoc networks (completing the picture shown in [40]), authors give an estimation of the average number of hops between two randomly chosen nodes. If the density of nodes is high enough, then the hop count is proportional to the physical distance between two points in the plane (authors position nodes in the unit disk). Thus, calling $H_N^h(h)$ the probability function, i.e. the probability that in a network with density $N \pi \rho^2$ two nodes are $h$ hops distant, it can be approximated with the probability that two nodes are physically at distance $d$, with $(h - 1)r < d < hr$ (for $h > 0$).

The latter probability density, for a square with area 1, is the following:

$$p(d) = \begin{cases} 
2d(d^2 - 4d + \pi) & d \leq 1 \\
2d(\text{arcsin}(\frac{2-d^2}{d^2})) + 4\sqrt{(d^2 - 1) - d^2 - 2} & d > 1
\end{cases}$$

Unfortunately, this distribution describes only situations with an extremely high number of nodes, while usually it is very important to model the quality of the paths when the density approaches the critical value discriminating between a connected network and a partitioned one. In particular, in the analysis shown in Chapter 4 we will need an estimation of the path length increment when nodes density decreases.
In [9] it is given the explicit formula for the probability of having a path length of 1 and of 2. It corresponds to finding the values of $H^N_N(1)$ and $H^N_N(2)$. In [43], it is conjectured that the average hopcount between two nodes, namely $E[H^0_N]$, behaves like $\frac{1}{N}$.

In Appendix A it is shown that the average number of hops is better described by $\frac{1}{c}$, for a constant $c > 1$.

### 3.3 Uniformly distributed nodes, Random Waypoint mobility

Disliked by many researchers, the Random Waypoint Mobility Model (RWP - [41, 14]) has become a very important mobility model since a huge number of published works validate theoretical results against such a model.

The model is very simple, but its statistical properties are proving to be quite complex to be described.

Nodes are initially positioned uniformly at random in the network area. Every node independently chooses a random destination (a waypoint) uniformly in the field. Then, it moves from its current position to the selected destination with constant speed, picked uniformly at random in the interval $[v_{\text{min}}, v_{\text{max}}]$. At destination, nodes choose a pause time picked uniformly at random in the interval $[p_{\text{min}}, p_{\text{max}}]$. This process is repeated for the whole network lifetime.

A very important result is presented in [69], where it was shown that $v_{\text{min}}$ must be not null. Otherwise, the model has not a steady state average speed and for long simulation times, nodes will be permanently moving to far away waypoints at lowest speed. Thus, results obtained with null minimum speed and long simulation time are not accurate.

Since, at our best knowledge, there are no established results in hopcount distribution and multipath connectivity, we fixed the network parameters for all the simulations to $R = 1$, $r = 0.05$, $v_{\text{min}} = 0.02$, $v_{\text{max}} = 0.04$, $p_{\text{min}} = p_{\text{max}} = 0$, which is a scenario similar to the one used in [46] but with high mobility (in order to have a scenario quite different from the no mobility one).

#### 3.3.1 Steady State Spatial Distribution

It was believed for a long time that, since nodes moved independently, choosing uniformly at random waypoints, speed and pause time, the steady state spatial distribution followed a uniformly distributed law.

In [11] was noted (by means of intensive simulations) that nodes tend to concentrate in the center of the network area. Authors also gave a formal description of position distribution for the unidimensional case (i.e. with nodes moving along a line), and gave an approximation for the two-dimensional case.
In particular, in a square network with side $R$, the steady state pdf of nodes location is well described by the following formula:

$$X(x, y) = q f_{X_p} + (1 - q) f_{X_m},$$

where $q$ is the pause probability, $f_{X_p}$ is the pdf of the pausing nodes, and $f_{X_m}$ is the pdf of the moving nodes.

The first component is a uniform distribution over the network area, while the second component is well described by:

$$f_{X_m}(x, y) \simeq \frac{9}{16R^6}(4x^2 - R^2)(4y^2 - R^2),$$

where for convenience the network is represented by the square $(-R/2, -R/2) : (R/2, R/2)$. Figures 3.2(a) and 3.2(b) shows the nodes density with two extreme value of the pause probability.

In [12] the hypothesis was further validated by a large amount of simulations. In particular, the hypothesis that nodes were distributed following a uniform law in the case or RWP mobility was shown to be invalid with a set of statistical tests (just in the case of very large pause times the hypothesis could hold).

In [10] the approximation result was enhanced and extended to a more general model, in which a certain percentage of nodes is not mobile.

In [51] the pdf was explicitly found, but given its complex form, the approximations in [10] and [11] are preferred for practical cases.
3.3.2 Expected number of neighbors

In [8] it is analyzed the expected degree per node in such a scenario. As for the spatial distribution, the average number of neighbors is conditioned by pausing nodes (uniformly distributed in the network area) and by moving nodes, i.e.

$$\Delta = qN \frac{\pi r^2}{R^2} + (1 - q)\mu,$$

where $q$ is the pausing probability and $\mu$ is the expected number of neighbors deriving from the spatial distribution described in 3.3.1 (with the method described in 3.1).

Again in [8] an approximation (based on the Taylor series expansion of the formula) is given, resulting (for $r \ll R$):

$$\Delta \approx N\rho^2 \frac{N\rho^2}{3}((4 - 2q + q^2) - 4q^2\rho - 3(1 - q)\rho^2).$$

In Figure 3.3(a) it is shown how the average number of neighbors is clearly linear in the number of nodes composing the network.

3.3.3 Connectivity

In [8] it is also presented a lower bound result for connectivity of networks under the random waypoint mobility model.

Fixing $R$, the $(N, r)$ pairs that yield the probability that a node is not connected lower than 0.01 are found. The probability of no isolated node in a network is an approximation of the probability that the network is connected (especially for high probabilities, like in this case).

In Figure 3.3(b) it is shown a qualitative graph of the increment of nodes (in percentage) needed to have a connected network under the RWP model with respect to a network composed by static uniform distributed nodes.

3.3.4 Existence of multiple paths

As in 3.2.3 we run a set of simulations to estimate the distribution $C_\Delta(i)$ for small values of $i$ and for $\Delta = 0.05$. In Figure 3.4 it is shown the probability to find up to 3 disjoint paths among random chosen source/destinations.

Comparing with Figure 3.1(a), it is clear that, for a fixed density, mobility causes less nodes to be connected to the giant component, and less disjoint paths exist among nodes.

3.3.5 Hopcount distribution

At our best knowledge there are not established results for hopcount, assuming uniformly distributed source destination pairs and routing able to find shortest paths.
(a) Average number of neighbors in a network with $\rho = 0.05$, $v_{\text{min}} = 0.02R$ and $v_{\text{max}} = 0.04R$.

(b) Increment of nodes needed to have a network connected when using a RWP model.

Figure 3.3: Connectivity under the RWP model.

Figure 3.4: A simulative study of $C_{0.05}(i)$ for small values of $i$. 
3.4. NODES POSITIONING IN THE NEIGHBORHOOD OF ALREADY CONNECTED NODES

In [57] are reported simulation results concerning average node distance, which at high enough densities is related to hop distance. As expected, the average distance of nodes is shorter than in the case of uniformly distributed nodes: this is given by the greater density of nodes near the network center.

In [8] an explicit formula for the nodes distance is given, and it is noted that the expected distance decreases with $q$: more mobility implies lower distances among nodes.

Since there are no results in this field, we run a set of simulations with network parameters described in Section 3.3 to determine the pdf of the hop count in such a case, and the variation of average hop count with nodes density.

In Figure 3.5(a) it is shown the pdf of the hop count for different values of nodes density (from a disconnected network to a very dense one, as shown in Figure 3.4).

3.4 Nodes positioning in the neighborhood of already connected nodes

When an ad hoc network is composed by users joining the network at different times, uniformly distribution of nodes is probably not a good system description.

We analyze the following scenario: at time 0 there is a single node with id 0 (for example an access point) positioned in the center of the network. At time $t_i$ the $i^{th}$ node arrives and it places itself in the area covered by one of the already existing $i - 1$ nodes. This assumption reflects the fact that a new user joining the network will look for a location covered by the network.

We assume that the newcomer node chooses a node uniformly at random, and it chooses uniformly at random a point inside the area covered by the chosen node.
Nodes do not disappear and do not move.

In this scenario the connectivity is not a problem: the network is assumed to be connected. For this reason we concentrate on nodes spatial distribution, average degree (i.e. how density is related with number of nodes) and hopcount distribution.

Moreover, for densities not extremely high, the value of $R$ and $r$ does not change the results: the only important parameter is $N$.

For this reason, we will omit the $\rho$ and $N$ indexes where not needed.

### 3.4.1 Steady State Spatial Distribution

The network formation process causes nodes to be concentrated in the center. The average degree of node $i$ is higher than the average degree of node $j$ if $i < j$, (see Section 3.4.2) and thus node 0 will have, on average, more nodes placed in its covered area (i.e. near the center) than other nodes placed in distant locations.

In Figures 3.6 are shown two scenarios with different node density.

Since we are not interested in spatial distribution, we will not describe the distribution of nodes in this scenario.

### 3.4.2 Expected number of neighbors

It is possible to give a lower bound on the expected degree per node by considering that every newcomer node attaches to one and only one node previously in the network. Clearly the probability that a node $j$ is neighbor of another node $i$ is lower in this case, resulting in a lower bound for the expected value.

Note that the graph created under this simplified model is a tree: every newly arrived node connects to its father and becomes one of its children.

Under this assumption, it holds the following

**Theorem 1.** The expected number of neighbors for node $i$ in a network composed by $N$ nodes is:

$$\nu_i^N = \begin{cases} \sum_{j=1}^{N-1} \frac{1}{j} & \text{if } i = 0 \\ 1 + \sum_{j=i+1}^{N-1} \frac{1}{j} & \text{if } i > 0 \end{cases}$$

**Proof.** It clearly holds $\nu_0^1 = 0$ (the first node has no neighbors in the beginning) and $\nu_i^{i+1} = 1$ (every node has exactly one neighbor when it arrives, by hypothesis).

When node $N$ arrives, it has probability $\frac{1}{N}$ to attach to any of the previously existing $N$ nodes.

Thus for $i < N$ it holds:

$$\nu_i^{i+1} = \frac{N - 1}{N} \nu_i^N + \frac{1}{N} \nu_i^N + 1 = \nu_i^N + \frac{1}{N}.$$
3.4. NODES POSITIONING IN THE NEIGHBORHOOD OF ALREADY CONNECTED NODES

Figure 3.6: Spatial distribution of nodes when newcomers connect to already existing nodes.

Unfolding the previous recurrence relation, and recalling that \( \nu_i^k = 0 \) for \( k < i \), we obtain the formula in the enunciation, with the different case for node 0 deriving from the fact it has 0 neighbors at its arrival time.

This theorem gives us a method to exactly compute a lower bound on the average degree of every node, but it is possible to obtain a more (visually) appealing result by approximating the sum in the formula.

**Corollary 1.** The function \( \nu_i^N \) can be bounded in the following way:

\[
\max \left( 1, 1 + \frac{\lfloor \log(N) \rfloor}{2} - \lfloor \log(i) \rfloor \right) \leq \nu_i^N \leq 1 + \lfloor \log(N) \rfloor - \frac{\lfloor \log(i) \rfloor}{2}
\]

**Proof.** It directly follows from the observation that

\[
1 + \frac{\lfloor \log(N) \rfloor}{2} \leq \sum_{i=1}^{N} \frac{1}{i} \leq 1 + \lfloor \log(N) \rfloor.
\]

However, the lower bound is not very accurate: in Figure 3.7(a) it is shown the average number of nodes generated for given densities: it should be observed that \( \Delta \simeq 0.1N \), while the lower bound presented in this Section is \( O(\log(N)) \).

\(^2\)The functions \( \nu_0^N \) and \( \nu_1^N \) take the same values.
3.4.3 Existence of multiple paths

For the way nodes connect to the network, at any density the network will be connected, i.e. $C_\Delta(1) = 1$.

In Figure 3.7(b) it is shown the value of $C_\Delta(i)$ for small values of $i$ ($\rho = 0.05$).

In this case, opposed to the two previously described scenarios, the existence of disjoint paths is more probable given the network formation process.

3.4.4 Hopcount distribution

The average hop count is upper bounded by the diameter of the network in hops. We begin this Section presenting an upper bound on the network diameter, and then we present some simulative results on relation of the average number of hops with density.

As in the previous Section, we will use the simplified model in which every node attaches to one and only one of the previously existing nodes. The diameter of the resulting graph (a spanning tree of the network) is clearly greater than the diameter of the network we are considering.

Under this simplified model, the nodes arrange themselves in a tree. At the beginning, there are just nodes 0 and 1 that form a sort of dipole (see Figure 3.8(a)). Then, after some node arrival, the random tree can be seen as the composition of the subtree attached to node 0 and the subtree attached to node 1 (see Figure 3.8(b), where heights are not meaningful). Note that the tree is not binary: nodes 0 and 1 have, in general, more than 2 sons.

In order to derive the expected height of the tree (which is our goal), it is useful to understand how these subtrees grow on average. In fact, given that every node chooses uniformly its parent, between the already existing nodes, it is very unlikely
that such a tree will be balanced: the third node will attach to one of the subtrees, that at the arrival of the fourth node will have probability $\frac{2}{3}$ to be chosen by the newcomer node.

Interestingly enough, every possible combination of sizes for the subtrees have almost the same probability to be reached, as shown by the following:

**Lemma 1.** Let $\text{Big}(n)$ be the random variable that counts the number of nodes in the largest of $T_0$ and $T_1$ when the total number of nodes is $n \geq 3$. This takes all the values between $\lceil \frac{n}{2} \rceil$ and $n - 1$ with the following probabilities:

- if $n$ is odd, then all the values have the same probability $\frac{2}{n-1}$;
- if $n$ is even, then $\text{Big}(n)$ is $\frac{n}{2}$ with probability $\frac{1}{n-1}$, and is any other value between $\frac{n}{2} + 1$ and $n - 1$ with probability $\frac{2}{n-1}$.

*Proof.* It is easy to check that the probabilities given form a discrete distribution. The proof that the values are correct is given by induction.

For $n = 3$, the fact is banally true. When there are only 3 nodes the largest tree will have 2 nodes with probability $1 = \frac{2}{3}$. When $n > 3$, let us assume the lemma is true for $i < n$, using that in order to show that it also holds for $i = n$, and let us suppose $n$ is even.

The probability to have $k$ nodes in the largest tree when the total number of nodes is $n$, for $n - 1 \neq k \neq \frac{n}{2}$ (which are border cases), is given by

$$P\{\text{Big}(n) = k\} = P\{\text{Big}(n - 1) = k - 1\} \cdot \frac{k-1}{n-1} + P\{\text{Big}(n - 1) = k\} \cdot \left(1 - \frac{k-1}{n-1}\right),$$

i.e. by the probability of having $k - 1$ nodes when there are $n - 1$ nodes times the probability for the newcomer node to attach to the largest tree plus the probability of having $k$ nodes when there are $n - 1$ nodes times the probability for the newcomer node to attach to the smallest tree. Now, we know both $P\{\text{Big}(n - 1) = k - 1\}$ and $P\{\text{Big}(n - 1) = k\}$ by an inductive step. Since $n - 1$ is odd, and the two probabilities are $\frac{2}{n-2}$. With some algebra it is easy to check that the wanted probability is exactly $\frac{2}{n-1}$.

For $k = n - 1$ we have the same result just by considering that $P\{\text{Big}(n - 1) = n - 1\} = 0$, and using just the first part of the formula.

Finally, if $k = \frac{n}{2}$, then $P\{\text{Big}(n - 1) = k - 1\} = 0$, and from the second part of the formula we get the value $\frac{1}{n-1}$.

The case of $n$ odd is similar, and its proof is left to the interested reader. \qed

Since we fully characterized the random variable that describes the nodes distributions, it is now possible to compute its average value:
CHAPTER 3. PROBABILISTIC TOPOLOGICAL PROPERTIES

(a) First 2 nodes.

(b) A random tree structure.

Figure 3.8: The tree creation.

**Theorem 2.** The limit of the average ratio of the size of the two subtrees $T_0$ and $T_1$ is 3, i.e:

$$\lim_{n \to \infty} E \left[ \frac{\max(|T_0|, |T_1|)}{\min(|T_0|, |T_1|)} \right] = 3$$

**Proof.** For Lemma 1, the random variable $Sm(n)$, which counts the number of nodes in the smallest between $T_0$ and $T_1$ takes all the values $k$ between 1 and $\lfloor \frac{n}{2} \rfloor$ with the same probabilities that $Big(n)$ takes $n - k$. It is then possible to compute the expected value of $Sm(n)$ in the following way (we show the case for $n$ even):

$$E[Sm(n)] = \sum_{i=1}^{\frac{n}{2}} i \cdot Pr\{Sm(n) = i\}$$

$$= \frac{2}{n-1} \cdot \sum_{i=1}^{\frac{n}{2}} i - \frac{1}{n-1}$$

$$= \frac{2}{n-1} \cdot \frac{n^2}{2} - \frac{1}{n-1}$$

$$= \frac{1}{4} \cdot \frac{n^2}{n-1} + O(\frac{1}{n})$$

For $n$ odd, the expected value is exactly $\frac{n+1}{4}$, which is always $O(\frac{n}{4})$. The expected ratio between the sizes of the two subtrees is then

$$E \left[ \frac{\max(|T_0|, |T_1|)}{\min(|T_0|, |T_1|)} \right] = \left\{ \begin{array}{ll} \frac{4n^2-4}{4n^2+2n-4} - 1 & \text{for } n \text{ even} \\ \frac{4n^2}{4n^2+2n-4} - 1 & \text{for } n \text{ odd} \end{array} \right.$$ 

For this reason,

$$\lim_{n \to \infty} E \left[ \frac{\max(|T_0|, |T_1|)}{\min(|T_0|, |T_1|)} \right] = 4 - 1.$$

By Theorem 2, one of the first two nodes is the root of a tree that, on average, contains three times the number of nodes in the tree rooted by the other node.

The same argument can be recursively applied to all the other nodes that become first child of an already connected node.
3.4. NODES POSITIONING IN THE NEIGHBORHOOD OF ALREADY CONNECTED NODES

Figure 3.9: Average hop count.

This causes a structure quite regular, i.e. is an unbalanced tree, in which every node is root of a subtree containing (on average) \( \frac{1}{4} \) or \( \frac{3}{4} \) of the nodes of the subtree routed by its father.

It is then not difficult to give the following bound:

**Lemma 2.** The average height \( h(n) \) of the random tree described in this Section is \( \Theta(\log(n)) \).

*Proof.* The height of such a random tree \( (h(n)) \), for its regular structure on average, is surely greater than the height of a similar tree with one forth of the nodes augmented by one (the case in which the subtree rooted by node 1 is smaller than the one rooted by node 0). This consideration leads to a lower bound of

\[
h(n) \geq \log_4(n) = \frac{\ln(n)}{2}
\]

On the other side, \( h(n) \) is lower than the height of a similar tree with three fourths of the nodes augmented by one (the case in which the subtree rooted by node 1 is larger than the one rooted by node 0). This consideration leads to an upper bound of

\[
h(n) \leq \log_4(n) \simeq 3.47 \cdot \ln(n).
\]

Then the tree height is bounded (both above and below) by a logarithm.

We run a set of simulations in order to understand how the average number of hops is related to density in the considered scenario.

For a given density, we generated 5000 different networks with enough nodes. Then, a random node was picked and its distance in hops from node 0 was computed. For every path length, we recorded its frequency and computed the
average path found.

In Figure 3.9 the results are shown. Figure 3.9(a) shows the distribution of the hop count for two densities. As the number of nodes increases, the average number of hops increases, as it is shown in Figure 3.9(b). The average hopcount is well described by the function $1.4 \log(\delta)^{1/2}$.

3.5 Conclusions

In this chapter we presented general concepts about topology of ad hoc networks. In particular, we introduced the node density, given in terms of average number of neighbors per node, and we characterized a network in terms of its initial node spatial distribution, mobility model and communications patterns.

We presented three different scenarios, describing for each of them its fundamental features: relation between number of nodes and density, connectivity and existence of disjoint paths among nodes, and probability distribution of hop count.

We reviewed the existing literature (Sections 3.2.2, 3.2.4, 3.3.1, 3.3.2 and 3.3.3), completing the missing part of the picture by means of simulations (Sections 3.2.3, A, 3.3.4 and 3.3.5).

The model and the bounds presented in Section 3.4 are original.
Chapter 4

Performance degradation due to selfishness: an analysis

Selfishness is an advantageous behavior as long as other selfish nodes are not present in the network. In fact, if almost all the nodes behave selfishly, nothing works.

It is often forgot that selfish nodes are rational entities that need to use the network: they try to save energy by not forwarding for other nodes, but if the network stops working, they must react to this situation.

Assuming that network problems can be caused only by the presence of selfish nodes, possible reactions to a network stall for a cooperative node can be:

- move to another location, hoping to find more cooperation,
- stop using the network, temporarily or permanently,

Moving to another location can be effective if the number of selfish nodes is not critically high, or if the destinations of all the connections become one hop neighbors. Moreover, it is possible only when the user has a full control on mobility.

Stop using the network temporarily (as a sender and as a forwarder), hoping that the problem is fixed in the meantime is the only convenient action.

On the other hand, a selfish node experiencing communication problems has the same options, but it can also start cooperating, hoping to raise a positive chain reaction that establishes again the functionality of the part of the network it is interested in. Note that selfish nodes (if they are more than one) have exactly the same problems caused by selfishness than cooperative nodes (the only difference is that the latter spend some energy to forward).

It is clear that the waiting strategy alone is not useful: things would not change if all the nodes stick to this strategy. Thus, the more rational behavior for a selfish node unable to communicate is to start cooperating, at least temporarily. This strategy helps also in case of application of punishment mechanisms to the network: after a punishment starts, the node can redeem by start cooperating.
Then, if $\gamma$ is the maximum loss rate (or, alternatively, minimum goodput) that a node is willing to tolerate, it can be deducted that this will be the maximum average loss rate (or minimum average goodput) offered by the network (assuming that only selfishness can change their value).

In fact, either selfish nodes are not enough to have a loss rate of $\gamma$, or the nodes will behave as described: the percentage of selfish nodes in the network would be auto-limited in order to never pass the $\gamma$ threshold.

This Chapter presents a simple analysis of the performances degradation a node experience because a given fraction of selfish nodes is present in the network. Differently from the simulative results presented in [46] and [47], we consider here the node (and not the system) point of view. Also the selfish node performances are analyzed in order to understand which is the maximum fraction of selfish nodes that can be tolerated. Such a value, if found, could be an upper bound on the number of nodes acting selfishly at the same time.

The analysis is voluntarily kept simple: it can be used as a basis to build a more realistic model. To show the benefit such a model could yield, we compare the performances in absence of countermeasures to selfishness and in presence of a very simple mechanism which just changes path when the chosen one is not working. However, given the simplifications made, we do not compare analytical results with simulations, leaving this as future work when a more precise model will be developed.

The plot of analytical results are shown in the three scenarios analyzed in Chapter 3:

- nodes uniformly distributed, no mobility,
- nodes following a RWP mobility model,
- nodes positioning near nodes already in the network.

### 4.1 General model

In this chapter, we assume an ad hoc network deployed on an square area with fixed size $R$. The signal propagation model is the free pathloss model, i.e. every node can communicate with all the nodes that are inside its covered area (a disk of radius $r$ centered at the node position). We do not consider any specific mobility model or spatial distribution: we assume that the combination of mobility model and initial spatial distribution yields a steady state spatial distribution that allow to study the network at discrete intervals.

The network is composed by $N$ nodes, a fraction $\alpha$ of which (i.e. $\alpha N$ nodes) is selfish. Selfish nodes are uniformly distributed among all the nodes. Thus, calling $\Delta$ the density (expressed in terms of average neighbors per node, Chapter 3), we will denote by $\Delta|_{\alpha}$ the density given by the same network parameter but for only $(1 - \alpha)N$ nodes.
4.2. A SIMPLE SELFISHNESS AVOIDANCE METHOD

We assume the pdf of the hop count is known, and we will denote $H_\Delta(h)$ the random variable describing the hop count. The average number of hops between two nodes is denoted by $\bar{H}_\Delta$.

We describe the system as a sequence of snapshots. Every snapshot is the resume of $t$ seconds. The value of $t$ is determined in order to be small enough to consider the network static during a single snapshot.

We analyze two different classes of selfish nodes, assuming that in a given network only node of a single class can be present:

- pure selfish nodes, namely nodes that do not participate to any protocol (in particular, route discovery),
- tricky selfish nodes, which participate to other protocols but do not forward traffic (as described in [46]).

The existence of the latter class is justified only in case of enforcement mechanisms which monitors (for example) the route establishment phase. Otherwise, a selfish node would avoid also that waste of energy.

Assuming that all packets have the same size, we will use the notation $c_\sigma$ and $c_\rho$ for the cost of sending and receiving a packet. We assume that the average cost for the eventual channel contention protocol (for example, the RTS/CTS message exchange in 802.11 networks) is included in the $c_\sigma$ and $c_\rho$.

The network is composed by homogeneous nodes. During a snapshot, every node generates (on average) $p$ packets to be sent to $d$ different destinations. We assume on average $\frac{p}{d}$ packets are sent to every destination.

We are interested in the average energy consumption of cooperative and selfish nodes in presence of $\alpha$ selfish nodes, and in the average goodput in the same conditions.

Note that a selfish node will consume the same amount of energy as a sender and receiver, because it can not avoid other selfish nodes in paths (the main cause of energy waste).

Pure selfish nodes will not spend energy in forwarding, while tricky selfish nodes have a waste of energy described in Section 4.3.2.

We are also interested in describing the goodput, i.e. the ratio of delivered packets over packets sent as a source.

Note that the term network throughput used in [46] is equivalent to the goodput described later in this Chapter (with $l = 1$ and $k = \frac{p}{d}$).

4.2 A simple selfishness avoidance method

Normally it is assumed that, in absence of cooperation enforcement, all the packets are sent through the route returned by the routing algorithm. If one or more selfish nodes compose the route, all the packets are lost.
We present a very basic mechanism that does some path-probing before to send all the traffic along the route. If a route is not working, a new one is chosen up to a maximum number of trial.

The first packet sent after a route is established, or after a new snapshot has started, must be explicitly acknowledged by the destination. Until the reception of acknowledgement, no other packets are sent. If no acknowledgement is received after a timeout, the packet is re-sent up to $k$ times (we will use $k = 3$ in our formulas).

If the route is not working, a new route composed by nodes not present in the non working one is chosen, if available. The new route is chosen completely disjoint from the previous in order to lower the probability to insert a selfish node, but other rules may be used (comparison of two routes sharing some nodes could lead to useful information to individuate the selfish nodes that caused the previous route to fail).

Up to different $l$ routes are used (if found) before to give up.

We denote $C_\Delta(i)$ the probability that there exist at least $i$ disjoint routes between any two nodes.

Note that from the analysis point of view, the scenario in which no mechanism is used can be treated as if the mechanism is applied with $k = p$ (i.e. all the packets are always sent) and $l = 1$ (i.e. no alternative routes are considered).

### 4.3 Evaluation of energy spent in presence of selfish nodes

#### 4.3.1 Energy spent as a sender

We compute the average energy spent by nodes as sources of communication: the amount is the same for selfish and cooperative nodes.

**No selfish nodes**

In the case $\alpha = 0$, every node sends on average $\frac{p}{2}$ packets for every destination, if the path could be established.

The average spent energy as a source is then $C_\Delta(1)pc_\sigma$.

**Pure selfish nodes**

In this case, the energy spent by senders is on average $C_{\Delta\alpha}(1)pc_\sigma$, i.e. the probability to find a path to the destination times the cost to send $p$ packets.

**Tricky selfish nodes**

The probability to insert $i$ selfish nodes in a path with length $h$ is the same than to draw $i$ red balls in $h$ extractions from a box containing $\alpha N$ red balls and $(1 - \alpha)N$ white balls. This is described by a hypergeometric law with parameters $N - 2$ (the
4.3. EVALUATION OF ENERGY SPENT IN PRESENCE OF SELFISH NODES

number of nodes that can be chosen), \( h - 1 \) (the number of extractions) and \( \alpha N \) (the number of selfish nodes). The probability to establish a working path is then (approximating \( N - 2 \) with \( N \))

\[
p_{\text{succ}} = \sum_{h} H(h) \frac{N(1 - \alpha)!((N - h)!)^2}{N!((N(1 - \alpha) - h)!)^2} = \sum_{h} H(h) \prod_{i=0}^{h-1} 1 - \frac{\alpha N}{N - i}
\]

With a rough approximation\(^1\), the \( i^{th} \) chosen route will be working with probability \( C(i) p_{\text{succ}} \), i.e. the probability to find an \( i^{th} \) path among two nodes times the probability to have a selfish-free path.

The sequence of failed attempts can be modelled with a random variable \( X \) quite similar to a geometric one. So, it holds:

\[
p(X = x) = \begin{cases} C(x + 1) p_{\text{succ}} (1 - p_{\text{succ}})^x & \text{if } x < l \\ 1 - \sum_{i=1}^{l} p(X = i) & \text{if } x = l. \end{cases}
\]

Note that:

- the probability is 0 for values larger than \( l \) (since there are no more than \( l \) trials),

- the average number of trials before a valid route is found is \( E[X] = \sum_{i=1}^{l} ip(X = i) \),

- the probability to find a path to the destination is \( P(d) = \sum_{i=1}^{l} C(i) P(X = i - 1) \),

The average number of packets sent will then be \( kdE[X] + P(d)(1 - P(X = l))p \).

Multiplying this value for \( c_{\sigma} \), we obtain the energy spent on average.

4.3.2 Energy spent as a forwarder

Energy spent to forward packets for other nodes is what selfish nodes try to eliminate: pure selfish nodes will not spend any energy for this process (even if in reality they spend to receive the broadcast packets used for example in routing and to discard them), while tricky selfish nodes will spend energy to participate to the routing protocol. However, such an energy is not considered in this analysis.

\(^1\)The approximation is caused by the fact that the hop count distribution, at the \( i^{th} \) attempt, is not exactly the same as at the first attempt, given that the nodes composing the best route have been discarded. However, for a few attempts, this difference is small.
CHAPTER 4. PERFORMANCE DEGRADATION DUE TO SELFISHNESS: AN ANALYSIS

No selfish nodes
For every source-destination pair, the probability that a node \( i \) is chosen as a forwarder is \( \frac{H_{\Delta - 1}}{N-1} \). In fact, node \( i \) is chosen as a forwarder with probability

\[ P(i \text{ is not destination}) \times \sum_h H_{\Delta}(h) \times P(i \text{ is one of the } h - 1 \text{ forwarders}) \]

The first probability is \( \frac{N-2}{N-1} \), and the second one is \( \frac{h-1}{N-2} \).

So, the average number of forwarded packets will be \( p(\Delta - 1) \), for an average total cost of \( C_{\Delta}(1)(c_{\rho} + c_{\sigma})(\Delta - 1)p \).

Pure selfish nodes
The eventual change in energy spent in this case is caused by the reduced probability to find a path due to the lower density, by the higher average number of hops in a lower density network and by the higher probability to be inserted in a path.

Then, the average total cost of forwarding will be: \( C_{\Delta}(1)(\frac{N-2}{N(1-\alpha)-2})(\Delta_{\alpha} - 1)p(c_{\rho} + c_{\sigma}) \).

Tricky selfish nodes
In order to quantify the packets a cooperative node has to rely for other nodes, it is necessary to estimate the probability that a node is asked to forward in a faulty route and the probability that the same happens in a safe route, for any source.

Recalling that the random variable \( X \) models the sequence of attempts made by a generic source, the probability that a node is in a faulty route (denoted as \( P(f) \)) can be computed in the following way (partition of the certain event):

\[ P(f) = \sum_{i=1}^{l} P(f \mid X = i)P(X = i) \]

With a rough approximation, if \( i \) is not large, given that if a source uses \( i \) faulty routes, then it picks up on average \( i(\Delta - 1) \) relay nodes uniformly at random, it holds \( P(f \mid X = i) \approx \frac{i(\Delta - 1)}{N-1} \).

Thus,

\[ P(f) \approx \sum_{i=1}^{l} \frac{i(\Delta - 1)}{N-1}P(X = i) \]

The probability to be inserted in a successful path as a forwarder can be similarly found:

\[ P(s) = \sum_{i=0}^{l-1} P(s \mid X = i)P(X = i) \]
With the same approximation, \( P(s) \) can be approximated as follows:

\[
P(s) \simeq \sum_{i=0}^{i-1} (1 - P(f|X = i)) \frac{\bar{H}_\Delta - 1}{N - 1} P(X = i)
\]

We ignore the fact that being inserted in a faulty path not always causes a node to have to forward packets. If a node is inserted in position \( i \) of the path and there is a selfish node in a position \( j < i \), then no packets will arrive to the node to forward. However, if the number of trials with the probing packet \( (k) \) and the number of different routes used \( (l) \) are small, the increase of the probability to have to forward a packet is very small.

The average energy spent to forward packets generated by other nodes is then:

\[
E_f \simeq ((N - 1)kdP(f) + (N - 1)pP(s))(c_\sigma + c_\rho)
\]

A selfish node will not send any packet, but it will receive all the packets, saving the \( c_\sigma \) component in the last formula. Moreover, for a selfish node it will hold \( P(s) = 0 \).

### 4.3.3 Energy spent as a receiver

The energy spent as a receiver is given by the same formulas of the sender case, substituting \( c_\sigma \) with \( c_\rho \) for the case of pure selfish nodes and no selfish nodes.

In the case of tricky selfish nodes, we have the same formula as for the sender, without the \( kdE[X] \) term (a receiver will not clearly receive packets sent over faulty routes).

### 4.4 Evaluation of goodput in presence of selfish nodes

#### 4.4.1 No selfish nodes

Since we are ignoring the loss of packets for causes not related to selfishness, and the control packets (for example to discover routes, etc), goodput in this case will be 1. Namely, all the sent packets arrive to destination.

#### 4.4.2 Pure selfish nodes

Also in this case, since selfish nodes do not appear as forwarders, all the sent packets will arrive at destination.
CHAPTER 4. PERFORMANCE DEGRADATION DUE TO SELFISHNESS: AN ANALYSIS

4.4.3 Tricky selfish nodes

The goodput analysis is different when the simple selfishness detection mechanism described in 4.2 is used, and for this reason we split the discussion in two cases.

Selfishness avoidance mechanism

As noted in Sections 4.3.1 and 4.3.2, the average number of sent packets is \( dkE[X] + p(1 - P(X = l)) \), while the average number of received packets is \( p(1 - P(X = l)) \).

The goodput is the ratio of these two values: thus, considering only packet loss due to selfishness, the average goodput will be \( \frac{p(1 - P(X = l))}{dkE[X] + p(1 - P(X = l))} \).

No selfishness avoidance mechanism

The random variable \( X \) can have just two values: 0 (i.e. success) or 1 (i.e. failure). The probability to succeed is \( C(1)p_{suc} \), while the probability to fail is \( 1 - C(1)p_{suc} \), which is also the expected value of \( X \).

The average number of packets sent as a source is then \( C(1)p \).

The probability that a cooperative node is inserted in a faulty route as a forwarder is \( (1 - C(1)p_{suc}) \frac{H_{\Delta - 1}}{N - 1} \), while the probability to be inserted in a working route is \( (C(1)p_{suc}) \frac{H_{\Delta - 1}}{N - 1} \), with an average total number of packets to forward \( (H_{\Delta - 1})p \).

The goodput is given by \( C(1)p_{suc} \).

In the next section, fixing all the parameters, we will show some numerical data relative to this case.
4.5 Examples

The formulas found in the previous sections are highly parametric: as shown in Chapter 3 the values of $\Delta$, $C_\Delta(l)$, and $\bar{H}_\Delta$ heavily depend on the network analyzed.

In this Section we present their numerical instantiation of the formulas in the three realistic scenarios described in Chapter 3.

We consider $R = 1$, $r = 0.05$, $\Delta = 20$, $p = 100$ and $d = 2$. Moreover, we will use as costs $c_\sigma = 1$ and $c_p = 0.5$.

4.5.1 Uniformly distributed nodes, no mobility

It is clear that in this model, the energy spent in the forwarding phase is much higher than the energy spent for sending and receiving: a node is asked to participate, on average, to a number of routes equal to the average path length (minus one). In large networks, like the one used in the example, this value can be of the order of tens.

**Pure selfish nodes**

In Figure 4.1 it is shown the energy consumption, of a sender and a receiver for cooperative and selfish nodes (Figure 4.1(a)) and of a forwarder (Figure 4.1(b)).

The energy used to send and receive decreases for extremely high percentages of selfish nodes, due to the low probability in finding a route to the destination at the densities resulting by not involving selfish nodes.

Inversely, the energy for forwarding other nodes packets, dramatically increases as more selfish nodes are introduced, because the routes are on average longer and not selfish nodes are asked more frequently to forward.

Note that, in the best case, the energy spent to forward is almost 12 times the energy spent to send and receive, motivating a selfish strategy.

The energy spent as a sender also represents the fraction of generated packets that is sent with success (scaled by a $10^2$ factor), while, as said in Section 4.4.2, the goodput is 1, given that packets are sent only if a route can be established (we are not considering packet loss caused by interferences, congestion, etc).

From the Figures, it is clear that even a consistent fraction of selfish nodes (up to 50%) causes almost no degradation in number of packets that every node can successfully send, but increases the energy consumption of the relay nodes intolerably (more than the 100%). For lower nodes densities, smaller fractions would have the same effect.

**Tricky selfish nodes**

In Figure 4.2, the energy consumption in different communication phases and average goodput when tricky selfish nodes are introduced is shown. We are interested
in comparing the scenarios in which $k = \frac{P}{a}$ and $l = 1$ (i.e. no selfishness avoidance is used) and $k = 3, l = 3$ (i.e. a packets is sent up to 3 times if the path is not working, and up to 3 different routes are used).

Note that, when no probing packet is used, the sender always sends all its packets, keeping constant the energy cost of sending (the same happens for forwarding\(^2\)), while the increase in selfish nodes causes a smaller number of packets to be received.

Reducing the value of $k$ and changing path when another one is available, allows a reduction in energy spent in sending and forwarding, because just one packet is sent ($k$ times) instead of the $p/d$ that would be lost.

The increased energy consumption for receiving reflects an increase in goodput (Fig. 4.2(c)). However, the increased goodput is only partially due to the usage of different routes: a large part of this increase is due to the smaller number of packets sent by sources.

### 4.5.2 Uniformly distributed nodes, RWP mobility

In this Section, we analyze a different scenario, considering nodes mobility under RWP model, with parameters described in Chapter 3.3.

**Pure selfish nodes**

Figure 4.3 shows the average energy cost of all the communication phases.

The depicted situation is very similar to the one of Figure 4.1, with less energy spent to forward because of the shorter average paths (caused by mobility). The same mobility induces a smaller number of paths to be found for a given density, causing the selfishness to affect more the goodput than in the static case.

**Tricky selfish nodes**

In Figure 4.4 the performance of a network where tricky selfish nodes are introduced is shown.

The data are very similar to the static case, with a little gain in goodput when using $k = 3, l = 3$.

### 4.5.3 Nodes attaching to already connected nodes, no mobility

We conclude the examples by considering the very simple model analyzed in Chapter 3.4. This model shows several peculiarities with respect to the previously analyzed ones: it offers perfect connectivity at any density, and its average path length increases with density.

\(^2\)Actually, as the number of selfish nodes increases, it is very unlikely that a packet to forward reaches a cooperative node.
4.6. DISCUSSION

Pure selfish nodes

Given the properties of the model under examination, pure selfish nodes have a less dramatic impact than in other cases. In fact, a path can always be established, even for extremely high percentages of non-cooperative nodes. Figure 4.5(a) shows that the energy cost of sending and receiving remains constant (and so does goodput, which is 1 for any percentage of selfish nodes).

The relay nodes experience an increase of energy used to help other nodes, but such an increase is lower than in previous cases, because the new found paths are in general shorter when less nodes compose the network. Figure 4.5(b) shows the energy cost of forwarding.

Tricky selfish nodes

Tricky selfish nodes have a devastating effect also under this peculiar model. As shown in Figure 4.6, the number of sent packets (when using $k = 3, l = 3$) decreases almost linearly with the fraction of selfish nodes (Figure 4.6(a)), and so does goodput. The latter can be increased to quite high levels, but only by decreasing the number of sent packets (Figure 4.6(c)).

4.6 Discussion

We analyzed the impact of two classes of selfish nodes on the performance of ad hoc networks, ignoring other causes of packet loss.

We assumed that selfish nodes discard all the data packets they should forward. Let us consider nodes that fix a certain percentage $\beta$ of packets to discard among all the received ones. This can be represented by having $\alpha \beta N$ totally selfish nodes (instead of $\alpha N$) in the network.

Tricky selfish nodes, namely nodes which participate to routing but not to data packet forwarding, have a strong effect in network performances, while the degradation in goodput caused by pure selfish nodes is visible just when their number is so high that the real density of the network falls below the connectivity threshold. However, their effect is noticeable in the increased energy consumption for the forwarding phase in cooperative nodes.

In reality, we believe that a tricky selfish behavior is less dangerous (and less advantageous) than a pure selfish one. In fact, from all the plots it is clear that they also suffer a high decrease in goodput. For the informal discussion made at the beginning of this chapter, if we fix at $\gamma$ the minimum goodput a node is willing to tolerate before to change strategy, it may be found the maximum percentage of nodes simultaneously acting in a selfish way once network parameters are known. For example, in the static scenario presented in Section 4.5.1, fixing $\gamma$ to 0.8, less than the 20% of nodes in the network will be selfish. In reality the percentage would
be much smaller if we define the goodput as the number of delivered packets over
the number of generated packets.

Moreover, the presence of tricky selfish nodes is justified only by the presence of
a cooperation enforcement mechanism that punish just who does not perform route
discovery: if no mechanism is used at all, it is more plausible that pure selfish nodes
will be present. In fact, tricky selfish nodes do not experience a zero cost in forward:
they have to actively participate to the routing protocol, and they must receive all
the packets they agreed to forward, since their MAC address is used in the MAC
envelope of packets.

The methodology described in this chapter was useful to describe the advantages
offered by a simple mechanism which probes all paths before to start sending traffic
on them: we believe that with slightly more realistic hypotheses on route formation
the framework could be extremely useful in describing real cooperation enforcement
methods and in comparing them in an objective way.

4.7 Conclusions

In this Chapter we presented a simple performance model for generic ad hoc net-
works, in which a certain fraction of nodes is selfish. We described two classes of
selfish nodes, one which does not participate to routing (but generates traffic) and
another which allows its components to be inserted in routes, discarding all the
traffic that they should forward.

The analysis is voluntarily simplistic, in order to derive simple formulas with
basic computation. We leave as future work a more realistic model to be validated
against simulations.
4.7. CONCLUSIONS

(a) Energy spent to send and receive.

(b) Energy spent to forward for other nodes.

(c) Average goodput.

Figure 4.2: Energy spent in different communication phases and average goodput when tricky selfish nodes are present in the network. Static nodes, uniformly distributed.
(a) Energy spent to send and receive.  (b) Energy spent to forward for other nodes.

Figure 4.3: Energy spent in different communication phases when pure selfish are present in the network. RWP mobility model.
4.7. CONCLUSIONS

Figure 4.4: Energy spent in different communication phases and average goodput when tricky selfish are present in the network. RWP mobility model.
(a) Energy spent to send and receive.  
(b) Energy spent to forward for other nodes.

Figure 4.5: Energy spent in different communication phases when pure selfish are present in the network. Always connected network.
Figure 4.6: Energy spent in different communication phases and average goodput when tricky selfish are present in the network. Always connected network.
Chapter 5

A formal description of punishment systems

In Chapter 4 we analyzed the impact of two classes of selfish nodes on the performances of a generic ad hoc network. With a rather informal discussion, we pointed out that a non malicious selfish node will limit its non cooperative behavior when network starts to collapse due to the high percentage of selfish nodes. While this argument reduces the estimation of the damage that can be caused by selfish nodes, it also clearly states that a degradation in performances of the network is to be expected. And a small average goodput reduction could mean that some nodes are temporarily cut off from the network services.

Thus, some effective countermeasures to selfishness are still needed in order to guarantee the correct network functioning and a fair distribution of energetic costs to keep the network up.

In this chapter, we present and analyze two general models, each able to formally describe different properties that any punishment mechanism must have.

We model a perfect world, in which every node knows exactly how many of its packets have been forwarded by its neighbors in the past. Even with this optimistic (and unrealistic) assumption, we will prove that a perfect mechanism based only on local observations can not exist, mainly for two reasons:

- a punishment could always raise a chain reaction which leads to “turn off” entire parts of the network,
- mobility may turn the punishment useless: a selfish node will change neighborhood often enough to be able to elude the punishment.

We conclude the Chapter with another criticism, based on energetic considerations, to punishment mechanisms based on promiscuous listening of the medium.
5.1 Network Model

This section presents the model assumptions and definitions employed in our analysis. Our assumptions are general enough to include many typical ad hoc networks scenarios.

We are interested in ad hoc networks whose nodes are part of many different domains managed by different administrators. Scenarios like rescue or military operations, with all the nodes managed by a single administrator are in fact not likely to suffer from lack of cooperation among nodes.

In this Chapter we focus on packet forwarding functionality only: other protocols (like routing, etc) have also to be enforced ([47]), but we will leave this analysis to future works.

As in previous chapters, we assume that the network is composed by \( N \) nodes positioned in some area. We do not assume any spatial distribution or mobility model in particular.

We assume the time is divided in slots, i.e. fixed size intervals. The slot size is assumed to be short enough to allow to consider the network static during its duration. The network is asynchronous.

Every node has a limited sending capacity\(^1\) of \( L_i \) packets per slot, assumed not null.

**On the notation used in this chapter**

The following conventions have been used to simplify the usage of presented notation:

- Percentages (between 0 and 1) are denoted by Greek letters (the unique exception is \( \alpha \), which can be greater than 1 but is very similar to a normalized weight).
- Absolute numbers are represented by capital roman letters.

When a percentage is strictly related to an already defined value, then it is denoted with the Greek letter better representing the latter (e.g., \( \alpha \) for \( a \), \( \beta \) for \( b \), etc).

5.2 Cooperation as a strategic game

Nodes in an ad hoc network interact slot by slot by sending their own traffic and forwarding for other nodes the amount of traffic they decide. We describe such an interaction as a game, whose payoff is defined in order to take into account the eventual presence of a cooperation enforcement mechanism based on punishment.

\(^1\)It is not hard to extend the model in order to consider also limited capacity in reception, but we prefer to keep the model as simple as possible.
We analyze the properties that Nash equilibria ([50]) must exhibit, without explicitly find all the equilibria.

The multihop communication is modelled as follows: in every slot, every node sends traffic that is waiting in two queues: One queue holds packets generated by the node itself, while in the other are stored packets accepted to forward during the previous slot.

Moreover, nodes decide how much new traffic to forward is going to be queued, in order to send it during the following slot. Such a decision cannot be changed: once packets are accepted and stored, they will be automatically sent during the following slot.

At the end of every slot, nodes know how many of their packets have been accepted for forwarding. The last assumption models a theoretical perfect monitoring system, which would allow nodes to understand without any uncertainty how is being served their traffic (but not why is traffic eventually being discarded). Even with this strong assumption, we will show that punishment mechanisms are problematic to adopt.

We assume that at the beginning of every slot, generic node $i$ knowledge includes:

- $\mathcal{N}_i$, the set of its neighbors during this slot,
- $Q_i = \sum_{j \in \mathcal{N}_i} Q^j_i$, the number of packets generated by $i$ that has to sent during the slot, and how they will be split among neighbors,
- $A_i = \sum_{j \in \mathcal{N}_i} A^j_i$, the number of packets that $i$ has agreed to forward during previous slot, and through which neighbors they will continue their route. Recalling the model assumptions, these packets were queued during last time slot, and can not be discarded.

We assume that $Q_i$ is the same for all the slots, considering, for example, the average number of packets generated per slot. The number of packets that can be forwarded $A_i$ is then always limited to $\max(L_i - Q_i, 0)$.

It would be not hard to change the model in order to consider packets generated during the current slot as limiting the number of packets that can be accepted for forwarding. However, such a change would quite complicate the model, without adding any expressive power to it.

### 5.2.1 The game of forwarding

Communication during a single slot $k$ can be modelled by the following game:

**Players:** Nodes in the network are players

**Moves:** Actions of player $i$ consist in setting the following quantities (all between 0 and 1):
• \(\forall j \in \mathcal{N}_i, \sigma^j_i\): the fraction of the \(Q^j_i\) packets that \(i\) effectively sends to neighbor \(j\). From this set of values it is possible to compute the total fraction of sent packets

\[
\sigma_i = \frac{\sum_{j \in \mathcal{N}_i} \sigma^j_i Q^j_i}{Q^i_i}.
\]

• \(\Lambda_i\): the fraction of residual energy that \(i\) dedicates to forward other nodes packets. Recalling that \(L_i\) is the maximum number of packets that node \(i\) can send during a slot, if a node decides to send \(\sigma_i Q^i_i\) of its own packets, \(\Lambda_i\) is the percentage of the remaining \(L_i - \sigma_i Q^i_i\) packets that a node is offering to its neighbors,

• \(\forall j \in \mathcal{N}_i, \phi^j_i\): the percentage of forward requests by neighbor \(j\) that \(i\) will accept, if possible.

The simultaneous usage of \(\Lambda_i\) and \(\phi^j_i\) may seem, at a first sight, redundant. However, they are both necessary in order to describe any possible forwarding policy. For example, if \(\lambda_i\) was not used, then it would not be possible to model the following policy:

• use 50% of all the residual battery power to forward, wherever the incoming traffic comes from. This can be done by setting \(\Lambda_i = 0.5\) and, for every neighbor \(j\), \(\phi^j_i = 1\). Setting only the \(\phi^j_i\) to 0.5 would not be enough, since half of the received traffic can need more or less than the half of the residual energy.

On the other side, only with the \(\Lambda_i\) parameter, it would not be possible to specify the following policy:

• Forward all the traffic proceeding from neighbor \(j\), and no traffic from other nodes. Clearly, this policy can be described only using the values \(\phi^j_i\).

**Payoff:** After all players reveal their moves simultaneously, each of them is aware of the treatment received by neighbors\(^2\).

In particular, the following information is known by node \(i\) after moves are played:

• \(S_i = \sum_{j \in \mathcal{N}_i} S^j_i = \sum_{j \in \mathcal{N}_i} \sigma^j_i Q^j_i\): the number of packets it effectively sends, where \(S^j_i\) indicates how many packets are sent to each neighbor \(j\).

\(^2\)Actually, a player knows the moves chosen by all the players of the game, but since we model nodes deciding only by means of local (i.e. related to the neighborhood) knowledge, we define the payoff as influenced only by neighbors moves.
5.2. COOPERATION AS A STRATEGIC GAME

- \( F_i \): the number of packets it accepted to forward during the following slot (i.e., \( A_i \) for next slot). It is defined as

\[
F_i = \min \left( \sum_{j \in \mathcal{N}_i} \phi_i^j(S_j^i + A_j^i), \Lambda_i(L_i - S_i) \right).
\]

In simpler terms, the total number of forwarded packets it is either the number of packets decided by means of \( \phi_i^j \) values, or, if the limit fixed with \( \Lambda_i \) is exceeded, exactly a fraction \( \Lambda_i \) of all the packets to be sent. It is also possible to compute how many packets are sent to any neighbors, as follows:

\[
F_i^j = \frac{F_i}{\sum_{j \in \mathcal{N}_i} \phi_i^j(S_j^i + A_j^i)} \phi_i^j(S_j^i + A_j^i)
\]

Note that it holds \( F_i = \sum_{j \in \mathcal{N}_i} F_i^j \), as expected.

- \( \forall i \in \mathcal{N}_i, \pi_i^j = \frac{F_i^j}{S_i^j + A_i^j} \), i.e. the percentage of packets every neighbor accepted to forward for \( i \).

We denote \( S \) and \( F \) the Cartesian product of the all the \( S_i \)s and \( F_i \)s, i.e. a snapshot of the global state (and of the global move, since from the values of \( S_i \) and \( F_i \) it is possible to find which move chose every player).

We define two metrics used to characterize the user payoff:

- \( \eta_i(S, F) \) (normalized spent energy), defined by the number of packets that node \( i \) sent and forwarded during the slot over the maximum number of packets it could have sent/forwarded:

\[
\eta_i \triangleq \frac{S_i + F_i}{L_i}
\]

- \( \chi_i(S, F) \) (locally estimated cooperation), i.e. the ratio of packets successfully forwarded by the neighbors of node \( i \) over packets that, in the optimal case, should have been forwarded, projected to following slot:

\[
\chi_i \triangleq \frac{\sum_{j \in \mathcal{N}_i} \pi_i^j \cdot (S_i^j + \alpha F_i^j)}{Q_i^j + \alpha F_i^j}
\]

Note that both \( \eta \) and \( \chi \) are computed using \( F_i \) instead of \( A_i \); since \( A_i \) is the result of a decision taken during the previous slot, it does not influence the current satisfaction of users. On the other hand, nodes evaluate the consequences of accepting \( F_i \) packets at the current network conditions, making a guess on the treatment it will receive during following time slot.
The parameter $\alpha \geq 0$ models the weight that other nodes packets have in the utility of node $i$: $\alpha = 0$ implies no incidence of how forwarded traffic is forwarded again by neighbors, while a high value of $\alpha$ means that other nodes traffic is more important than self generated traffic. This can be used to model the effectiveness of a punishment system: if no cooperation enforcement is used at all, then $\alpha = 0$, while a good mechanism should approach $\alpha = 1$, i.e. make other nodes traffic as important as personal traffic. An approximation of $\alpha$ for real punishment mechanisms can be determined as the average number of packets blocked to a selfish node per packet the same node has discarded with no reason\(^3\).

Both $\eta(S,F)$ and $\chi(S,F)$ take values between 0 and 1.

User satisfaction deriving from global move $(S,F)$ can then be computed by means of these two metrics as

$$u_i(\chi(S,F), \eta(S,F)).$$

Without additional assumptions on the network model, it is not possible to make the function $u_i$ explicit. However, we can highlight some interesting features it must exhibit. It cannot be increasing in $\eta_i$ nor decreasing in $\chi_i$. This implies that

- If $\chi_i$ is fixed, a node receives more utility by spending less energy.
- If $\eta_i$ is fixed, a node receives more utility when more cooperation is observed.

For this reason, assuming that the function $u_i$ takes only finite values, it must have a minimum in the point $(1,0)$, and a maximum in the (theoretical) point $(0,1)$.

Our payoﬀ function is not in contrast with the ones used in [49] and in [61]. However, while in [49] the payoﬀ received by all cooperative nodes is the same, in our case it depends on how they are treated effectively. Authors of [61] assume that nodes are willing to use all their battery, as long as energetic constraint are met, while we feel that nodes are better described by a greedy behavior, willing to save as much power as possible. Even so, a payoﬀ that ignores spent energy does not violate our assumptions (but makes the analysis less interesting).

### 5.2.2 Equilibria study

We analyze the properties of the model, with particular interest in studying the Nash equilibria ([50]) of the game. These peculiar combinations of actions describe stable points in which nodes have no interest in unilaterally changing their behavior.

\(^3\)It is possible to deﬁne punishments that variate in time, thus not having an average number of blocked packets, but we concentrate on stationary mechanisms.
Let us define the NULL move: this is obtained by setting \( \sigma_i = 0 \) and either \( \Lambda_i = 0 \) or \( \forall j \in \mathcal{N}_i, \pi^j_i = 0 \) and models a non working network. Note that when setting \( \Lambda_i = 0 \), the values assigned to \( \phi^j_i \) do not change the payoffs, since in reality no packets will be forwarded. The same is true when setting \( \pi^j_i \) to 0 and \( \Lambda \) to any value. For this reason, the NULL move is in reality a set containing \( \mathcal{N}_i + 1 \) moves that have exactly the same effect.

We start with a negative result:

**Theorem 3.** All the players playing the NULL move is a Nash equilibrium.

**Proof.** Let us consider a generic node \( i \): if it sets \( \sigma_i \neq 0 \) but all its neighbors play \( \Lambda_j = 0 \), it will result in \( F_j^i = 0, \pi^j_i = 0 \) and \( \chi_i = 0 \).

Thus, reducing sent and forwarded packets by \( i \) will not decrease its utility (in all practical cases it will indeed increase it, with less power consumption), leading to \( \sigma_i = 0 \) and \( \Lambda_i = 0 \) as a best response. \( \square \)

In order to generally study other (possible) equilibria, more information is needed about utility functions.

Note that any reasonable strategy leads to a behavior of local insatiableness: nodes try to get as much cooperation as possible and to save as much energy as possible. This means that the function \( u_i \) will be in general strictly increasing in received cooperation (\( \chi(S, F) \)) and strictly decreasing in spent energy (\( \eta(S, F) \)).

It is then possible to prove the following

**Theorem 4.** In all the eventual equilibria, for each node \( i \) one of the following conditions holds:

- \( i \) plays the NULL move
- \( \forall j \in \mathcal{N}_i, \pi^j_i = 1 \)

**Proof.** Let us suppose both the previous conditions are false.

If node \( i \) sends or forwards some packets that need further forwarding but are not in turn forwarded by its neighbors, it could increase its utility by avoiding such transmissions: if the same move is played by its neighbors, cooperation would not decrease, while spent energy would be less. For the strict monotonicity of utility function, utility would grow. \( \square \)

Moreover, it is possible to prove the following facts about the quality of (eventual) cooperation enforcement mechanism:

**Corollary 2.** If \( 0 \leq \alpha < 1 \), then in all the eventual equilibria, for every node \( i \) at least one of the following conditions holds:

- \( \Lambda_i = 0 \)
- \( \sigma_i = 1 \).
Proof. Let us assume that none of the previous conditions hold at the equilibrium for node $i$, i.e. $\sigma_i > 0$, while it is not sending all its generated traffic ($\sigma_i < 1$).

For theorem 4, all the packets sent and forwarded by $i$ must be in turn forwarded by its neighbors. But then, lowering $\Lambda_i$ in order to forward a packet less (i.e. decrease $F_i$ by one) and raising $\sigma_i$ in order to send an own packet more (i.e. increment $S_i$ by one) would leave $\eta$ unchanged, while increasing $\chi$ (because $\alpha < 1$), leading to a greater utility.

Thus, the previous scenario could not be a Nash equilibrium.

Corollary 2 means that either a node does not forward any packet or at least it sends all the packets it generated: nodes prefer to send their packets.

On the other side, the following fact can be proved:

**Corollary 3.** If $\alpha > 1$, then in all the eventual equilibria, for every node $i$ at least one of the following conditions must hold:

- $\sigma_i = 0$
- $\Lambda_i = 1$ and $\forall j \in N_i, \phi^j_i = 1$.

*Proof.* The proof is very similar to the one of Corollary 2. The condition on $\phi^j_i$s is necessary because, being $\Lambda_i \neq 0$, the value of payoffs depends on them.

Finally, it is easy to prove a result very similar to the one found in [36]. Defining the communication graph as the graph whose nodes are the network nodes and whose directed edges connect nodes for which $F^j_i \neq 0$, it is possible to prove the following

**Theorem 5.** In all the eventual equilibria, the communication graph has no edges (i.e. no nodes communicate) or is composed only by cycles (i.e. starting a walk in the graph from node $i$, any path chosen will pass again for $i$).

*Proof.* Assuming that the communication graph has some edges, we prove by contradiction that it is not possible to have cycle free maximal paths.

Suppose that there exists a node $i$ which is receiving traffic but not sending, i.e. it has only incoming arcs in the connection graph. If there exist a non cyclic maximal path in the graph, there must exist such a node.

Then, nodes which are sending to $i$ would have a better payoff by not sending anything, since their traffic is not being forwarded, and the edges to $i$ would disappear from the graph.

If after such a change some other node have no outgoing edges, the procedure can be iterated, leading to a graph with no edges at all or in which all the nodes having at least one incoming edge also have at least one outgoing edge.

Since the number of nodes is finite, there must be a cycle.

In order to prove that the graph is composed just by cycles, a similar reasoning can be applied to eliminate edges departing from nodes with no incoming edges.
5.2.3 Discussion

The presented model describes communications in ad hoc networks as a game. Such a game is local: every player’s utility is conditioned only by its move and by its neighbors behavior. With this assumption, we modeled a network in which nodes’ decisions are triggered by local observation. In principle, the game could be extended in order to consider in the payoff the treatment received by all the nodes in the network that should forward traffic for a node.

We kept the utility function absolutely general, assuming only that the function must not decrease if received cooperation increases, and must not increase if spent energy increases.

The presence of an eventual cooperation enforcement mechanism is taken into account with an appropriate parameter (denoted as $\alpha$) that changes the weight other nodes traffic have in utility. When no cooperation enforcement protocol is used, $\alpha$ is null, since non personal traffic should not be influent on a node’s satisfaction. When a specific mechanism is used, the value of $\alpha$ can be determined by computing the average number of packets that are blocked to a selfish node per every packet it discarded (i.e. $\alpha$ is a measure on the severity of the punishment). Note that an incentive mechanism can not be described by such a parameter, unless an explicit monetary value is given to personal packets, in order to be compared with the value of packets to be forwarded. However we feel this impossible, since personal packets may have a subjective value which changes depending on time, network conditions and reason they are being sent.

Theorem 3 presents a negative result: no matter whether a mechanism is used or not, the equilibrium in which nothing works cannot be eliminated.

In fact, Theorem 4 completes the picture: it states that, at the equilibria points, if a node is using the network, it will stabilize its contribution (in term of sent and forwarded packets) to the quality of service it is experiencing, in order to avoid transmissions that would not be forwarded.

Corollaries 2 and 3 describe the effect of tuning the parameter $\alpha$, i.e. setting the intensity of punishments for selfish nodes.

If punishment is light, i.e. $\alpha < 1$, meaning that for every discarded packet a selfish node must pay with less than one stopped packet on average, it cannot be avoided to have a network composed by fully working parts (namely, with nodes sending all their traffic and being fully served by neighbors) and by sectors in which nothing works. This is because if a node sees that its packets are not being forwarded by neighbors (not necessarily for selfishness), then the node itself will slow down its sending rate discarding preferably packets it should forward, starting a chain reaction.

On the opposite side, a very strict mechanism which inflicts a punishment to selfish nodes more severe than what they caused to their neighbors (formally, $\alpha > 1$), leads to a similar scenario with different causes: if neighbors of a node are not able to forward all the traffic the node itself is forwarding for others, then it is better
for them to send less self generated traffic. But then, in presence of congestion, the network would be composed by parts in which “slave” nodes just forward without being able to participate, and other parts in which everything works well.

Among the two conditions, the second is surely less dramatic from a network point of view (capacity is fully used), but it is less fair from a node point of view.

Finally, the condition $\alpha = 1$ would not necessarily lead to better equilibria: it it just not possible to prove any general condition for this case. However, it is to be conjectured that with a perfect mechanism a node which decides to discard some packets because they would not arrive to destination for sure, would discard in equal fractions its own packets and packets to be forwarded, with a smoother network behavior.

5.3 Cooperation as a repeated game

In the previous section we described the behavior of nodes in an ad hoc network as a strategic game, namely a game to be played only once. Cooperation enforcement was modelled as a parameter that can influence the payoff of nodes depending on how forwarded traffic is re-forwarded. This allowed to prove interesting properties on how traffic is adapted to the received service, and on what can be obtained if the punishment is too strict or too light.

In particular, we showed that tuning a punishment system is very difficult, since it would need a perfect monitoring system and a global coordination in order to always punish selfish nodes, and not to exceed in the task.

Many works proposed in literature (e.g., [48] and [15]) present solutions that are meant to be approximations of a punishment system. Such mechanisms can be seen as a behavior that cooperative nodes must adopt in order to limit the damage caused by parasitism. If in [15] there is a coordination among small sets of nodes (in order to spread the bad reputation of parasite nodes), in other works observations and reputation management is completely local. If a node do not forward traffic for one of its neighbors, than only the damaged node could punish it. Clearly, punishments are in general less effective, since a parasite node could use nodes that are not using it as a forwarder, and deny its help to all the others. However, this class of solutions seems to work quite well in practice.

As for many human and animal behaviors, it is possible to model all this mechanisms as strategies in a repeated game (built up on a strategic game very similar to the presented in Section 5.2). In a strategic game, every player must choose a move only by analyzing the game rules and payoffs, trying to maximize her own payoff, no matter what other players choose. There is no place for experiments or generosity: if a bad move is chosen, the received payoff is not optimal, and there is no possibility to change this fact. When a game is repeated, players may experiment different moves, switching to the optimal behavior if other players penalize

\[4\]In game theory literature, players are denoted as female.
her. This is useful because in general the Nash equilibria are not the moves yielding the highest payoff, but are the ones that minimize the damage that others can cause (see the Prisoner Dilemma in Appendix B for an example). A player could then try to propose a risky (but more convenient) action, and if the other players accept it, the game would lead to a cumulative payoff much higher than a sequence of payoffs deriving from a Nash equilibrium. If, on the other hand, other players choose a competitive strategy, the loss of payoff in the first round can be not dramatic in the total game history.

Every player, in a repeated game, prepares a strategy: Formally, a strategy with memory $m$ is a function that takes as input a sequence of $m$ global moves, and returns a move to be played. It chooses the move that a player should chose if the last $m$ observed moves are the ones passed as input. The number of strategies is exponential in the memory size $m$ and in the number of actions available to players.

For example, in a simple two persons game with only two moves for both players (say, $A$ and $B$), all the possible strategies with memory 1 are:

- Play the same move that other player used in previous round.
- Play the opposite move that other player used in previous round.
- Always play $A$, no matter what other played did in previous round.
- Always play $B$, no matter what other played did in previous round.

A strategy is said to be reactive if the proposed move effectively depends on the past history (the first two strategies in the previous example), unreactive otherwise (the last two strategies in the example).

Given a combination of strategies, it is possible to compute which will be the payoff for all the players. For this reason, a strategy can be considered as a move in the repeated game, and the Nash equilibrium can be extended to repeated games considering combinations of strategies that are not enhanceable by unilaterally deviation.

It is often possible to enforce equilibria in repeated games that are not sequences of Nash equilibria: For this reason, in our case, cooperation may arise also as the result of a strategy that cooperative nodes adopt, at least in principle.

Unfortunately, we show now that a non cooperative behavior cannot be excluded by any mechanism, as long as nodes act locally (i.e. as long as common actions cannot be taken), since mobility mitigates the effect of punishments.

In order to describe a repeated game, two basic ingredients must be provided:

- the basic game, i.e. a strategic form game which is repeatedly played,
- the number of repetitions (possibly infinite) and how single round payoffs are combined to compute payoff after $r$ rounds. Furthermore, as opposite to previous model, here we define the strategy as the observation of past games,
and the action taken in the current game. Some example of that are given in 5.3.3.

5.3.1 The basic game

We make the same assumptions of Section 5.2 about what is known by nodes at the beginning of each slot. In particular, we assume that the traffic generated by every node is $Q_i$ at every round in order to keep the complexity of the model low.

Interaction during the single slot is then modelled as the following game:

Players: nodes of the network are mapped into players,

Actions: the same actions than in the game described in Section 5.2.1,

Payoff: as in Section 5.2.1, once a node discovers the values of $S_i$, $F_i$ and $\pi^j_i$, it receives a payoff $u_i$ which depends on normalized energy spent ($\eta_i(S, F)$, defined as in 5.2.1) and received cooperation $\chi'_i(S, F)$ defined as follows:

$$\chi'_i(S, F) = \frac{\sum_{j \in N_i} F_j}{S_i + A_i}.$$

Note that in this case the observed cooperation is the cooperation effectively received during the slot, not a guess on future.

We assume $u_i$ strictly increasing in $\chi'_i(S, F)$ and strictly decreasing in $\eta_i(S, F)$, i.e. a locally insatiable behavior.

It is possible to prove the following:

Theorem 6. In all the pure Nash equilibria of the game, every node $i$ plays the NULL move.

Proof. It is easy to check that if a node is forwarding something for at least one of its neighbors (i.e. $A_i > 0$), it would raise its utility by simply decreasing $F_i$: the $\chi$ component of its utility would not change, while the $\eta$ component would decrease, increasing the whole payoff. This implies that all the nodes that receive forwarding requests by their neighbors, would set $\Lambda_i = 0$ or $\phi_i^j = 0$ for all the requesting neighbors.

A node whose neighbor are not forwarding for it would better stop sending, since the $\chi$ component of its payoff would not change (it would be null in every case), but the spent energy ($\eta$) would decrease, as in the previous case. Thus all the nodes set $\sigma_i = 0$. 

As previously pointed out, the Nash equilibria of a repeated game can be, in general, different than sequences of the basic game equilibria. For this reason, a game having as unique equilibria a non working network is not necessarily a bad
model. It just means that parasitism (or better, a nihilistic behavior) can easily occur if nodes are put together for a very limited time, knowing in advance that their actions will not have any effect in the future. If nodes can interact for a larger time, a different behavior is to be expected.

5.3.2 Repetition

In our model, every player plays against its neighbors in the represented network, since its payoff can be influenced just by their behavior. The game is played repeatedly until the neighborhood changes, at which point a new game must be started (because it formally changes the way the payoff of some nodes are computed).

Since it is not known in advance when a topology change is to be expected, it is not possible to model the network as a finitely repeated game. However, this is a positive fact since equilibria of finitely repeated games are sequence of equilibria of the game itself (§52). In our case, a non working network would be the only possible scenario, which would imply a bad model.

On the other hand, every player in our case knows that the game is not infinite, but it is not know when it is going to stop. For this reason, among the three variants of infinitely repeated games (§52, Chapter 8), only games with a discount factor are suitable, since they model the uncertainty about the prosecution of the game.

If, when playing for the \( r \)th time, every player knows that it could be the last time with a probability \( q \), then the computation of the cumulative payoff (namely, the payoff after that the basic game has been played \( r - 1 \) times) is defined as:

\[
\sum_{t=1}^{r-1} q^t u_i(t),
\]

where \( u_i(t) \) is the instantaneous payoff received after round \( t \). The uncertainty about the future makes current outcome more valuable than the future one. In fact, since the game could stop in every moment, it is better to perform well during early rounds than to have to wait an uncertain future to start receiving a good payoff.

In our model, if \( q \) is the probability that a neighborhood changes, the payoff for player \( i \) after \( r \) time slots will be denoted as:

\[
\sum_{t=1}^{r} q^t u_i(\eta(S_i(t), F_i(t))); \chi(S_i(t), F_i(t))),
\]

where \( S_i(t) \) and \( F_i(t) \) are the moves played by player \( i \) during slot \( t \).

The probability \( q \) is very important, since its value induces the set of equilibria that can be enforced. A very low value implies that the game is going to be repeated very unlikely, and for this reason only Nash equilibria of the basic game are possible. On the other hand, a value near to 1 indicates that the average number of times that the game will be repeated is quite large. Then, almost any non masochist
strategy can be enforced, if all the players agree on it. Basically, all the players play the enforced sequence of moves until one (or more) of them do not deviate. When such an event occurs, all the players switch to a “punishment” state, choosing the worst possible move against the deviating player, until the benefit she obtained with her deviation is compensated by the loss of utility experienced. Then, the enforced strategy is played again. This is possible only if $q$ is high enough to allow a satisfactory punishment to be carried out in a finite number of rounds.

5.3.3 Discussion

Searching Nash equilibria deriving from combinations of strategies can be a very frustrating task. In fact, there is generally a huge amount of equilibria even for very simple games. The repeated game of cooperation is moreover quite complex, as it can be understood by analyzing the following strategies with memory 1 (which are just a few of the possible strategies):

- unreactive cooperation of level $\chi$: for any history play $\Lambda_i = \chi$ and $\phi_i^j = 1$, and set $\sigma_i$ to some value,

- pure selfishness: for any history set $\Lambda_i = 0$ ($\sigma_i$ can be set to some fixed value, leading to an unreactive strategy, or may adapt to the past history),

- silence: for any history, play the NULL move,

- tit for tat on forwarding: set $\Lambda_i = 1$ and for every neighbor $j$ set $\phi_i^j(t) = \phi_i^j(t - 1)$, where $\phi_i^j(t)$ is the percentage of packages that node $i$ forwarded for $j$ during round $t$. Again, $\sigma_i$ may be set to any value.

However, some interesting general properties can still be found:

**Theorem 7 (Unreactiveness).** If only unreactive strategies are used, then the unique equilibria are sequences of the NULL move.

**Proof.** If all the neighbors of node $i$ use unreactive strategies, they will not adapt their behavior to how $i$ acted in the past. Thus $i$ would better stop forwarding, since this behavior is more convenient from an energetic point of view. Since this reasoning is valid also for neighbors of $i$, no traffic will be forwarded, and $i$ would increase its own payoff by stop sending, since less energy is spent.

This theorem was also proved in [36], but with a graph theoretical model and a more complex proof.

In general, repetitions of basic game Nash equilibria are still equilibria, but in the case of infinitely repeated games it is possible to enforce many sequences of moves as an equilibrium.

Informally, the basic idea is that all the players agree in a sequence of actions to be repeated endlessly. This sequence of action may be socially desirable, i.e.
maximizing the global utility of the system (for example the sum of all the payoffs), but often it is not locally optimal: there may exist actions that, taken by a single player would guarantee to this a higher payoff.

If a player deviates from the initially agreed behavior to take advantage of this condition, all the other players start a punishment. This consists in playing the action that minimizes the deviating node utility for a number of rounds sufficient to cancel the benefit obtained by deviating.

Not all the possible actions sequences are enforceable (for example the punishment could be too severe also for punishers), but the Nash folk theorems ([52]) define the classes of equilibria that can be enforced for any game.

Unfortunately, repeated games with discount criterion have much less enforceable equilibria than other models. This is due to the fact that the severity of the punishment is mitigated by the discount factor itself, and for low $q$ even an infinite lasting punishment could not be enough. Thus, if monitoring and punishment must applied only by neighbors, mobility makes the design of a punishment mechanism feasible only when nodes are almost static, i.e., when $q$ is very high. Otherwise, it could always be convenient to cooperate only partially, and to wait a neighborhood change in order to start with a fresh reputation.

The scenario would be extremely different if a deviation was observable by all the nodes in the network (or at least by all 2 or 3 hops distant). As observed in [48], however, this would imply a concrete security risk, since malicious node could accuse innocent nodes of parasitism. The issue is addressed in [16], where it is solved in the case of very large networks.

The impossibility to apply a punishment scheme in every network is partially mitigated by the results described in [4]: there exists very simple strategies (like TIT FOR TAT) that are not equilibria, but perform very well against many other strategies (both cooperative and uncooperative).

## 5.4 Monitoring and energy

All the punishment mechanisms proposed in literature are based on a reputation system which assigns a reliability index to every node in the network. If a node reaches a reputation lower than some threshold, then the node is assumed to be selfish and consequently punished (by means of temporal or permanent exclusion from the network). Reputation of a node is updated every time the node itself is observed while performing some action (e.g. forwarding a packet, replying to a route request), or every time a node is not observed acting as it is supposed to do (e.g. re-broadcasting a route request or reply).

Reputation systems are implemented at every node, and all the proposed systems can be divided in two classes:

- systems based on the observation of traffic flowing through the node,
systems based on the promiscuous listening of the shared medium.

Nodes adopting solutions in the former class try to understand how other nodes are behaving by analyzing just traffic that flows through them following the normal network dynamics. For example, the reception of a tcp ack (if forging is not possible) signals that all the nodes in the route to the destination have forwarded the corresponding packet. The reception of a forwarded packet (to forward again or to deliver to some application) is possible only when all the packets in the route from the source have worked well.

The latter class contains more aggressive solutions, consisting in the analysis of every packet that can be heard by setting the network interface in promiscuous mode. Clearly, solutions in this class are able to analyze much more events, if traffic is not extremely low, and allow a faster detection of selfish nodes. However, their application is very limited, since keeping the radio in promiscuous mode have an energetic cost. In fact, nodes must keep their radio always (or for a very long time) on.

When using energy saving protocols based on alternation of sleep and activity, the effectiveness of promiscuous listening have to be tested.

Both solutions are error prone and must be enhanced with a good security layer that prevent other nodes to modify the content of packets not generated by them, adding more complexity to the network.

For this reason, an incentive mechanism could be more suitable to increase cooperation in ad hoc networks.

5.5 Conclusions

We proposed two general models based on game theory to study the cooperation of nodes in medium-large wireless ad hoc networks. We shown that the behavior of nodes, when they are not under the same control authority, is in essence selfish, meaning that (as expected) they tend to send their traffic before than thinking to others.

With every model we were able to analyze different properties of the cooperation in ad hoc networks that may be used when designing a cooperation enforcement mechanism.

The first model is based on strategic games, and more focused on describe the effect of a punishment system on the behavior of nodes. It was shown that bad tuning of punishment can lead to undesired equilibria, and, even with perfect observations, a fair cooperation is difficult to enforce.

The second model, based on repeated games, is more useful when trying to understand when a punishment mechanism can successfully dissuade all the selfish nodes, and it was shown that a local mechanism (i.e. where only neighbors punish a selfish node) can be eluded if mobility is high, while global mechanisms need ad-
ditional security mechanisms that block malicious nodes from accusing cooperating nodes.

This consideration, paired with the analysis of selfishness presented in Chapter 3, suggest that very simple reactive strategies are probably much more effective in the mitigation of selfishness.

In order to completely eliminate selfishness, an incentive mechanism is probably more suitable, since it would allow an easier distributed implementation, and would be easier to tune against unfair conditions.
Chapter 6

Cooperation as a way to save energy

As noted in Chapter 2, the dichotomy between punishment for selfish nodes and rewards for cooperative ones has not yet produced a winning solution to the parasitism in ad hoc networks.

In fact, rewards intended as monetary payment (either in virtual or real currencies) to nodes that forward are practical just in a limited number of scenarios (e.g. next generation multi-hop cellular networks [45], which count with a credit management infrastructure), or adopting special, tamper-proof hardware. On the other hand, punishment mechanisms are more practical, and in fact some approximation of them (e.g. CORE [48] and CONFIDANT [15]) have been implemented, at least in widely used simulators. The main problem of these solutions is that they are quite costly, in terms of energy used to monitor, and they do not guarantee success (even if they seem to behave generally well, by simulative results).

In this Chapter we propose a novel rewarding mechanism, which allows node that seem to cooperate to enter in a power save mode during which their packets will be buffered by active nodes and transmitted to them at scheduled times. Node which seem to not cooperate will not have their packets buffered, forcing them to stay in idle mode and thus wasting more energy during their network usage, even if they do not spend to forward for others. In the remainder of the Chapter the terms energy and power will be used with almost the same meaning (in general paired with the term saving).

The focus is on the power save protocol, called PRES (Pseudo Random Energy Saving), while, for the development of the whole cooperation stimulation mechanism, a good detection mechanism not based on promiscuous listening must still be designed.

At our best knowledge, this is the first proposal that conciliate power saving with cooperation stimulation, rewarding cooperation in energetic terms (which is what selfish nodes are trying to save) and not in money.

Before to introduce and analyze the PRES protocol, we present a survey of other
energy saving protocols based on nodes periodically entering a sleep time, explaining why a new protocol is needed in the case of cooperation enforcement.

6.1 Energy saving protocols for wireless mobile ad hoc networks.

Many ad hoc network are mainly composed by battery powered nodes, making the problem of energy saving extremely important. This is specially true in the sensor networks domain, whose nodes lifetime is, in general, limited by the battery.

As pointed out in Chapter 4, the radio is a significant source of energy consumption in any kind of wireless network. In the case of uncoordinated ad hoc networks the situation is particularly critical, since nodes should always keep the radio in idle mode, consuming just little less energy than in reception/transmission mode.

The radio must be turned off as often as possible, in order to save energy when no traffic must be managed. However, in absence of any kind of coordination, it is not possible to know when traffic is going to be delivered to a node, making impossible the creation of a sleep schedule with any delivery guarantee.

Many solutions to this coordination problem have been proposed in literature, witnessing the relevance of the topic. In this Section we present just the ones considered more influencing on the remainder of the Chapter, starting with a quick description of the 802.11 BSS and IBSS power saving techniques.

For the same reason, we focus on protocols based on the establishment of a sleep/awake schedule for all the nodes composing the network, not presenting other types of energy saving protocols (e.g., application level solutions, etc.)

Moreover, we quickly review a protocol proposed for identification networks (a very special class of wireless networks, not included in the ad hoc model), since it was the main inspiration for the proposal of the PRES protocol.

6.1.1 IEEE 802.11 Energy saving

The standard 802.11 defines two possible operating modes:

- BSS, which is based on an infrastructure composed by access points (all the nodes in the network are connected to an access point),

- IBSS, which is infrastructureless and more similar to an ad hoc network.

In BSS mode, access points are not energy constrained: they can stay constantly turned on and have a global knowledge of all the nodes in their cell. In particular, they can periodically synchronize nodes in their cell, and buffer packets directed to nodes in power save mode.

Synchronization is obtained by periodically broadcasting a beacon message. Such a message contains a synchronization time stamp and a traffic announcement (Traffic
6.1. ENERGY SAVING PROTOCOLS FOR WIRELESS MOBILE AD HOC NETWORKS

Indication Map, TIM). The TIM indicates which nodes have traffic buffered at the station. Those nodes send an ack message to the base station and stay awake until all the traffic is received. Nodes for which no traffic is buffered can return to sleep until the following beacon.

Also broadcast and multicast traffic is announced, and it is sent before than unicast traffic (to allow nodes with only broadcast traffic pending to go to sleep soon).

The infrastructureless version of 802.11 is a trivial extension of the BSS mechanism. The beacon interval is decided by the node starting the network, while the synchronization is distributed: at the beginning of every beacon interval, every node tries to send (with a random delay) a synchronization time stamp. If a node receives such a time stamp before to be able to transmit its own, it synchronizes to the received one, canceling the transmission.

The TIM is substituted by an ATIM (Ad hoc Traffic Indication Map), sent by every node to every other node for which it has buffered traffic. A node which did not send nor receive any ATIM can return to sleep (after the reception of eventual broadcast/multicast traffic).

In [21], the IBSS 802.11 power save mode is compared with 802.11 without power saving (and with a newly proposed protocol, called SPAN and presented later in this Section) by means of simulations, showing very little savings in energy (with a slight performances degradation).

6.1.2 Identification networks

An identification network is a wireless network composed by a few interconnected base stations and a large number of tiny devices called RFID (Radio Frequency IDentification) tags or RFIDs in short. A RFID is a very simple device, with extremely reduced computational capabilities, small size and a low power radio used to communicate with a base station. They are normally used to track objects (like items in stores), and should be inexpensive. Usually their battery is not rechargeable: once it is exhausted, the RFID is dead.

Typically, base stations periodically poll all the RFIDs, in order to discover where are they (or better, where are located the objects associated with RFIDs). Alternatively, RFIDs periodically broadcast a HELLO message, to inform the base station about their status and position.

Since the base stations are not energy constrained and are connected to all the RFIDs in a given cell, they can coordinate the RFIDs like access points in infrastructure 802.11 do with their associated nodes. In particular, time can be divided in slots, with exactly one communication being performed during every slot. If a node do not know in advance when it is going to be polled, or when other nodes are going to send their HELLO messages, it should stay awake most of the time in order to wait for the right time, consuming much energy.
A very simple pseudo-random access protocol that allows almost all the RFIDs to sleep in a given time slot, and the base stations to contact the ones which woke up in the same slot is presented in [23] and [24].

The idea is very simple: every RFID has a pseudo-random number generator initialized with a seed known to the base stations. For every slot, a number between 0 and 1 is randomly generated by every RFID. If the number is high enough, the RFID turns on its radio, otherwise it continues sleeping. Since the base stations can generate the same numbers, they will have an accurate map of active nodes.

6.1.3 PAMAS

The PAMAS protocol ([54]) is a very simple protocol, that allow nodes to turn off their radios while a neighbor is receiving a packet. The protocol shows energy savings of almost a 10% in simulations, but the protocol uses separate channels for signaling and transmitting data, making it unsuitable for many scenarios.

6.1.4 BECA and AFECA

A more complex and effective solution is presented in [65]: The BECA (Basic Energy Conservation Algorithm) protocol, and its extension AFECA (Adaptive Fidelity Energy Conserving Algorithm).

Nodes using BECA alternate between an active state and a power save state during which they sleep and periodically check for incoming traffic. When new traffic is received or generated, a node returns to the active state. There is no guarantee on network connectivity, but if an on demand routing is used, it is enough to set the sleep time as an integral multiple of the RREQ retry timeout. In this way, it is easy to find an upper bound on the number of RREQ to send in order to find a node. However, this solution implies a consistent network flooding every time that a route discovery is performed.

AFECA is an enhancement of BECA, in which the sleep time, for any node, is proportional to the number of its neighbors.

Simulations show a great reduction in energy consumed, but an increase in latency of the order of seconds.

6.1.5 GAF

Nodes geographic position information (when known) can also be used to save energy, as shown in the GAF protocol ([66]). The network is divided in virtual grids, such that all the nodes inside a virtual grid can communicate with all the nodes in adjacent grids. For the networking point of view, all the nodes inside a virtual grid are equivalent. Thus, if at least one node per grid is active, the network will be connected.
Simulative results show that the nodes (and network) lifetime greatly increases when using GAF as routing, with the energy saved increasing with nodes density (because on average more nodes will be in the same virtual grid).

### 6.1.6 S-DMAC

The GAF approach ([66]) does not allow great energy savings when many virtual grids contain only one node. In [6] it is argued that, depending on the ration between the transmission radius $r$ and the network area side $R$, the number of grids containing only one node can be greater than the 50% of the total grids, independent on the number of nodes $N \leq 300$.

For this reason, authors propose a different approach based on the DMAC clustering algorithm ([5]), in which nodes build and maintain a spanning tree, which is in general composed by less nodes than the number of virtual grids needed by GAF. Nodes in the backbone are chosen among those having higher residual energy, and it is guaranteed that all the nodes will be able to sleep when their energy becomes lower that the energy of one or more neighbors. Simulations show energy savings of the order of the $50 - 60\%$, with little overhead.

However, this approach is specially directed to sensor networks, in which nodes are position aware, no very mobile (although mobility is tolerated by S-DMAC) and programmed by a unique owner: In fact, the weight that allow a node to leave the backbone and sleep are declared by the node itself, and are assumed to be truthful. In a multi-domain ad hoc networks, nodes can be instructed to cheat about their energy level, in order to never be included in the backbone.

### 6.1.7 SPAN

The SPAN protocol ([21]) aims at finding and maintaining a connected dominating set in the network graph, namely a subset of connected nodes that cover all the network (a node is in the set or directly connected with a node in the set).

With such a backbone always active, the other nodes can sleep, periodically waking to exchange traffic. The election of coordinators is distributed, and based on broadcast communications: every node broadcasts its neighborhood state, and if a node has two (or more) neighbors that are not connected with any coordinator, it is eligible to be a coordinator.

SPAN does not rely on any specific MAC protocol, but authors optimized it to be used with IEEE 802.11, using the beacon coordination offered by the protocol. After the ATIM window, an advertised traffic window starts during which non-coordinator nodes exchange traffic. After this time, they can sleep, while coordinator nodes can use the remainder of the beacon period to communicate between them.

Simulative analysis shows that the energy saved is almost the 50% more than with IEEE 802.11.
6.1.8 An adaptive approach

In [70] it is presented an adaptive protocol very similar to BECA ([65]). Nodes alternate active and power save states, but authors optimize the approach by using the 802.11 facilities. In particular, nodes enter the listening state at the beginning of a new beacon period, reducing to 2 the number of RREQs eventually needed to find another node.

Simulations show that the energy consumption is greatly reduced, while throughput and latency are almost the same than with the normal 802.11 protocol.

6.1.9 A totally distributed approach

An interesting approach in which nodes synchronization comes almost for free is described and analyzed in [35].

Every node has its own beacon period with length $T$, and it is active from the beginning of the beacon for $(1/2 + \epsilon)T$ seconds, sleeping in the remainder of the beacon interval. It is proven that, for every node, during the first or the last $\epsilon$ seconds of its activity time, all the neighbor will be also active. For this reason, broadcast and multicast communications should be performed (and repeated) during those intervals (called ATIM0 and ATIM1 in the paper). When a node has some traffic to send to another node, it sends a traffic announcement during the ATIM windows, together with its current estimation of the phase of the other node. If this estimation is wrong, the neighbor sends a message containing its correct phase (relatively to the announcing node phase). In this way, nodes need to discover the difference in phase just of nodes for which they have traffic directed to. However, nodes will be able to sleep only less than half the time, in any traffic conditions.

6.2 Power save and cooperation stimulation: rationale

In a infrastructure-less network without any form of coordination, every node should stay in idle mode when expecting traffic (directed to it or to forward). The IBSS mode of 802.11 (see 6.1) does not offer great savings, and does not guarantee the correct working of a multihop network.

Even selfish nodes, if they do not know when traffic directed to them is arriving, should stay permanently turned on in order to receive it correctly. As shown in Section 6.1, many protocols to coordinate nodes in order to allow them to periodically turn off the radio have been proposed and analyzed in literature.

However, the basic assumption underlying these protocols is that every node must be allowed to participate, while we want a protocol to which just nodes considered well behaving can participate.
Moreover, the protocol should maintain basic network performances (in terms of nodes reachability, throughput and end to end delay), or at least offer a good trade-off between a slight performances degradation and energy saved by nodes.

The protocol should work with any physical and MAC layer, and should not require modifications of other network protocols.

We believe that in a hypothetic protocol stack, a power saving/cooperation enforcement mechanism should be placed between the MAC layer and the network layer, in order to intercept all the packets coming from the network and analyze them (eventually blocking some of them - e.g. RREP of selfish nodes), and all the traffic coming from the upper layers, that can be buffered until the node is not ready to send.

Moreover, it must guarantee energy savings greater than those possible by not forwarding traffic. The condition can be seen also as the limit of such an approach in dealing with parasitism: only in cases where the energy spent in forwarding is not excessive it is possible to reward with energy. Scenarios like the ones depicted in Chapter 4, in which the energy spent in forwarding overwhelms the energy spent in personal communications, needs a different approach. However, we feel that spending the greatest part of energy for non personal traffic is not a good strategy for multi-domain ad hoc networks.

Obviously, the mechanism could benefit from accessing information available at higher levels (or a discussion of issues caused by selfishness at different layers, see [27]).

Finally, the protocol should be distributed, avoiding costly system-wide synchronization and allowing nodes to schedule transmissions to other nodes in a simple way with as few communications as possible.

The major benefit of rewarding cooperative nodes with sleep time is that, in case of false positives (namely, cooperative nodes accused of selfishness), they will be still able to fully use the network, sending and receiving their traffic, but an increased energetic cost. We believe that this solution is less drastic than excluding supposedly selfish nodes large part of from network operations.

### 6.3 Protocol description

The PRES protocol is composed by a decentralized reputation system, that returns a reliability index for any node in the network, and a scheduler, that receives packets in input from the upper layers (with information on the next hop to follow) and decides when to send them. The scheduler also controls the status of the radio, turning it on and off.

Given the difficulty of the design, we do not provide any special reputation system. However, we suggest a few rules to increase or decrease nodes reputation only with traffic analysis. Any other system not based on promiscuous mode can be used.
In fact, almost all the proposed solutions are rather complex systems based on the promiscuous listening of the medium and on event detection. For our purpose, we need to provide a traffic analysis based only on locally delivered traffic that successfully discovers parasite nodes, at least in probability. We leave this part of the protocol as future works.

6.3.1 Reputation system

The idea behind a reputation system is to label every node composing the network with an index describing the quality of the behavior shown by the node itself. In CONFIDANT ([15]) such an index is maintained in a cooperative way: every node that believes that another node is selfish warns all its friend nodes, that can lower the reputation of the presumably selfish node. As observed in [48], this solution, even if very efficient, can open the door to denial of service attacks, if malicious nodes start blaming innocent ones. For this reason, in the same paper it is presented CORE, which is a totally distributed mechanism. Every node using CORE forms its belief on other nodes in the network just by personally observing (if possible) their behavior. As in the CONFIDANT approach, nodes keep the radio in promiscuous mode and basically analyze all the sniffed packets to discover if they are forwarded or not. This solution is slower in discovering selfish nodes, but it is safer because it is not based on the trust on other nodes.

CONFIDANT was updated in [16] in order to filter the warnings and consider a node selfish only after a high enough number of messages accusing it.

The analyzed reputation system can be not effective when energy saving is used: promiscuous listening during fixed time windows (i.e. only when radio is turned on) can be of little help, since in many cases only traffic directed to node itself could be received.

For this reason, we describe a few guidelines for the design of an alternative mechanism, just to make an example: this mechanism is not analyzed and it could be less performing than the ones already proposed in literature.

In principle, any mechanism receiving in input a node and returning a reliability index based on the node behavior as a forwarder can be used. We assume that the reliability is an integer number, with higher values suggesting higher reliability.

We denote \( r_i(j) \) the reliability index that node \( i \) gives to node \( j \).

If node \( i \) has never interacted with node \( j \), then \( r_i(j) = 0 \), which is a neutral value. As noted in [4], optimism when nothing is known about a peer is generally a good strategy, and for this reason nodes with null reliability index should be allowed to participate to the energy saving protocol.

Every time that node \( i \) observe a negative event caused by \( j \), then \( r_i(j) \) is decremented. Negative conditions depends on the traffic involved in communications, and sometimes it is not possible to observe them.

In particular, it can be observed:
6.3. PROTOCOL DESCRIPTION

- if $i$ is the source of a tcp connection and $j$ is the first forwarder, and an ack is not received (or a nack is received, if used) its reliability index can be decremented. Clearly, there must be a way to avoid the ack forging,

- if an on demand routing is used, every time that $i$ sends a $RREQ$, then all its neighbors should re-broadcast it (with the exception of the nodes $i$ received the same $RREQ$ from): if after a reasonable timeout the $RREQ$ is not heard from some neighbors, their reputations should be lowered (with probability inversely proportional to collisions heard during the sampling window),

- if the node is receiving real time traffic (e.g. multimedia streaming) and the rate is not as expected, then it can decrement the reputation of the last forwarder that is delivering the stream.

Note that, if a node has its reputation lowered, it does not imply that the node is selfish. It just implies that it is in one or more unreliable routes, in which something is not going as expected.

Reputation can be raised in one of the following cases:

- when a tcp source receives an ack it can raise the reputation of the first forwarder in the route to the destination and of all the other known nodes in the route,

- when it overhears a $RREQ$ and when a $RREP$ is received by a node which is not the destination.

Note that all the informations are locally collected and analyzed, and the radio does not need to be put in promiscuous mode. However, other security mechanism are needed, like messages encryption to avoid ack forging.

If a node reputation falls down a REPUTATION_WARN threshold, the node is excluded by the power saving mechanism, but it is still used in routes formation (raising its reputation only in case of acks), while if the reputations becomes less than a REPUTATION_ALERT threshold, the node is temporarily excluded by the routes. In both cases, the node recognized as not well behaving is warned with a special message, in order to inform it that its traffic will be sent at random times and not only when it is awake.

In this way, a node will be still able to communicate, even if in general it will need to stay in a idle mode because other nodes are not buffering for it. It can still raise its reputation by actively participating to forwarding (in case of not very low reputation) or to routing and other eventual monitored protocols (in case of extremely low reputation).

6.3.2 The scheduler

We describe how the scheduler turn on and off the radio for each node independently, and how nodes synchronize in order to communicate.
We present two flavors of the protocol: one in which senders passively discover neighbors sleep/wake schedules by keeping their radio turned on for a bounded time, and one in which senders actively query neighbors in order to discover their schedule.

**Sleep and activity**

Every node in the network alternates active and sleep state. In active state, the radio is turned on and node can receive and transmit to active neighbors, while in sleep state typically no communication can be performed since normally the radio is turned off. During sleep state radio can be turned on in order to synchronize with one or more neighbors or to send packets.

Active state is kept for a fixed and common duration of $T_a$ seconds, while sleep state has a variable duration which is given by the invocation of a pseudo random number generator whose seed was initialized with the node MAC address. The pseudo random number generator is the same for all the nodes. Any distribution between $S_{\text{min}}$ and $S_{\text{max}}$ can be simulated, provided that is the same for the whole network.

We denote as $S_i(n)$ the duration of the sleep state entered for the $n^{th}$ time by node $i$. Sleep duration is cyclic: $\forall i, n, S_i(n) = S_i(n + p)$, where $p$ is the period of the duration. It means that the generator is re-seeded every $p$ generations (or a fixed size list of sleep durations is kept by every node). We also denote activity index the number of times a node has entered the sleep state at a given moment (modulo $p$). For example, is a node is entering its first (or $kp^{th}$ for some integer $k$) activity cycle, its activity index will be 0, if entering its second activity cycle, its activity index will be 1, and so on.

Finally, $P_i$ is the duration of a period for node $i$. It holds:

$$P_i = pT_a + \sum_{n=1}^{p} S_i(n). \quad (6.1)$$

Figure 6.1 shows the alternation of states in two nodes: note that the duration of sleep state is different, and that nodes “turn on” in different moments, thus not even the first activity time is in general superposed. In this example, we set $p = 2$, thus for $S_i(3) = S_i(1)$, $S_i(4) = S_i(2)$ (for $i = \{1, 2\}$).

We prefer a pseudo random approach to a deterministic one (namely, with fixed sleep duration equal for all the nodes) in order to avoid frequent superposition of activity time in neighbors. In fact, while at first it may seem convenient to have neighbors almost synchronized, in general this would greatly reduce the available capacity. For example, in Figure 6.2 it is shown a simple three nodes configuration. Suppose node $b$ always has traffic to send to $a$. If $a$ and $c$ have always the same wake/sleep schedule, with $a$ waking up always a little before than $c$, the latter node will almost always find the channel busy and would not be able to communicate. Similar cases can be easily found.
6.3. Protocol Description

On the other hand, with a pseudo random protocol, node $c$ would find the channel free with higher probability. Moreover, the superposition of active cycles would be fairly distributed in the whole network, not penalizing just a few unlucky nodes. Note that since the period length of every node is different, the repetition of a particular configuration (i.e. combination of nodes active and sleeping at the same time) in the future will be very far, since the period of the system is the minimum multiple common to all the periods.

### 6.3.3 Passive power saving protocol description

#### Discovering other nodes sleep schedule

In order to successfully communicate, nodes have to synchronize among themselves. In fact, in principle they have different sleep/activity cycles, and do not know other nodes scheduling. For this reason, every node with traffic to be sent, has to discover the state schedule of all the neighbors interested by the transmission.

In the passive protocol, every $k$ times a node enters an active state, it broadcasts a SYNC message. The message contains the address of the sender and its activity index modulo the period $p$.

If a node wants to synchronize with a single neighbor whose sleep schedule is unknown, it has to keep the radio turned on also during its sleep state, and wait to
receive a SYNC message from the desired node. If after $k(T_a + S_{max})$ seconds the message is not received, then the node is not in the neighborhood or a collision has happened. Assuming low collision probability, $k(T_a + S_{max})$ (or a little multiple) can be used as timeout for the discovery operation.

At the reception of a SYNC message, a node can compute its neighbor’s schedule by computing the starting time of the current period, and then the duration of every sleep cycle composing the period itself.

The time elapsed since the beginning of the current period can be computed by initializing the pseudo random number generator with the address of the neighbor, computing a number of sleep state durations equal to the received activity index, and adding $T_a$ times the same activity index and fixing in order to consider the propagation delay suffered by the SYNC message.

Note that no time information was passed in the message: all the time information is relative to the node receiving the message. It is necessary to improve the time computation by considering propagation radio state switch delay, but for the sake of simplicity we consider them null in the remainder of the Chapter.

**Sending unicast traffic**

When a node has traffic to send to a given neighbor, it first enqueue it in a buffer for the outgoing traffic, and then checks whether the schedule of the desired neighbor is known or not. In the former case, it sets a timer for the next time the neighbor will turn on its radio, while in the latter case it applies the discovery process previously described.

As soon as a SYNC message is received, buffered packets for the sender can be sent, since the neighbor is just entering an active state.

In Figure 6.3 it is shown a simple example, in which node $N_1$ enters into discovery state after receiving a packet to be forwarded to node $N_2$, whose schedule is unknown. During the discovery phase its radio is never turned off, and at the reception of a SYNC message, buffered data is forwarded (and schedule is computed for future transmission, which will not need discovery). In the example, $k = 2$. 

![Figure 6.3: Passive discovery of a neighbor.](image-url)
6.3. PROTOCOL DESCRIPTION

Sending broadcast traffic

Broadcast communication needs a different approach. In fact, it may not exist a time at which all the neighbors of a node are simultaneously active, or it could be very far in the future. For this reason, broadcast receivers need to be warned of the incoming traffic in order to turn their radio on at the right time.

A node needing to perform a broadcast selects a broadcast time in the future (at least after \(k(T_a + S^{max})\) seconds), and enters a special discovery state, which lasts exactly for \(k(T_a + S^{max})\) seconds. During this time, it does not turn off its radio, and for every incoming SYNC message, it records the neighbor information (if not already known or outdated) and sends a special BROAD message to the sender. The message contains the time of the scheduled broadcast relative to the current time.

After receiving a BROAD message, a node schedules a special broadcast event after the time specified in the message (minus a constant accounting for propagation delay). At the broadcast time, nodes in sleep state will have to turn on their radio, and keep it on at least for \(T_a\) seconds.

Protocol properties

In absence of traffic, during a period node \(i\) keeps its radio on for \(pT_a\) seconds, and off for \(P^i - pT_a\) seconds, where \(P^i\) is the duration a whole period, as reported in 6.1. This amount depends on the average sleep time simulated with the pseudo random number generator.

The overhead of the protocol is quite high: a message must be sent after every \(k\) sleep intervals, even if no other traffic is actually being sent over the network. A small value of \(k\) (i.e. frequent SYNC messages) may be quite expensive, but it reduces the time neighbors have to keep radio on in the discovery phase. A high value of \(k\) is desirable when the network traffic is low and when nodes do not need to discover neighbors often.

The number of BROAD messages, needed to announce broadcast traffic, depends on the nodes density: one per neighbor is sent.

6.3.4 Active power saving protocol description

Discovering other nodes sleep schedule

In the active version of the protocol, nodes do not send any SYNC message, which are very costly especially in presence of low traffic. Senders are required to actively discover when a neighbor is awake, in order to query its phase.

The mechanism is based on the following

**Theorem 8.** Let us assume that \(S^{max} = kT_a\) for some integer \(k\).
Let $t$ be the time at which node $i$ needs to communicate with node $j$. Node $j$ will be awake at time $t + i(T_a - \epsilon)$, for at least one value of $i$ in the integer interval $\{0, \ldots, k + 1\}$ and $\epsilon < \frac{T_a}{k + 1}$.

**Proof.** Node $j$ must wake up at least once in the time interval $t, \ldots, t + S_{\text{max}} + \epsilon$, for a positive $\epsilon$, since it is larger than the maximum sleep time.

Let us suppose $j$ is not awake at time $t$ (otherwise the discovery would succeed at the first trial). Then, if $j$ wakes up between $t + iT_a - i\epsilon$ and $t + (i + 1)T_a - (i + 1)\epsilon$, it will surely be awake also at $t + (i + 1)T_a - (i + 1)\epsilon$, since the activity state last exactly $T_a > T_a - \epsilon$.

Since there must exist a time during $t, \ldots, t + (k + 1)(T_a - \epsilon) = t + S_{\text{max}} - \epsilon$ at which $j$ wakes up, it must exist at least an index $i$ for which the Theorem holds. \hfill $\square$

For this reason, a node looking for a neighbor have to send a special REQ message every $T_a - \epsilon$ seconds, up to $k + 1$ times. The REQ message only contains the address of the requested node (it may contain information about the phase of the querying node, in order to prevent eventual future queries).

A node, at the reception of a REQ message directed to itself, will reply with a SYNC message, which must contain the time elapsed since the last period beginning. In this way, the discovering node can easily compute the schedule of the desired neighbor in a similar way than in the passive version of the protocol.

In Figure 6.4 it is shown a simple example in which node $N_1$, after the reception of a packet directed to node $N_2$, starts a discovery process by sending a REQ message every $T_a - \epsilon$ seconds, until it receives a SYNC message containing the synchronization data. Note that, differently from the case depicted in Figure 6.4, node $N_1$ can turn off its radio after querying without an answer.

### Sending unicast traffic

The process of sending unicast packets is the same as in the passive discovery protocol: if a node knows the schedule of the needed neighbor, it just have to set an alarm. Otherwise, a discovery is started.
6.3. PROTOCOL DESCRIPTION

Sending broadcast traffic

In order to alert all the neighbors about a future broadcast communication, Theorem 8 can also be used: if a broadcast packet is sent every \( T_a - \epsilon \) seconds, for \( k+1 \) times, it will reach every neighbor at least once. There is no further need to synchronization, in this case.

Protocol properties

The overhead of the protocol can be less than in the passive discovery version. In particular, in absence of traffic, no additional packets are sent. The number of REQ packets depends on the number of discoveries. After a broadcast communication all the neighbors are known, and no discovery must be done for them.

The number of messages needed to complete a broadcast communication is constant, and it is \( k + 1 \).

The protocol is not capacity preserving: capacity in input is clearly constrained by the activity state duration, since just when a neighbor is active neighbors will send traffic to it. It would be good to extend the protocol in order to adapt the ration of active and sleep state duration to the traffic conditions. This extension is left as future works.

The node traversal time of a packet, namely the time a packet spends into the buffers of a node in average, in presence of very low traffic is dominated by the time a sender must wait for a neighbor to enter its active state. This time is \( S_{max} \) in the worst case, and \( S_{max} + S_{min} \) in the average case. Thus, a packet traversing \( h \) nodes will have an average delay caused by the protocol of \( h \frac{S_{max} + S_{min}}{2} \).

6.3.5 Excluding selfish nodes

The protocol described allows all the nodes to periodically turn off their radio, being sure that packets will be buffered by neighbors during the sleep cycle.

It is possible to exclude nodes with a low reputation by not guaranteeing that packets will be delivered during their activity times. Such a decision must be explicitly communicated, since we want to guarantee that all nodes are able to use the networks. For excluded nodes, a random time to delivery can be computed when traffic arrives, in order to make the delivery unpredictable and force nodes to keep radio turned on if interested in receiving.

At a first sight, it may seem penalizing for cooperative nodes to have to wake up at random times to keep parasites awake. However, even in absence of parasitisms, nodes have to interrupt their sleep cycles to send packets to their neighbors. The discovery of a parasite node can be treated as a change of schedule of that neighbor, thus not penalizing any node.

Clearly, the mechanisms can stimulate cooperation only if enough nodes detect a selfish node at almost the same time. This situation is similar to any punishment
system, which is effective only if enough nodes punish the selfish one.

In fact, a selfish node detected only by one neighbor could try to set up new paths not containing accusing nodes. In this way, it could continue its normal sleep/wake schedule, without caring about the only node not buffering for it.

In order to make the exclusion from the energy saving protocol more penalizing, it is advisable to introduce a soft warning system. Nodes that discovery a parasite can spread this information to their neighbors (eventually up to a distance of 2 or 3 hops, to mitigate the mobility effect). Upon the reception of a warning, a node decrease the reputation of the presumably selfish node to a value little higher than REPUTATION_WARN. More warnings about the same node do not cause higher punishments: every warning will reset the reputation to the chosen danger level. In this way, detected node will be under special surveillance in its neighborhood, but not yet excluded by the protocol by all the nodes. If the accusation was wrong, nothing would happen: The node would have its reputation raised again by its correct behavior.

Note that the reputation system previously described lower the reputation of not selfish nodes that are on paths containing selfish nodes: if a node is accused of selfishness, and the accused node is cooperating, two things may be possibly happening:

- One or more nodes in the paths used by the accusing node are not forwarding (for parasitism or congestion). For this reason, the reputation of all the neighbors on routes where also the accusing node is present is lowered to a value slightly higher than REPUTATION_WARN. Those neighbors are not excluded by the energy saving mechanism, until parasite behavior is not directly observed.

- The accusing node is wrong, either maliciously or erroneously. For this reason, also the accusing node has its reputation lowered to a value less than REPUTATION_WARN. In this case, the accusation does not need to be propagated, since it is the result of a previous action.

Note that a false accusation of a node do not leave to any propagation of the lie. Moreover, the malicious node will have to renounce the help of the accused node, if it wants to use the energy saving protocol, and it will be specially observed by all the neighbors. An accusation caused by some other node in the path will result in the temporary discarding of the link between the two nodes, permitting to the accused node to check whether the problem is caused by one of its neighbors or not, and deviating the traffic if alternative routes are present.

Note that the routing protocol may use the reputation in order to use paths passing from highly reputed nodes, leaving low reputation nodes recover their condition, if they are not parasites. This can have good side effects also when dealing with congestion or localized interferences.
The real effectiveness of this approach needs to be validated with an extensive simulative work. However, since the design of a good reputation mechanism is way too complex to be presented in this thesis, we leave also this part as future works, concentrating in validating the PRES protocol in its energy saving component.

6.4 Performances evaluation

We analyze the performances of the PRES protocol in its active version by means of simulations run in the monarch extensions to the ns2 simulator ([1]).

6.4.1 Implementation details

We implemented the PRES protocol by modifying the 802.11 mac protocol that can be found in the Monarch wireless extension to ns2 ([1]).

In reality the MAC protocol was only marginally modified, and the PRES protocol has an almost independent existence.

No queuing module was used between the network and the MAC layers, since the PRES protocol has its own queues manager.

For every node, a list of 15 sleep duration was generated as described in this chapter, and stored in an array. A random starting time between 0 and 3 seconds was also generated (to simulate nodes arriving in the network at slightly different moments).

In order to buffer packets and keep track of neighbors schedule, we defined the neighbor_descriptor data structure as follows:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Int</td>
<td>Neighbor’s MAC address</td>
</tr>
<tr>
<td>start_period</td>
<td>Int</td>
<td>Last starting time for neighbor</td>
</tr>
<tr>
<td>sleep_duration</td>
<td>Int[15]</td>
<td>Neighbor’s sleep state durations</td>
</tr>
<tr>
<td>waiting_packets</td>
<td>Packet queue (size 64)</td>
<td>Packets waiting to be sent</td>
</tr>
<tr>
<td>next_wake</td>
<td>Timer</td>
<td>Timer to synchronize with neighbor</td>
</tr>
<tr>
<td>next_REQ</td>
<td>Timer</td>
<td>Timer to send REQ messages</td>
</tr>
</tbody>
</table>

Upon the arrival of a packet for neighbor $i$ (including also the broadcast address as a neighbor), the corresponding neighbor_descriptor is searched. If it does not exists, a new entry is created and inserted into the neighbors array. If the neighbor schedule it is not known, the discovery phase is started by sending a REQ message and setting the timer in order to send the following REQ message after $T_a - \epsilon$ seconds. On the other hand, if the schedule is known, the next_wake timer is set if the neighbor is currently sleeping.

A few variables to keep track of the neighbors actually being served, and of the
number of broadcast being waited are necessary, in order to keep the node awake when some communication is being performed, even if the sleep timer elapses.

When a timer relative to a neighbor elapses, the first packet in the corresponding queue is sent as normally. After a successful transmission, if the queue is not empty, more packets can be sent provided that the neighbor is still awake at the actual packet sending time (i.e., after the defer or eventual backoff).

6.4.2 Simulation scenarios

The network area was set to a rectangle with sides 1500m and 300m, and the simulation time was 900 seconds. The number of nodes was set to 50.

Two different scenarios were simulated:

- static nodes: nodes are placed uniformly at random in the area and do not move,

- mobile nodes: nodes are placed uniformly at random in the area and move using a Random Waypoint model with speed chosen uniformly at random between 8 and 8.5 m/s (i.e. nearly the average speed of cars in normally congested a city) and pause time uniformly distributed between 0 and 60 seconds.

6.4.3 Simulation aim

The simulations we run are directed to evaluate the end to end average delay, the packet loss and the energy saving caused by the PRES protocol.

In fact, we are interested in finding a good tradeoff between energy saved and performances degradation. From the protocol description, it is possible to put every node to sleep for any desired percentage of time on average by suitably setting all the parameters. However, this causes a loss of network capacity inversely proportional to energy savings, that could make the network not usable. Thus, we look for configurations guaranteeing high energy savings (on the order of 30-50%), while not causing high packet loss or extremely high end to end delay, which are two metrics very important for many common network applications.

Given the enormous numbers of parameters that can be varied to tune the protocol, we started with a low traffic scenario in which 4 source/destination pairs are extracted uniformly at random, allowing a node to be source or destination in more than one connection. Every source starts a cbr agent at a time uniformly distributed between 0 and 10, sending 8 512 bytes packets per second to the selected destination for the duration of the whole simulation minus 2 seconds (in order to allow the last buffered packets to reach their destination before the end of the simulation).

The following aspects were considered:
6.4. PERFORMANCES EVALUATION

- impact of $T_a$: $T_a$ was varied from 10 to 200 ms, setting every time $S_{max} = T_a$ and $S_{min} = T_a - 5$ ms.
- impact of $S_{max}$: $T_a$ and $S_{min}$ were fixed, while $S_{max}$ was varied between $T_a$ and $10T_a$.
- impact of $S_{min}$: for fixed $T_a$ and $T_{max} = 2T_a$, $S_{min}$ is varied between 0 and $S_{max}$.

After individuating a good combination of the three basic parameters, we made things (relatively) more complicated, by fixing the number of packets per second to 8 and varying the number of active source/destination pairs from 1 to 10.

The routing algorithm used was AODV, and the buffer size per neighbor was set to 64 packets. The buffer was managed in the PRES module, so the normal DropTail queue normally used in ns2 was bypassed.

Every experiment was repeated 10 times, and for a given number of sources and number of packets per seconds we used always the same traffic pattern.

6.4.4 Tuning the basic parameters

As previously described, we start by simulating the effect of every basic parameter on end to end delay, goodput and spent energy.

The role of $T_a$

In Figures 6.5-6.7 are presented the effects of variation in activity time on the performances we are interested in.

The time to complete a send with free channel and no collision nor interferences is in the simulator almost 5 ms. Thus, a very short activity time, of the order of 10 ms causes high delays and low goodput since the capacity of the network is sensibly reduced even for very low traffic. It also causes more energy to be spent since we set sleep time equal to activity time in this experiment, reducing thus the time spent in sleep mode. For $T_a$ values of about 70 ms the delivery time is minimized and the goodput maximized, while the spent energy is around the 60 – 70% than in the case of no energy saving (because $T_a = S_{max}$).

However, the optimal time depends on the simulated traffic, and higher loads may cause higher times to be more effective, even if at an increased end to end delivery time.

The role of $S_{max}$

Figures 6.8-6.10 present network performances when the maximum sleeping time is variated between $T_a$ and $10T_a$, each time setting the minimum sleep time as the maximum one.
CHAPTER 6. COOPERATION AS A WAY TO SAVE ENERGY

Figure 6.5: End to end delay when the common activity time is varied. In all the cases, the maximum and minimum sleep time are set as equal to the activity time.

Figure 6.6: Goodput when the common activity time is varied. In all the cases, the maximum and minimum sleep time are set as equal to the activity time.
6.4. PERFORMANCES EVALUATION

Figure 6.7: Energy spent when the common activity time is varied. In all the cases, the maximum and minimum sleep time are set as equal to the activity time. Values are given as fraction of energy spent when no energy saving is used.

We focused on three activity times that gave good results with fixed sleep time, in particular 5 ms, 7 ms and 9 ms. The objective is to find a good tradeoff that allows more energy savings at not dramatic performances degradations. As shown in Figure 6.8, the increase in delivery time is almost linear in the number of cycles spent sleeping, as expected. Also goodput (Figure 6.9) decreases as expected, while spent energy can be lowered to very low values, when sleep time is very high (Figure 6.10).

We consider that for common network applications, nodes may spend sleeping at maximum twice the time they spend in active state, but an adaptive approach could guarantee much better performances.

The role of $S^{\text{min}}$

In Figures 6.11-6.13 are presented the simulation results caused by variation of the minimum sleep time. For a fixed $S^{\text{max}} = 2T_{\text{act}}$, increasing $S^{\text{min}}$ causes a capacity reduction since the average sleep time of nodes also increases.

Moreover, it causes sleep intervals to have a less variable length, possibly increasing the number of collisions. This is reflected by an increasing delivery time (Figure 6.11) and a decreasing goodput (Figure 6.12).

Low values of $S^{\text{min}}$, that allows high energy savings as shown in Figure 6.13, could cause unfairness on spent energy, with some nodes spending more than others just because of their MAC address.
Figure 6.8: End to end delay when the ratio between the activity time and the maximum sleep time is varied. In all the cases, the minimum sleep time is set as equal to the maximum one.

Figure 6.9: Goodput when the ratio between the activity time and the maximum sleep time is varied.
6.4. PERFORMANCES EVALUATION

Figure 6.10: Energy spent when the ratio between the activity time and the maximum sleep time is varied. In all the cases, the maximum and minimum sleep time are set as equal to the activity time. Values are given as fraction of energy spent when no energy saving is used.

Figure 6.11: End to end delay when the ratio between the maximum and minimum sleep time is varied.
Figure 6.12: Goodput when the ratio between the maximum and minimum sleep time is varied.

Figure 6.13: Energy spent when the ratio between the maximum and minimum sleep time is varied. Values are given as fraction of energy spent when no energy saving is used.
6.4. PERFORMANCES EVALUATION

Figure 6.14: Performances of the PRES protocol in static networks when the number of cbr connections increases. The number of packets per second sent in every connection was fixed to 4.

6.4.5 Varying the traffic

We fixed $T_a = 50ms$, $S^{MAX} = 2T_a = 100ms$ and $S_{min} = 50ms$, and investigated the network performances as the total traffic increases. Given that the curves presented in previous sections have a similar shape when mobility is null or high, we concentrate on high mobility scenario, that represent a very bad case for the application of the PRES protocol.

Varying the number of connections

We simulated networks with an increasing number of cbr connections, varying from 2 to 10, and performances when using the PRES protocol are compared with those
Figure 6.15: Performances of the PRES protocol in static networks when the number of packet for every cbr connection increases. The number of connections in the network was fixed to 4.

Performances degradation introduced by PRES get worse as the number of traffic sources increases, showing an amplification of the normal effect of the traffic increase. In particular, 10 sources cause a very high packets delay (more than 1 seconds, 6 times higher than in the full capacity network) and a low goodput (a little less than 0.75).

Average spent energy also increases with the traffic load, since nodes have to forward more packets and collisions are more frequent, causing many retransmissions and route requests (following the failed delivery of a packet due to the retransmission limit reached) are performed. However, energy consumption is always less than \( \frac{2}{3} \) than in the absence of energy saving.
6.5. CONCLUSIONS

Varying the rate of packet per connection

An experiment similar to the one presented in Section 6.4.5 have been repeated, fixing the number of sources/destination pairs to 4, and varying the number of packets per second sent on every connection from 2 to 16.

End to end delay (Figure 6.15(a)) and goodput (Figure sensibly 6.15(b)) decrease when more traffic is injected in the network. However, they show a curve very similar to the one recorded when no energy saving is used, and in fact energy consumed does not varies in a sensible way.

6.5 Conclusions

In this chapter a framework to allow cooperative nodes to save energy was presented. The basic idea is to implement a distributed energy saving protocol and to allow to participate to it just to nodes which are correctly operating. This is possible only if a good reputation system not based on promiscuous listening of the medium is adopted in the network. Since the design of such a mechanism have many problems, we did not present any solution for this component of the system, concentrating on the energy saving protocol.

We presented and analyzed the PRES protocol, which is totally distributed and suitable to be used in a cooperation enforcement mechanism, since nodes can be excluded and admitted by its usage very easily.

Simulation results show the correct working of the PRES protocol, and suggest that an adaptive version could offer a good compromise between network performances and spent energy.
Conclusions

In this thesis we offered an alternative view of the problem of cooperation stimulation in wireless ad hoc networks.

We provided a simple statistical model that, enriched with some general consideration about rational behavior (that also selfish nodes must adopt) we mitigated the view of the damages that can be caused in a network by the presence of selfish nodes.

However, selfishness is surely a problem for ad hoc networks that should be possibly eliminated. For this reason we investigated what can be done by a generic punishment mechanism, i.e. a protocol to exclude from the network nodes that do not forward packets. It was possible to prove that a good and fair punishment is very difficult (if not impossible) to implement, since mobility and limited resources could in make any punishment too light to be effective, or too strict to allow a fair network usage.

For this reason, the main message of this work is that selfishness must be made less dangerous by using, in every cooperative node, a reactive behavior able to deal with service interruption. This approach would not guarantee the eradication of the selfishness by any network, but it could at least result in usable network.

If a solution that guarantees the elimination of selfishness is needed, then it is probably more promising to use an incentive system, in which cooperative nodes receive something for their help. This kind of systems is in general more difficult to implement, but in this thesis we also proposed a concrete and original example in which nodes are paid with sleep time. Thus, in principle, energy spent to forward packets would be paid in energy saved when sleeping, making the incentive system easy to formally describe and to implement.

As it was to be expected, this thesis do not offer any final solution to the problem of selfishness in ad hoc networks. Such a problem trespass on many different fields, and it is intrinsic of any system in which no centralized control is applied.

Future works include the design and implementation of a reputation system not based on promiscuous listening, in order to apply the framework describe in Chapter 6, and a refinement of the model shown in Chapter 4, which can be of great help in studying the effect of selfishness even at higher levels.
Bibliography


Appendix A

A simulative estimation of $H_{N}^{\rho}(h)$ in static networks

We run a series of experiments in order to determine the nature of the probability function $H_{N}^{\rho}(h)$, namely the average hop count in a network with parameters $\rho$ and $N$ (see 3.1 for a resume of the notation used).

First, fixed $R$ and $r$ and $N$, 1000 networks were generated. For every network, a random source was picked, and its minimum distance in hops from all the other nodes was computed (with a breadth first search on the connection graph).

For every found distance $h$, we computed the total number of nodes found at $h$ hops from the randomly picked sources over the total number of nodes considered: this is the frequency a given hop count shown in our experiments, and it can be taken as the approximation of the probability two randomly chosen nodes are a minimum hop distance $h$ in a general network with the same parameters. Note that in this case the total number of experiment was $1000N$, and the result is not an average but a frequency.

The average path length over the $1000N$ hop counts was also computed.

The experiment was repeated for densities varying from 7 (i.e. a barely connected network for many choices of network size) to 50 (a dense network: every node has almost 50 neighbors on average).

We begin by explaining the notation $H_{N}^{\rho}(h)$. In Figure A.1(a) it is shown $H_{802}^{0.05}(h)$ for different values of $R$ and $r$ (but with fixed ratio $\frac{1}{20}$) and $\Delta = 7$. The curves are almost indistinguishable (the same happens for different values of $\Delta$), and we can conclude that $H_{N}^{\rho}$ does not change if the ratio of $r$ and $R$ is fixed.

On the other side, Figure A.1(b) shows $H_{N}^{\rho}(h)$ when $\Delta = 7$, but $\rho$ and $N$ are different. Obviously, the same density does not implies same distribution in the hop count.

In order to understand how the distribution is related to $\Delta$, we fixed $R = 1$, and studied the average of the collected data, namely:
APPENDIX A. A SIMULATIVE ESTIMATION OF $H_N^\rho(h)$ IN STATIC NETWORKS

![Graphs showing the influence of $\rho$, $N$ and $\Delta$ on $H_N^\rho(h)$.](image)

(a) Plot of the function $H_N^\rho(h)$ when $\rho = 0.05$, $N = 892$ and $\Delta = 7$.

(b) Plot of the function $H_N^\rho(h)$ when $\Delta = 7$ but $\rho$ and $N$ are varied.

Figure A.1: Influence of $\rho$, $N$ and $\Delta$ on $H_N^\rho(h)$.

$$E[H_N^\rho] = \sum_{h=0}^{\infty} h H_N^\rho(h).$$

It is then possible to study the behavior of the average as $\rho$ or $N$ increases. Given the nature of our experiments, we keep $\rho$ fixed and variate $N$ in order to have densities from 7 to 50. We will study the following functions:

$$\mathcal{H}_\rho(N \pi \rho^2) = E[H_N^\rho]$$

Note that, for convenience, we study the average as function of $\Delta$, but with fixed $\rho$: this is equivalent to study them as functions of $N$.

Figure A.2 shows the plot of $\mathcal{H}_\rho$ for different values of $\rho$. For every curve, the confidence interval of level $1 - \alpha = 0.95$ is also shown (for $\rho = 0.01$ and $\rho = 0.02$ the confidence interval is extremely narrow, given the huge number of samples examined).

In [43] it is conjectured that the average minimum distance among two nodes, in hops, is proportional to $\frac{1}{\Delta}$, i.e.

$$\mathcal{H}_\rho(\Delta) = a_\rho \frac{1}{\Delta} + b_\rho,$$

for some constants $a_\rho$ and $b_\rho$. However, nothing is said about the relation of constants with $\rho$.

Figure A.3(a) shows a plot of $\mathcal{H}_\rho$ against $\frac{1}{\Delta}$, together with the line resulting from linear regression, in the case $r = 0.05$. 
Figure A.2: Average hop count for different values of $r$ ($R$ fixed to 1).

Figure A.3: Linear regression for two hypotheses on the nature of $\mathcal{H}_\rho$.

It is possible to have a better fitting by making the assumption that the data is proportional to $\frac{1}{\Delta^{2.25}}$, i.e.

$$\mathcal{H}_\rho(\Delta) = a_\rho \frac{1}{\Delta^{1.25}} + b_\rho,$$

for some constants $a_\rho$ and $b_\rho$. Figure A.3(b) shows the linear regression for the $r = 0.05$ scenario, where it is possible to notice that the experimental values are closer to the theoretical ones.

As shown in Figures A.4(a) and A.4(b), the least square error (absolute and relative) for various values of $r$ is lower in the case of the second hypothesis. Note that in this case the error is always below the 2.5%.

In order to complete the picture, it is necessary to relate the constants $a_\rho$ and $b_\rho$ with $\rho$. Figure A.5(a) show the plot of the interpolated values of $a_\rho$ against $\rho$, and in Figure A.5(b) is plotted $a_\rho$ against $\frac{1}{\rho^{1.25}}$. 
It is clear that there is a linear relation between $a_{\rho}$ and $\frac{1}{\rho}$, specially for low values of $\rho$ (i.e. for $r \ll R$). In fact, linear interpolation gives $a_{\rho} \approx \frac{4.5}{\rho^{0.85}} - 1.3$.

The same methodology was applied in order to relate $b_{\rho}$ and $\rho$. In Figures A.6 are shown plots of $b_{\rho}$ against $\rho$ and $\frac{1}{\rho}$. Again, with linear interpolation it is possible to conclude that $b_{\rho} \approx \frac{0.55}{\rho} + \frac{1}{3}$.

Putting things together, it is possible to conclude that $\mathcal{H}_{\rho}(\Delta) \approx \frac{4.5}{\rho^{0.85} \Delta} + \frac{0.55}{\rho}$.
Figure A.5: Determining the relation between $a_\rho$ and $\rho$.

Figure A.6: Determining the relation between $b_\rho$ and $\rho$. 
APPENDIX A. A SIMULATIVE ESTIMATION OF $H^c_N(H)$ IN STATIC NETWORKS
Appendix B

Basic concepts of game theory.

We present here a few basic concepts of game theory that are useful to better understand Chapters 2 and 5. Obviously we do not pretend to be exhaustive, so the reader is really encouraged to get a deeper view in a book entirely focused on the subject (it is impossible for us to select just one or two of the many books and to suggest them).

When $n$ rational entities interact for some reason, they can be modeled as players in a game. A strategic game has its rules, in terms of moves every player can make, while every player has personal preferences about the results of the game. Formally, the basic blocks of a game\footnote{We present here a simplified version of strategic games, where user preferences are substituted with payoffs. While this is not generally equivalent to the general case, in many practical cases there is no difference.} are:

- a set $N$ of $n$ players,
- a set $A_i$ of moves for every player, defining a global action space
  \[ A^* = \times_{i=1}^n A_i, \]
- a payoff function $p_i : A^* \to \mathbb{Z}$ for every player, defining the global payoff:
  \[ p^*(a) = \times_{i=1}^n p_i(a). \]

Every player knows the rules of the game, and, for her rationality, acts in order to maximize her payoff.

Example 1. Probably the most classical problem is the so called Prisoner’s Dilemma (see [31] for a history). Two suspects of a heinous crime are caught after a minor infractions. They are kept into separate rooms, and each one has to autonomously decide whether to confess the big crime or not. If only one of them confesses, he will be freed and used as witness in the trial, while the other one will receive a sentence...
of ten years. If they both confess, they will spend 5 years in prison, while if neither does it, for the minor crime they will pay with “just” one year of reclusion.

This can be seen as game with two players, where each one has two moves: Confess and Don’t confess. The payoffs can be the number of years they are sentenced, but in negative (since it is a cost).

A very useful way to represent two players games is via payoff matrix:

\[
\begin{array}{c|cc}
& C & D \\
\hline
C & -7, -7 & 0, -10 \\
D & -10, 0 & -1, -1 \\
\end{array}
\]

In the rows (columns) are represented Player 1 (2) moves, and in position \((i, j)\) there are the payoffs when Player 1 chooses her \(i^{th}\) move and Player 2 plays her \(j^{th}\) move.

Probably the most important concept concerning games is the equilibrium: a move \(a^*\) is an equilibrium point if no player, with a unilateral deviation, can increase her payoff. In formal terms, \(a^* = (a_1^*, \ldots, a_n^*)\) is a pure equilibrium for the game \(G = (N, A^*, p^*)\), if (and only if):

\[
\forall i \in N, \forall a_i \in A_i, p_i(a^*) \geq p_i(a_i^*),
\]

where \(a_i^*\) is the same of \(a^*\) except that for Player \(i\), which plays \(a_i\) instead of \(a_i^*\).

Example 2. The prisoner’s dilemma has a unique pure equilibrium: the one in which both players confess!

The following game has no equilibrium:

\[
\begin{array}{c|cc}
& P & D \\
\hline
P & 1, -1 & -1, 1 \\
D & -1, 1 & 1, -1 \\
\end{array}
\]

Example 2 shows that the concept of pure equilibrium is too weak, and for this reason it has been generalized to mixed (Nash) equilibrium ([50]). Let \(G’\) be a game derived by \(G = (n, A^*, p^*)\) in the following way:

- the set of players is the same,
- the moves of player \(i\) are all the possible probability distributions over \(A_i\),
- the payoff functions are defined on lotteries over combinations of moves.

It is possible to prove that every game has at least a mixed equilibrium, even if the problem of finding the equilibria points has no efficient solution at this moment.

Many variants of the strategic games have been presented in literature: we only report here two basic forms.
In repeated games, a strategic game is repeatedly played, either for a finite or an infinite number of times. The moves of a single shot games are combined into strategies, i.e. action plans, that can be more or less complex. The payoff is a combination of single game payoffs, considering that past results are “heavier” than future ones, because there is always the possibility that a game interrupts, or that two players do not meet again (i.e. there is a discount factor on subsequent outcomes). The equilibrium is defined over strategies, and it is possible to prove the following facts:

**Proposition 1.** If the game is repeated a finite number of times, then the Nash equilibria are sequences of Nash equilibria of the constituent game.

**Proposition 2.** It is possible to have Nash equilibria in infinitely repeated games which are not sequences of Nash equilibria of the constituent game.

The former fact is the consequence of a “backwards induction” reasoning: if there is a last move that will be played, then it must be a Nash equilibrium of the constituent game. But then also the move before, and so on until the first one. The latter proposition is proven with the so called Nash Folk theorems: if all the players agree on a desired sequence of moves, then it is possible to enforce it by punishing deviating players for enough time. It is in fact sufficient to make profit derived from a deviation less that losses that follow this decision, and a rational player will avoid deviations.

In Bayesian games, every player has a secret (her type) which conditions her payoffs, and has a prior belief on the secrets of other players, i.e. a distribution for the type of every player. The equilibrium is now defined in terms of lotteries on personal beliefs: every player will chose her best response to the distribution of possible moves of others.