Institut Eurécom
Networking and Security Dept.
2229, route des Crètes
B.P. 193
06904 Sophia-Antipolis
FRANCE

Research Report RR-06-173

A survey of computational economics applied to computer networks

Pietro Michiardi

Tel : (+33) 4 93 00 81 45
Email : Pietro.Michiardi@eurecom.fr

[1]Institut Eurécom's research is partially supported by its industrial members: Swisscom, Thales, SFR, France Tééc om, Hitachi, STMicroelectronics, Bouygues Téécom, Sharp, Cisco Systems, BMW Group Research & Technology BMW Group Company.

Abstract

The perspective taken in this report is to provide background information on recent efforts toward the convergence of computer science (both theoretical, dealing with algorithm complexity, and applied, dealing with the implementation of algorithms) and micro-economics techniques. We term *computational economics* the science that explores the intersection of economics and computation. Similarly to [33], our intent is to offer the tools for systems' engineers to design algorithms capable of supporting the desired system-wide goals taking into account the presence of selfish elements. We take the stance that selfishness is an obstacle to system designers goals and survey techniques such as payments and digital reputation, to name a few, as a way to overcome this obstacle. We define a unified framework to describe computational economics problems applied to computer networks.

1 Introduction

The constantly increasing attention raised by communication systems stems from their ability to exploit existing network infrastructures (such as the Interenet) to build inter-connected collections of computers capable of offering services that allow the interaction between software agents or between individuals (e.g. users). Recent work in wireless communication systems further pushed the requirements for individuals interaction offering systems capable of autonomously organize and communicate eliminating the need to rely on a pre-deployed infrastructure.

A large part of research in computer-science is concerned with protocols and algorithms that allow the formation of inter-connected systems and that regulate the interaction among the parties involved in their execution. The parties participating in such mechanisms can be humans, but they can also be software agents that act in their owners interests, allowing to increase the number and complexity of these mechanisms even further.

From a computer science perspective, one unusual aspect of the resulting systems is that the designer has no direct control over one of the systems core components, namely the behavior of the agents. With the emergence of the Internet (both in its two declinations, wire-line and wireless) as the platform for computation, the implicit assumption that the elements participating in the system will act as instructed - neglecting the faulty or malicious ones - does not hold anymore. System components (for examples computers that build and use the Internet) belong to different individuals or organizations and will likely do what is most beneficial to their owners. These elements will pursue their own self-interest, rather than follow any prescribed behavior. This opportunistic behavior can prevent the realization of the systems potential benefits, which are determined by the system designers' goals. The only solution is to design systems relying on mechanisms able to guide the system operating point to a desirable one in spite of the strategic behavior of the system components. Strategic behavior calls for an integration effort that combines techniques from economics - specifically, from game theory (GT) and mechanism design (MD) - into computer science.

The perspective taken in this report is to provide background information on recent efforts toward the convergence of computer science (both theoretical, dealing with algorithm complexity, and applied, dealing with the implementation of algorithms) and micro-economics techniques. We term *computational economics* the science that explores the intersection of economics and computation.

Similarly to [33], our intent is to offer the tools for systems' engineers to design algorithms capable of supporting the desired system-wide goals taking into account the presence of selfish elements. We take the stance that selfishness is an obstacle to system designers goals and survey techniques such as payments and digital reputation, to name a few, as a way to overcome this obstacle. We define a unified framework to describe computational economics problems applied to computer networks.

2 Mechanism design: tradition and evolutions

Game theory has developed powerful tools for modeling and analyzing strategic decision making in systems with (multiple) autonomous actors. These tools, when tailored to computational settings, provide a foundation for the analysis of selfish behavior on computer systems and their operation in equilibrium. While game theory aims at explaining what happens when independent individuals act selfishly, mechanism design deals with how to design systems so that certain system wide properties (for example, efficiency, stability, and fairness) emerge in equilibrium from the interaction of individuals that hold private information about their preferences.

The field of mechanism design aims to study how privately known preferences of multiple individuals can be aggregated towards a ”social choice”. However, aggregating the preferences of self-interested players is complicated by the fact that they will misreport their preferences if this is to their benefit. MD allows to study how to aggregate preferences in such a way that the system wide properties defined by the mechanism designer are obtained in spite of such strategic behavior.

In the following we provide a formal definition of traditional MD theory using models introduced in [33]. The interested reader can refer to the seminal works [18, 34, 36] for an authoritative introduction to game theory and mechanism design. We refer to *traditional* mechanism design to emphasize the evolution that this field has witnessed in recent years. The convergence of traditional MD with theoretical computer science has brought up what Nisan in its seminal work with Ronen called *algorithmic* mechanism design (AMD). Shenker and Feigenbaum further pushed the ideas of AMD to address the problems introduced by distributed systems, giving birth to *distributed* algorithmic mechanism design (DAMD). Recent work by Conitzer [8–10] shifted the notion of complexity behind AMD to the complexity of the actual design process of a mechanism. [10] introduces a new class of problems that go under the name of *automated* mechanism design (Auto-MD) that define possibility results as well as algorithms for the automatic design of mechanisms that traditionally have been designed manually.

Before delving into what traditionally has been defined as mechanism design in the following we provide some background information on game theorertical notations and definitions that will be used in the remainder of the report.

2.1 Some Basic Definitions from Game Theory

A *game* can be defined by a tuple $\langle V, \{S_i\}, \{u_i\}\rangle$, where:

- $V = \{v_1, v_2, \ldots, v_n\}$ denotes the set of *players*.
- S_i denotes the set of *strategies* s_i available to player v_i.
- $u_i(s)$ denotes the *utility* (or *payoff*) of player v_i under the *outcome* s, where $s = \{s_1, s_2, \ldots, s_n\}$ is a collection of the strategies played by the individual

players in an instantiation of the game (each player playing one strategy). Instead of utility, one can define a *cost* $c_i(s)$, in which case the players strive for minimization instead of maximization.

Let $s_{-i} = \{s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n\}$ denote an i-residual outcome, i.e., an outcome that includes the strategies of all players but v_i.

Definition 1 *A strategy s_i is a* best response *to s_{-i} iff $u_i(s_{-i} + \{s_i\}) \geq u_i(s_{-i} + \{s'_i\}), \forall s'_i \in S_i : s'_i \neq s_i$.*

Definition 2 *An outcome s is a* pure Nash equilibrium *iff for all $v_i \in V$, $u_i(s_{-i} + \{s_i\}) \geq u_i(s_{-i} + \{s'_i\}), \forall s'_i \in S_i : s'_i \neq s_i$. A similar definition applies when considering costs $c_i(s)$, the only difference being in the direction of the inequality.*

In other words, an outcome is a pure Nash equilibrium if all nodes select best responses concurrently. Under such an outcome, no node can "deviate" unilaterally and increase its profit.

2.2 Traditional Mechanism Design

Consider a distributed system composed of n individuals (or agents). Each individual $i \in I$ is characterized by a *utility function* that defines the preference ordering an agent has over the possible outcomes of the system. The outcomes are determined by the mechanism and defined by the system designer.

Formally, for a set of possible outcomes O each player has a utility function $u_i : O \rightarrow \Re$, where $u_i \in U$. Each agent holds private information $\theta_i \in \Theta_i$ that influence its preferences over the outcome of the system (θ_i is also known as the *type* of individual i).

Summarizing, agent i with type θ_i has utility $u_i(\theta_i, o)$ for $o \in O$. In this definition we let outcomes define also (eventual) payments made by the mechanism.

Definition 3 ***Mechanism***[1] *: A mechanism is defined by tuple $M = (\Sigma, g)$ that takes into account a strategy space Σ (i.e., the set of possible actions adopted by the agents) and an outcome rule $g(\sigma) \in O$ (i.e., the algorithm that defines the system outcomes) for $\sigma = (\sigma_1, \sigma_2, ..., \sigma_n) \in \Sigma^{|n|}$. A strategy $\sigma_i(\theta_i) \in \Sigma$ defines the actions selected by agent i for all possible types θ_i. Hence, the mechanism is specified by the strategy space, the outcome rule and the individuals types.*

Traditional MD can be sketched using the representation in Figure 1. The elements involved in the execution of a mechanisms are the system components and a special entity called *the center*, which is traditionally considered as a trusted third party executing the mechanism.

[1]The notation used in this report follows the one defined by Parkes in [12]. Alternative notations are used in [17, 33], but they do not fully grasp the roles of the entities involved in a mechanism.

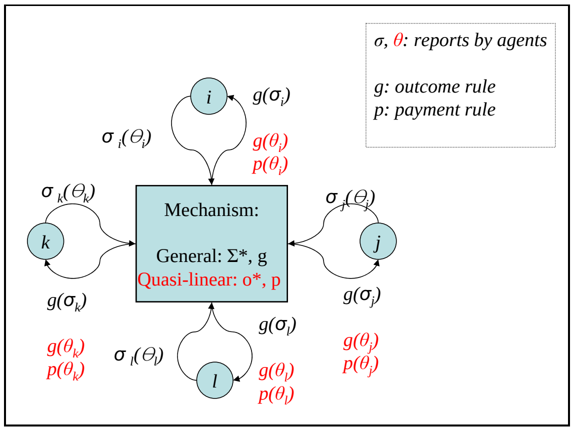

Figure 1: Sketch of a traditional Mechanism Design setting

In the previous definition we assume that agents are *rational* and that they select the best-response strategy to maximize their utility in equilibrium with other agents. In this case, no agents can benefit from any unilateral deviation from the equilibrium. Another important assumption we make is that the mechanism can implement and commit to the outcome rule. It is worth noticing that the set of possible outcomes O defines also eventual payments.

The general definition of a mechanism can be declined into an optimization problem, whereby the outcome rule is reduced to the optimization of an objective function, which represents the "goal" of the system designer.

The goal in traditional MD is to design a system in which rational agents interact in a way that leads to equilibriums with desired system wide properties. These properties are encapsulated in the *social choice function* (SCF)$f : \Theta \rightarrow O$, which defines the system outcomes for each possible set of agent types. For example, if the mechanism designer's goal is to achieve *efficiency* in the system by maximizing the total utility gained across all agents, then $f(\theta) = max_{o \in O} \sum u_i(\theta_i, o)$. Thus, the system designer is confronted with the problem of determining a set of outcomes $o \in O$ that maximizes the sum of the utility gained by every individuals taking part in the system, given that individuals will act rationally. The designers problem can be translated into an incentive allocation problem that guides agents' strategies in a way that allows the implementation of a particular SCF.

2.2.1 Quasi-linear mechanisms

An important class of problems related to traditional MD are those in which the utility function that characterizes individuals is *quasi-linear*. In this context the outcome space O can be factored into a set of system states O^* (or allocations) and a set of *payment* states $P \subseteq \Re^n$ that represent a vector of payoffs (or penalties). For a particular outcome $o = (o^*, p)$, agent i's utility function factors into $u_i(\theta_i, o) =$

$v_i(\theta_i, o^*) + p_i$, where $v_i : O^* \rightarrow \Re$ represents the agent's *valuation function* for each system state and p_i is the payment (if positive, taxation, charge or penalty if negative) made by the center.

For quasi-linear mechanism there is a class of mechanisms called Vickrey-Clarke-Groves (VCG). We dedicate section 2.2.5 to introduce VCG mechanisms as they play an important role in applied MD.

It is important to note that although the mechanism (O^*, P) is chosen by the system designer, the model that describes the interactions and the preferences of individuals is supposed to reflect reality. We argue that in addition to what was noted by [17], *i.e.,* the solution concepts that apply to the strategy selected by the agents should be tailored to a specific setting, also the system model (*i.e.,* the utility function) should reflect the context in which agents interact. We will come back on this issue in section 2.5.

2.2.2 Properties of traditional mechanisms

Many properties of a mechanism are stated in terms of the properties of the SCF function that the mechanism implements.

Definition 4 ***Pareto optimality :*** *a SCF function $f(\theta)$ is pareto optimal is for every $o' \neq f(\theta)$, and for all types $\theta = (\theta_1, ..., \theta_n)$,*

$$u_i(o', \theta_i) > u_i(o, \theta_i) \Rightarrow \exists j \neq i \in I \mid u_j(o', \theta_j) < u_j(o, \theta_j)$$

Informally, the implementation of an outcome is not Pareto-dominated by any other outcome, so no other outcomes make one agent better-off while making other agents worse-off.

Definition 5 ***Efficiency :*** *a SCF function $f(\theta) = (o^*(\theta), p(\theta))$ is efficient if for all agent types $\theta = (\theta_1, ..., \theta_n)$,*

$$\sum_{i=1}^{n} v_i(o^*, \theta_i) \geq \sum_{i=1}^{n} v_i(o', \theta_i), \forall o' \in O$$

The implementation of an *efficient* outcome maximizes the total value gained across the agents.

Definition 6 ***Budget Balance :*** *a SCF function $f(\theta) = (o^*(\theta), p(\theta))$ is budget-balanced, if for all $\theta = (\theta_1, ..., \theta_n)$,*

$$\sum_{i=1}^{n} p_i(\theta) = 0$$

Budget balance imposes that payments are not injected into or removed from the system. The budget balance property can be relaxed, so that the total transfer to the system is non-negative: the mechanism does not run a loss.

A SCF function $f(\theta) = (o^*(\theta), p(\theta))$ is *weakly budget-balanced*, if for all $\theta = (\theta_1, ..., \theta_n)$,

$$\sum_{i=1}^{n} p_i(\theta) \geq 0$$

The budget-balance property is fundamental in systems that must be **self-sustaining** and require no external entity to input money or central authority to collect payments. Yet, budget balance often conflicts with other desiderata, such as efficiency.

Definition 7 ***Individual Rationality :*** *Another important property of a mechanism is individual-rationality which allows an agent to decide to participate in a mechanism. Essentially, individual-rationality places constraints on the level of utility that an agent receives from participation.*

The most natural definition of individual-rationality (IR) is *interim* IR, which states that the utility to an agent, given prior beliefs about the preferences of other agents, is at least the utility gained from not participating to the mechanism. In a mechanism in which the agent's utility from participation must be at least equal to the maximum utility gained by non participating to the mechanism for all possible types of agents in the system, then individual-rationality is named *ex post* IR. Finally, *ex ante* IR states that the agent's utility for participation to the mechanism, averaged over all possible preferences, must be at least its utility without participating, also averaged over all possible preferences. In this case, an individual must choose to participate before it even knows its own preferences.

2.2.3 Equilibrium solution concepts

The game theory literature contains a very rich set of solution concepts that cannot be fully explored in this report. The designer's role is to prove the existence of an equilibrium of the system and to compute it. Hence, the feasibility of a mechanism greatly depends on the equilibrium concept used to find the solution (in GT terms) to the game the mechanism imposes to the agents. The game specification can be further enriched by specifying the context in which individuals interact; the system can be modeled by a repeated game or a single-shot game, for example, or by taking into account agents' collusion, agents' knowledge about other individuals' preferences, *etc*...

As a mechanism implements the SCF f in *equilibrium*, in the following we present the three most useful solution concepts, each successively requiring stronger assumptions about agents.

Definition 8 ***Dominant strategy equilibrium*** *: Each agent has a best-response strategy no matter what strategy the other agents select. Formally, we have*

$$\sigma_i^* = \arg\max_{\sigma_i} u_i(\theta_i, g(\sigma_i(\theta_i), \sigma_{-i}(\theta_{-i}))), \forall \sigma_{-i}, \theta_{-i}$$

A dominant-strategy equilibrium provides a robust solution concept because no information is required about individuals rationality or types distribution for an agent to select the strategy to be played.

Definition 9 ***Ex-post Nash equilibrium*** *: An ex post Nash equilibrium requires common knowledge about agents rationality but doesn't require any knowledge about type distributions: agents will not deviate from the selected strategy even once other agents types are known. Formally,*

$$\sigma_i^* = \arg\max_{\sigma_i} u_i(\theta_i, g(\sigma_i(\theta_i), \sigma_{-i}^*(\theta_{-i}))), \forall \theta_{-i}, \theta_i \in \Theta_i$$

where $\sigma_{-i}^*(\theta_{-i})$ denotes the equilibrium strategies played by other agents.

Definition 10 ***Bayesian Nash equilibrium*** *: The Bayesian-Nash equilibrium is the weakest solution concept adopted in MD. Every agent must hold both beliefs about other agents rationality and correct beliefs about the distribution on types of other agents. Formally,*

$$\sigma_i^* = \arg\max_{\sigma_i} E_{\theta_{-i}}[u_i(\theta_i, g(\sigma_i(\theta_i), \sigma_{-i}^*(\theta_{-i}))], \forall \theta_i \in \Theta_i$$

In a Bayesian Nash equilibrium each agent selects a best-response strategy to maximize its *expected utility* given its beliefs about the distribution over types, as long as the other agents also play an equilibrium strategy.

In [17], the authors further question the usefulness of traditional solution concepts and illustrate the possibility of exploiting *adaptive learning* techniques, used in the context of repeated games as introduced in [30]. We stress that game definitions and solution concepts must be tailored to the specific context in which the system designer operates. Game theory is a rich and evolving discipline that allows, as an additional example, to study the *evolution* of systems with techniques akin to genetic programming: recent advances in *autonomic systems* capable of adapting to changing environments suggest that solution concepts by which only the "fittest" strategy survives in a system is another research direction that mechanism designers are going to undertake.

In section 2.3 we explore the additional constraints that derive from the quest for realistic communication models. So far we intrinsically assumed individuals capable of fully understanding the protocols that govern their interactions and abide with them. We assumed cost-free, fault-less communications between agents and with the center. This assumption is not realistic in the specific context of computer networks. We assumed static agent population, as opposed to open systems in which a dynamic population might interact. Again, this is not realistic when modeling complex systems, such as peer-to-peer content distribution systems, wherein peer population evolves in *flash-crowds* bursts.

In reality, individuals involved in a mechanism and modeled by game theory have often constraints on their resources. MD has traditionally taken the conservative view that agents will always choose the action that is in their own best interest –the assumption of *perfect rationality*. Fundamental results in traditional MD theory that we discuss in the following section relies heavily on this assumption. However, this assumption may be overly conservative, in that agents may not always have the computational resources to find the action that is in their own best interest –that is, their *rationality is bounded*. Note that a realistic model of individuals does not always entail worse results. In their work, Conitzer *et al.* [11] showed that there can be significant benefits if the strict requirements imposed by perfect rationality are relaxed.

Moreover, game theory relies mostly on equilibrium notions wherein the optimal action for an individual to take depends on the actions of the other individuals, so that in general there is no clear *ex ante* optimal strategy. These equilibrium notions are known as solution concepts. Many solution concepts have been proposed in the literature, but the study of **how to compute** strategies according to these solution concepts has received only limited interest until recently. Knowing how to compute game-theoretic strategies has obvious applications to the design of software algorithms and agents. Furthermore discovering that some equilibria are easier to compute than others, may help resolve the well-known equilibrium selection problem: agents involved in a decision process are faced to the selection of which solution concept to implement, which implies that no action can be taken univocally. Considerations on the computational effort required to find a solution to a strategic situation have implications for mechanism design. If the solution concepts are too difficult to compute, then it is unreasonable to expect that agents will play according to them.

2.2.4 Incentive Compatibility and the Revelation Principle

Incentive compatibility and the revelation principle are important concepts in MD theory that apply to mechanisms in which the only action available to the agents is to report to the center their types θ_i. This kind of mechanism is often referred to as *direct mechanisms*.

Definition 11 ***Direct Revelation Mechanism*** *: A direct-revelation mechanism $M = ((\Theta_1, \Theta_2, ..., \Theta_n), g(\cdot))$ restricts the strategy set $\Sigma_i = \Theta_i, \forall i$, and has an outcome rule $g : \Theta_1, \Theta_2, ..., \Theta_n \rightarrow O$ which selects an outcome $g(\hat{\theta})$ based on reported preferences ($\hat{\theta} = (\hat{\theta_1}, ..., \hat{\theta_n})$.*

In a direct-revelation mechanism, the strategy of agent i is to report type$\hat{\theta}_i = \sigma_i(\theta_i)$, based on its *true* preferences θ_i.

As opposed to direct mechanism, the mechanism specified in Definition 3 is often referred to as *indirect mechanism*. In this case, the agents no longer directly reveal their types but instead choose strategies from the space Σ. The strategy selection is done selfishly in order to maximize the agents' utility. Note that the research community has focused mainly on direct mechanisms. The mechanism designer's goal is to provide incentives so that the best strategy for an agent is to tell the truth about their types, *i.e.,* $\hat{\theta}_i = \theta_i$ following the definition given in Definition 11.

Definition 12 ***Truth Revelation*** *: A strategy $\sigma_i \in \Sigma_i$ is truth-revealing is $\sigma_i(\theta_i) = \theta_i, \forall \theta_i \in \Theta_i$.*

Following what has been argued in [17], we question that direct mechanisms will play the same role in real-world implementations of MD theory as they played in the game theoretical literature. Revelation of private information, though guided by appropriate incentives provided by the system designer, can be unacceptable in the broader context of computer networks, e-commerce, e-voting systems to name a few, where **privacy** concerns can be deciding factors in favor or against a technology. In these contexts, the goal of MD is to allow agents' to compute a strategy that depends on private preferences without revealing anything that could disclose sensitive information. The cryptographic community provides a large number of security protocols that can be considered mechanisms themselves (in the game theoretical sense) such as *secure multi party function evaluation*, *zero-knowledge proof protocols*, *rational-exchange protocols*, *etc*, [17]. We argue that cryptography should be considered also from the perspective of a helper technology to guarantee privacy, data confidentiality, data integrity, *etc*, for applications that rely on traditional MD theory.

In an incentive-compatible (IC) mechanism the equilibrium strategy profile $\sigma^* = (\sigma_1^*, \sigma_2^*, ..., \sigma_n^*)$ has every agent reporting its true preferences to the mechanism.

Definition 13 ***Bayesian-Nash incentive-compatibility*** *: A direct-revelation mechanism M is Bayesian-Nash incentive-compatible if truth-revelation is a Bayesian-Nash equilibrium of the game induced by the mechanism.*

Definition 14 ***Strategy proofness*** *: A direct-revelation mechanism M is strategy-proof if it truth-revelation is a dominant-strategy equilibrium.*

Strategy-proofness is a very useful property, both game-theoretically and computationally. Dominant-strategy implementation is very robust to assumptions about agents, such as the information and rationality of agents. Computationally, an agent can compute its optimal strategy without modeling the preferences and strategies of other agents.

A simple equivalence exists between the outcome function $g(\hat{\theta})$ in a direct-revelation mechanism, which selects an outcome based on reported types $\hat{\theta}$ and the social choice function $f(\theta)$ implemented by the mechanism, i.e. computed in equilibrium.

Definition 15 ***Incentive-compatible implementation*** *: An incentive-compatible direct-revelation mechanism M implements the social choice function $f(\theta) = g(\theta)$, where $g(\theta)$ is the outcome rule of the mechanism. In an incentive-compatible mechanism the outcome rule is precisely the social choice function implemented by the mechanism.*

The revelation principle for dominant strategy implementation states that any social choice function than is implementable in dominant strategy is also implementable in a strategy-proof mechanism. In other words it is possible to restrict attention to truth-revealing direct-revelation mechanisms.

Theorem 1 ***Revelation Principle*** *: Suppose there exists a mechanism (direct or otherwise) M that implements the social-choice function $f(\cdot)$ in dominant strategies. Then $f(\cdot)$ is truthfully implementable in dominant strategies, i.e. in a strategy-proof mechanism. The interest reader can refer to [36] for the proof of the theorem.*

The revelation principle suggests that to identify which social choice functions SCF are implementable in dominant strategies, we need only identify those functions $f(\cdot)$ for which truth-revelation is a dominant strategy for agents in a direct-revelation mechanism with outcome rule $g(\cdot) = f(\cdot)$.

With the revelation principle in hand it is possible to prove impossibility results over the space of direct-revelation mechanisms, and construct possibility results over the space of direct-revelation mechanisms. However, the revelation principle ignores computational considerations and should not be taken as a statement that it is sufficient to consider only direct-revelation mechanisms in practical mechanism design. The revelation principle states what can be achieved, what cannot be achieved, but without stating the computational structure to achieve a particular set of properties. All equilibrium and outcome calculations are moved to the center, and we assume the agents can report their complete types, and that communications are cost-free.

In the following section we provide a practical example of a class of *efficient*, *strategy-proof* direct-revelation mechanisms, often called Grooves mechanisms [7, 20, 40], or VCG mechanisms.

2.2.5 Generalized VCG mechanisms

A class of direct-revelation, strategy-proof mechanisms, called VCG mechanisms, has been widely used to derive fundamental possibility results. VCG mechanisms have found many applications in networking (in particular, routing) problems. VCG mechanisms maximize the social welfare, that is, select the outcome that maximizes the total value across all agents.

Consider partitioning the outcome space into an allocation choice k and payments p. Outcome $o = (k, p)$ defines a choice $k \in K$ in the space of feasible choices K and payments $p = (p_1, \cdots, p_n)$ by (or to) agents. For instance, the choice set can describe all feasible resource allocations to agents. As is common in most auction theory [36], we restrict our attention to quasi-linear mechanisms whereby $u_i(\theta_i, o) = v_i(\theta_i, k) + p_i$, where $v_i(\theta_i, k)$ defines the value of allocation k to agent i given its type θ_i.

In the VCG mechanism the center receives claims $\hat{\theta}_i$ from agents about their valuations and calculates the allocation choice k^* that maximizes $\sum_{i=1}^{n} v_i(k^*, \hat{\theta}_i)$. Each agent makes payment $v_i(k, \hat{\theta}_i) - (V(N) - V(N\ i))$, where $V(N)$ is the total reported value of k^* and $V(N\ i)$ is the total reported value of the choice that would be implemented without agent i. The payment terms align an agents incentives with that of the mechanism and make truth-revelation a dominant strategy. In equilibrium, each agent receives as utility the marginal value that it contributes to the system. In other words, the VCG mechanism, requires an agent to pay the externality that its presence imposes on the other agents.

Unfortunately, general mechanisms such as the VCG mechanism invariably come with certain drawbacks. For example, the VCG mechanism is not budget-balanced: the agents' payments do not necessarily sum to 0, so that some of the payments must be collected by an outside party (or burned). If the budget must be balanced, it is in general impossible to create a truthful mechanism that always chooses the efficient outcome.

One important impossibility result, the Myerson-Satterthwaite theorem [25], shows that no efficient and balanced mechanism can exist in many simple settings. So, any indirect mechanism that implements an efficient allocation in equilibrium must also implement the payments defined in a VCG mechanism [36].

In general, if we choose to retain budget balance, we must accept some efficiency loss. Approaches to addressing the budget-balance problem include adjusting payments to get close to VCG payments but retain budget balance [37], and retaining truthfulness but explicitly clearing exchanges suboptimally to sacrifice some efficiency in return for budget balance [26].

2.3 Algorithmic Mechanism Design

Traditional MD theory provides tools that can be applied if a centralized approach and "ideal" models can be accepted. However real systems impose modifications to the way individuals compute, communicate and interact. For example, in a computer network setting, agents don't have the unbounded computational power that might be required to calculate their preferences for all possible outcomes or calculate equilibrium strategies. The mechanism infrastructure in a centralized mechanism might be unable to compute the outcome because the problem might be intractable, and communication between agents and the infrastructure might not necessarily be cost-free and could also be error-prone. Furthermore, real computer networks are dynamic, both because of physical characteristics and because of varying agent population. Nisan and Ronen [33], and Feigenbaum and Shenker [16] catalyzed the research aiming at studying the impact of computational complexity of traditional mechanism design.

Before exploring the definitions of complexity addressed in [15, 16, 33], we focus on *the origin* of computational complexity of traditional MD.

Much of classic mechanism design is driven by the revelation principle (see Theorem 1), which states informally that only direct-revelation mechanisms need to be studied since every mechanism can be translated into a direct mechanism. It is interesting to notice that direct mechanisms convert decentralized problems into centralized ones. In a direct-revelation mechanism agents are restricted to sending a single message to the mechanism, where that message makes a claim about the preferences of the agent over possible outcomes. The transformation assumed in the revelation principle from indirect mechanisms (*e.g.* an iterative auction) to direct-revelation mechanisms (*e.g.* a sealed-bid auction) assumes unlimited computational resources, both for agents in submitting valuation functions, and for the mechanism in computing the outcome of a mechanism. However, it can become impractical for an agent to compute and communicate its complete preferences to the mechanism, and for the mechanism to compute a solution to the centralized optimization problem.

We characterize the computation in a mechanism within two distinct levels:

- At the agent level:
  - **Valuation complexity**, that defines the computation required to select the best-response strategy in general or to provide preference information within a quasi-linear mechanism;
  - **Strategic complexity**, that defines the "modeling capabilities", *i.e.,* the knowledge about other agents involved in the system, available to an agent required to compute the best-response strategy;
- At the infrastructure level:
  - **Outcome rule-determination complexity**, that defines the computation required for the mechanism infrastructure to compute an outcome rule given the information provided by agents;
  - **Communication complexity**, that defines the amount of communication (in number of messages) between agents and the mechanism required to compute an outcome.

Focusing on the infrastructure level, the central question is to determine which problems are easy and which are hard in relevant computational models. Informally, an AMD problem can be considered ”easy” if it can be solved in a manner that is **both** incentive-compatible and computationally tractable. In this work we focus on the role of the interplay between incentive compatibility and computational complexity on hardness of a mechanism. Thus, a more useful distinction is made by defining an AMD problem to be *canonically hard* [17] if each of these two requirements can be satisfied individually, but they cannot be satisfied simultaneously.

One can also consider hardness of AMD problems by using notions of computational tractability that are appropriate in a centralized computational model. In this sense, Nisan and Ronen [33] initiated the study of AMD by adding computational tractability to the set of concerns that must be addressed in the design of incentive-compatible mechanisms. Succinctly stated, Nisan and Ronen's contribution to the mechanism-design framework is the notion of a (centralized) polynomial-time mechanism, *i.e.,* one in which $M(\cdot)$ is polynomial-time computable. However, as illustrated in [17], canonically hard AMD problems still remain elusive.

Recently, Feigenbaum *et. al.* [16] [15] introduce the *network-complexity criterion* which evaluates a mechanism $M(\cdot)$ in isolation based on its absolute computation and communication requirements. The notion of *protocol compatibility* [15] indicates a relative definition of complexity which requires a mechanism to be a simple extension of a widely deployed, standard protocol (*i.e.,* a protocol that has been designed without accounting for individuals misbehavior). The mechanism $M(\cdot)$ must have the same general algorithmic structure as the standard protocol and must not require substantially more computation, communication, local storage, or any other resource expenditure than the standard, regardless of whether the standard has high or low absolute network complexity. Protocol compatibility addresses two aspects of practical feasibility: computational tractability and deployability.

Approaches to resolve this tension between game-theoretic and computational properties include:

- Approximation methods. These methods let the (centralized) mechanism compute approximate outcomes based on agents' strategies while preserving the same game-theoretic properties of the original mechanism;

- Distributed computation. In this approach the outcome-rule (*i.e.,* the algorithm) of a mechanism is computed without relying on a centralized infrastructure;

- Compact preference representation languages. This approach study the question of how to provide agents with compact methods to express their preferences in order to avoid unnecessary details (thus minimizing the communication complexity) and make the problem of computing optimal outcomes more tractable;

- Dynamic mechanisms. Instead of requiring single-shot direct-revelation, this approach allows agents to provide incremental information about their preferences for different outcomes and solve easy problem instances without complete information revelation.

The challenge is to make mechanisms computationally feasible without sacrificing useful game-theoretic properties, such as efficiency and strategy-proofness. In the following section we explore recent efforts towards the design of mechanisms for distributed settings.

2.4 Distributed Algorithmic Mechanism Design

The centralized computational model of [33] is not adequate for the study of computer networks, where not only are the agents distributed, but so are the resources (e.g., bandwidth and storage). For example, Internet-based mechanisms involve distributed algorithms and any measure of their computational feasibility must reflect their distributed nature.

DAMD theory is still in its infancy and different definitions of distributed mechanisms appeared in the literature. In this section we illustrate two representative examples of a distributed setting: in the first case we consider mechanisms in which the role of a (computational) center can no longer be assumed, whereas in the second case we focus on the (distributed) communication structure that characterize the environment in which agents interact with a center.

In one attempt to address the issue of distributing the role of the computational center present in traditional MD and in AMD, Feigenbaum *et. al.* [16] adopt the concept of network complexity (see Section 2.3) and define a mechanism that satisfies the typical constraints of a distributed setting. The distributed algorithm executed over a network should limit the total number of messages sent over the network (ideally, this should be linear in the network size), the maximum number of messages sent over any one link in the network (ideally, this should be constant, to avoid hot spots), the maximum size of a message, and the local computational burden on agents. More generally, the theory of distributed implementations deals with DAMD in which the role of the center is replaced by self-interested individuals that are used to compute $g(\theta)$ and $p(\theta)$. The same agents are involved in the game created by the distributed mechanism and the goal is to bring computation, communication, and information revelation into an equilibrium. The resulting strategy space available to agents is expanded with respect to what has been discussed in Section 2.2. The two main problems one is faced to when designing a distributed mechanism are related to *computation*, whereby it is hard to determine the outcome (*i.e.*, execute in a distributed fashion the algorithm $g(\cdot)$) of the mechanism without a center to receive types and calculate $g(\theta)$ and $p(\theta)$, and the actual *execution* of the mechanism, that is to enforce the outcome of the mechanism and making sure agents (that replace the center not only for computation) make payments.

The issues that arise from the fact that the players in a mechanism are part of a distributed system are not only caused by the lack of a centralized component that bears the costs related to computation and that makes (or receives) payments. In a computational environment, mechanisms (protocols) are run by individuals that are connected by a communication network. However, the theory of mechanism design ignores this aspect, and in fact implicitly assumes a very simple communication network where each agent is directly connected to the center. This assumption, which is not realistic in most computational settings, is crucial for the mechanism design literature: as it was pointed out be Monderer and Tennenholtz in [31] its adaptation to general computational settings is non-trivial. In their work Monderer and Tennenholtz study the implications of the transition from mechanisms to protocols in communication networks. The nodes of a network are modeled as selfish agents involved in a mechanism. In this setting, a new game-theoretic feature (which expands the strategy space available to each individual) arises because agents may have the ability to modify the messages sent by other agents to the center. The goal of the system designer is to develop a mechanism that make such malicious behavior irrational for the agents.

Summarizing, the main focus of existing work has been on creating mechanisms whose outcome has various desirable game-theoretic and computational properties and in which agents have an incentive to correctly behave. Perhaps surprisingly, the traditional literature assumes participating agents to act selfishly, possibly untruthfully, if it is to their advantage, whereas the mechanism center is usually assumed to be honest and trustworthy, even when it has an incentive to be untruthful (e.g., by overstating the second-highest bid in Vickrey auctions if it gains a fraction of the selling price). As observed by Shenker [17] and by Brandt and Sandholm [3] the study of cryptographic (secure) protocols has several points in common with mechanism design. Moreover, decentralized trust requirements by which agents do no longer have to rely on a single center for controlling the mechanism, constitute an important intersection between DAMD and security schemes. As pointed out in [3], the computation of the outcome can be distributed across several distinct centers in order to eliminate a single point of failure. This is just a straightforward application of concepts akin to secure multi party computation, wherein threshold cryptography can be applied to generate shares of a secret information that have no utility if taken individually, but their ”union” can be used to reconstruct the original information.

Computational complexity that characterize the execution of a mechanism and the distributed implementation of such mechanism are not the only questions that need to be addressed to apply game-theoretic principles to computer networks. In the following section we explore the problems and the techniques that deal with the complexity of designing such mechanism.

2.5 Automated Mechanism Design

Mechanism design has traditionally been a addressed as a manual exercise wherein the mechanism designer uses experience and intuition to hypothesize that a certain outcome-rule is desirable, while attaining the SCF imposed on the system. Canonical mechanisms designed for specific settings are limited in number but have many desirable properties. These mechanisms do not rely on an a-priori knowledge of the utility functions that characterize the individuals involved in the mechanism (e.g., the VCG mechanism [7, 20, 40]), or they can be applied to any probability distribution over the utility functions of the agents (e.g., the dAGVA mechanism [13] and the Myerson auction [32]). However, these general mechanisms also have significant drawbacks, as identified in [38]:

- The most famous and most broadly applicable general mechanisms, VCG and dAGVA, only maximize social welfare. If the designer is self-interested, as is the case in many electronic commerce settings, these mechanisms do not maximize the designer's objective;

- These mechanisms only allow for payment maximization. In practice, the designer may also be interested in the outcome per se. For example, in a bandwidth allocation problem the designer may care for which player a wireless channel is reserved to.

- It is often assumed that payments can be used to tailor the agents' incentives, but this is not always practical or feasible. For example, when the players involved in a mechanism are software agents, it might be more desirable to construct mechanisms that do not rely on the ability to make payments, because many software agents do not have the infrastructure to make payments, or the infrastructure cannot be deployed.

- The most common mechanisms assume that the agents have quasilinear preferences (see Section 2.2). Under this restrictive assumption, the valuation function v_i only depends on agents' types and not on payments. Moreover individuals do not strategize on other agents' payments.

In sharp contrast to manual mechanism design, Conitzer and Sandholm [8] introduced a new approach called automated mechanism design where the mechanism is automatically created from a computationally-light algorithm. This approach has several advantages that stem from the possibility of applying AMD and DAMD results to broader settings. Furthermore, automated MD can yield better mechanisms (in terms of outcomes and/or stronger non-manipulability guarantees, [9–11]) than canonical mechanisms because of their ability to capitalize on the (probabilistic) information about the agents' preferences, whereas canonical mechanisms ignore that information.

3 Future research directions

While we suggest to refer to [38] for an extensive introduction to automated MD, we conclude the theoretical introduction to mechanism design and its declinations by adding a new degree of freedom in the design of mechanisms for computer networks.

Recent requirements in networked systems call for the design of protocols supporting autonomicity [19], whereby systems must be able to adapt in an autonomous and self-organized way to external stimuli such as dynamic network conditions, attacks, *etc*.

If the game-theoretic literature provides extensive examples of refinements of traditional MD aiming at solving problems raised by realistic settings (computational limitations, distributed environments) we propose the challenge of designing mechanisms in which the social choice function (SCF), which defines the ultimate goal of the system designer, is dynamically adjusted by the agents themselves. The idea is to have an auto-regulated system in which the SCF could change in time depending on the evolution of the system.

Here we suggest some potential research directions to achieve what could be defined as **autonomic mechanism design**. For agents to take collective decisions on the social choice function that regulates the system we identify the following non-exhaustive list of requirements:

- Context-awareness: for the SCF to be adjusted to changing conditions, agents need a feedback loop to infer the system state. Note that relevant information on system state need to be specified: *e.g.*, the number of agents in the system, the level of congestion that characterize the communication network, geographical position of agents, etc... System state information can be local or global, depending on the complexity and costs associated to the protocol used to capture the system state. Once the system state available, a (distributed) protocol binding the system state to a new SCF is required.

- Collective decision-making protocols: these protocols are necessary for agents to agree on the SCF representative for the system state. We envision a (distributed) voting scheme to reach a quorum on a new SCF. The voting scheme should take into account the communication network used to propagate agents votes (preferences) on the SCF. Indeed, messages containing voting information could be dropped and tampered with, depending on the paths they take to reach all intended destinations.

There are some problems that remain elusive though: how can the designer impose, during the system bootstrap, a finite set of SCFs agents are allowed to adopt during the system lifespan? Moreover, traditional security issues are relevant in the current context: the important decision to adopt one or another SCF should not be manipulated by adversaries that may attack the system. Also trust negotiation mechanisms should be in place to weight the decisions taken by a group of peers.

A trusted peer should have more influence than an untrusted peer in the decisions taken by the group.

Note also that agents could strategize during the decision making process: the features introduced by self-adapting the SCF to the system state should preserve the original game-theoretic properties of the mechanism.

Ultimately, the study of what goes under the name of the “Price of anarchy” represents another fertile ground for research: how does the uncoordinated decision making process compare to a centralized version of self-adaptation?

4 Application to communication systems

In this section we survey recent efforts made in the networking community that apply computational economics principles to computer networks. Instead of providing a mere description of the different approaches, we focus on the motivation that calls for the use of analytical tools akin to game theory and mechanism design. With the aim of defining a common denonominator that amalgamates different problems under the same theoretical perspective we use a unified notation to define the mechanisms presented in the following sections. Furthermore, we emphasize the game-theoretic assumptions made in each approach and discuss on their impact for a realistic deployment of such schemes.

The focus of this section is to survey methods used by computer-scientists to design or validate mechanisms and protocols taking into account possible deviations from the prescribed behavior of the actors involved in the execution of these protocols. This section does not fully explore the contribution to the game-theoretic literature made by researchers that pioneered the area of computational economics from a computer-science perspective. [17] provides a good overview of recent work that introduced new concepts in mechanism design and game theory in order to fulfill computational requirements dictated by the particularly difficult setting offered by computer networks.

4.1 MAC

In this section we present an overview of a recent effort [4] toward the study of the impact of selfish users in a CSMA/CA network. The authors focus on the problem of autonomous and independent users controlling the random deferment period (*i.e.,* the contention window) that regulates access to the medium in a wireless network. *Selfish users* are designated as the suers whoa re ready to tamper with their wireless interface in order to increase their own share of the common transmission resource.

The conflictual situation which at the same time wireless nodes and system designers face to is addressed within a game theoretical framework. Each node of the network is modeled as a player, the throughput enjoyed by a node is its payoff and the size of its contention window represents its strategy. In this context

[4] study the impact of ”cheater” nodes that modulate their contention window to obtain the best throughput and propose a **detection** and **punishment** mechanism against those players who exhibit a non-cooperative behavior.

The basic assumptions made in [4] limit the scope of the proposed solution to cellular networks in which all nodes are static and are within the same radio range. The authors also assume the existence of a security infrastructure to provide authentication services to the nodes.

The utility function that describes users' preferences over a possible set of system outcomes, and that specifically corresponds to the throughput perceived by a node n_i, is derived from Bianchi's [2] throughput model for the 802.11 protocol. The throughput r_i perceived by a node n_i depends on the access probability τ_i (that is related to the contention window) selected by n_i and on the access probability selected by all other nodes $n_{j,j\neq i}$. Formally, $r_i = \frac{\tau_i^c c_1^i}{\tau_i^c c_2^i + c_3^i}$, where $c_1^i = p_{-i}\hat{L}$; $c_2^i = p_{-i}(T^s - T^i) - s_{-i}(T^s - T^c)$; $c_3^i = (1 - p_{-i} - s_{-i})T^c + s_{-i}T^s + p_{-i}T^i$; with the following substitutions: $p_{-i} = \prod_{j\neq i}(1-\tau_j^c)\prod_k(1-\tau_k^l)$; $s_{-i} = \sum_{j\neq i}\tau_j^c \prod_{k,d\neq j,i}(1-\tau_k^c)(1-\tau_d^l)$. $\hat{L}$ is the average size of a packet, while T^i, T^s, T^c represents respectively the node's idle period, the average time to transmit a packet and the average time spent in the collision period. [4] shows that the presence of cheater nodes impose to the system an equilibrium which leads the system to stall: all player's payoff except for one most likely equals zero ($r_i = 0$) whereas the cheater fully enjoys the common spectrum. This result is known as *the tragedy of the commons* in economics.

To overcome this undesired result, the authors study the properties of a fair and optimal solution to the CSMA/CA game that has the following properties: the solution should be unique; the solution should result in a fair distribution of the system throughput; and the solution should result in the system optimal allocation of the available capacity. To derive such a solution [4] relies on the Nash bargaining framework from cooperative game theory [18]. The authors show that such an efficient and fair solution exists, and that it is achievable under the context of an optimization problem.

Having derived and calculated a desired operating point for the system is the first step toward the definition of a distributed mechanism that would enforce such an ideal equilibrium. The authors present a *dynamic* model that extends the Nash bargaining framework to allow players to interact in a **repeated** way. Repeated interaction is essential for the proposed detection and punishment mechanism to work properly. Indeed, nodes are asked to continuously monitor the activity of all other nodes and take an eventual punishment action in retaliation to *past* selfish behavior. In practice the punishment action greatly differs from the ideal punishment function that characterizes the utility function for a node n_i in the repeated game. The authors resort to selective jamming techniques to punish misbehaving nodes and rely on an imperfect (due to the randomness of the wireless channel) detection technique.

A more realistic model should take into account errors both during the detection and the punishment phase. Furthermore, a centralized solution, which would be realistic under the cellular network model, could avoid the computational burden for the nodes (in the proposed solution nodes are constantly involved in operations that are costly as compared to the normal activity of carrier sensing) and reduce detection errors.

4.2 Truthful Routing

In this section we focus on recent efforts toward truthful and cost-efficient routing protocols for wireless networks. While we refer the reader to [15–17] for a thorough overview of the seminal works in truthful routing, in this section we address a variation of the VCG [7,20,40] mechanism applied to mobile ad hoc networks in presence of selfish users. Anderegg *et. al.* propose a novel routing protocol called Ad hoc-VCG, which represent an alternative to reputation-based mechanisms (we will illustrate an example of reputation-based cooperation enforcement schemes in Section 4.3) to provide incentives for node cooperation.

Anderegg *et. al.* characterize the nodes of an ad hoc network as selfish and *omniscient* agents with a privately known type which takes into account a cost-of-energy parameter c_i and the emitting power P_{emit} required to reach neighboring nodes when forwarding packets. The omniscient adversarial model assumed by the authors is the key enabler for the wide range of possible cheating actions available to the nodes.

The network model assumed in [1] consists in a planar weighted graph in which selfish users are the nodes while a weight function assign a cost to every edge of the graph. The ultimate goal of the proposed routing protocol is to find the minimum cost path from a source of traffic to its destination. The fundamental assumption that constitutes a substantial deviation from traditional mechanism design is that agents are not assumed to know their own type. Moreover, selfish agents are assumed not to collude. The traffic model assumes a long time horizon, whereby long-lived data transfer session take place in the network making attractive for the nodes to collect as much money as possible in order to pay for sending their own data.

The routing protocol proposed in [1] belongs to the reactive (source) routing protocol: nodes initiate a route discovery when they need to send data traffic. During the route discovery phase all nodes append to a route request control packet their cost (which is the product $c_i P_{emit}$) allowing the destination to compute the complete network graph and calculate payments for intermediate nodes to relay data packets. The traffic source receives a route reply control message that include the more cost-efficient path and the payments due to intermediate nodes. Note that cost efficiency correspond to energy efficiency in the case of homogeneous cost-per-energy type. Two models are studied, the first in which the source pays intermediate nodes their cost for routing plus a premium that make truthful cost revelation the dominant strategy, the second in which the source only pays for the true cost of routing while an external entity pays provides incentives for intermediate nodes to act cooperatively. In both cases, the ad hoc-VCG protocol inherit one of the fundamental issues of mechanisms belonging to the VCG family: the routing protocol *is not budget balanced*, so an external source has to inject money in the system, which is not self-sustaining. However, the authors characterize a theoretical upper bound on the total overpayment in the system which is bounded by the path-loss exponent used for the modeling the wireless channel and the minimum and maximum cost-per-energy of the nodes.

Though ad hoc-VCG represent a starting point for truthful routing in mobile ad hoc networks, several research challenges remain elusive: as the authors point out a pro-active routing protocol could be used instead of a reactive one and several optimizations that have been proposed in the traditional ad hoc routing literature have to be investigated to understand their incentive compatibility. As an example, the use of route caches could reduce the amount of overhead generated by the original ad hoc-VCG protocol. Moreover the constantly evolving mechanism design theory allows truthfulness to be reached through a step-by-step process, increasing delays on one side but mitigating requirements on external sources of payments.

4.3 Cooperation enforcement

The effects of selfish behavior has been widely studied in the context of mobile ad hoc networks (MANET). MANETs offer a prominent example of cooperative networks whereby the nodes that form and use the network are supposed bear the cost of network operation and forward packets for the benefit of other nodes. Initially, ad hoc routing protocols assumed no deviation of the nodes from the prescribed behavior. Traditional security mechanisms to cope with malicious attackers have been subsequently presented as plugin protocols to support basic routing services. However, for example in [28], a much simpler model of node misbehavior has been showed to have a great impact on network performance. Selfish nodes, aiming at saving their energetic resources for self-interest purposes, can jeopardize the network by simply (selectively) ignoring requests to forward packets.

In [29], the authors present an analysis of coalition formation and cooperation strategies for mobile ad hoc networks. In their work, Michiardi *et. al.* model the network both with cooperative and non-cooperative game theory and show that without appropriate incentives for cooperation the network would settle to an equilibrium in which no nodes would cooperate.

Furthermore, they introduce a cooperation enforcement mechanism based on **punishment** instead of an incentive scheme based on rewards. The mechanism, named CORE [27], rely on a monitoring mechanism that exploits the broadcast nature of the wireless medium and base its punishment decisions on a *reputation* metric that represents node behavior.

In [29], the authors study for the first time the impact of **imperfect monitoring** in the iterated version of a classical game, the Prisoners' Dilemma (PD). The MANET is modeled as a playground in which random, pair-wise tournaments take place among the nodes. The important assumption made in their work is that the basic stage game (*i.e.*, the PD game) is repeated an unknown number of times, so that the iterated version falls into the class of *infinite horizon* games. The CORE mechanism is modeled as a strategy and is compared to other strategies such as the tit-for-tat (TFT) strategy to study the properties of a reputation-based strategy as opposed to a simple history-based strategy.

Strategy comparison is carried out in a numerical way, borrowing concepts akin to genetic programming: populations adopting different strategies start out with the same number of players. At each round of the iterated tournament, the size of the population using a ”loosing” strategy (*i.e.*, that is a strategy that obtain lower gains, in game theoretical terms) is decreased for the benefit of the ”winning” strategy. The use of *evolutionary game theory* allow to study the stability and robustness of a strategy, as compared to other strategies adopted by the players in the system. In [29], the authors introduce a *misperception noise* that causes players to make mistakes on the history of past moves of their opponents. The result is that strategies that have been traditionally deemed as optimal (such as the TFT strategy) diverge in presence of noise, while the CORE strategy, based on a complex methodology to build the reputation metric used to take punishment decisions, is robust against monitoring failures.

4.4 Content Distribution

Content distribution received increasing attention in the past and a wide range of mechanisms designed for efficient transfer of bulk data from one (or multiple) source to a potentially large set of destinations have emerged in the literature. In this section we focus on the impact of selfish users on the construction of the logical structures (overlays meshes or multicast trees) used to distribute the content rather than on the selfish behavior (often referred to as ”*free-riding*”) exhibited by users reluctant to share resources such as bandwidth and storage capacity.

4.4.1 Truthful multicast in selfish wireless networks

In their work [42], Wang *et. al.* address group communication among a potentially large set of users and study the impact of selfish agents deviating from the prescribed protocol intended to build and operate a multicast tree. The *truthful* multicat protocol presented in their paper is composed of a tree structure that connects the sources and receivers and a payment scheme to the relay nodes in the multicast tree.

Instead of reinventing the wheel, the authors focus on payment schemes that enforce truthful revelation of cost associated to an edge in the tree while they rely on well-known multicast-structures (and algorithms) to actually build the distribution tree such as the least cost path tree (LCPT), the pruning minimum spanning tree (PSMT), the virtual minimum spanning tree (VMST) and Steiner trees. Note that ensuring that each wireless terminal reports its cost truthfully is only one part of the story of truthful routing, which includes the routing subgame and the forwarding subgame. The proposed payment schemes guarantee that (internal) nodes will also forward packets from the tree root(s) to the destinations.

The fundamental assumptions made to model the multicast problem and to derive a payment scheme derive from the intuition behind VCG payment schemes: as explained in section 2.2.5, they rely on the *dominant strategy* solution concept whereby the utility describing other users in the system, and their corresponding actions, does not need to be taken into account for an agent to select its strategy. Moreover, as opposed to recent works that consider computationally bounded agents, in [42] the agents have no computational constraints and they are assumed not to collude.

The approach taken in [42] follows the principles of mechanism design: the multicast tree structures are regarded as the output of the mechanism (the allocation rule) while the payment scheme is used to guide the behavior of the selfish agents involved in the mechanism toward truthful revelation of their cost with the goal of maximizing *social efficiency*.

For a wide range of multicast structures, including both link weighted and node weighted families, the authors show that classical VCG payments do not guarantee truth-telling and propose alternative payment schemes (which are variations on the VCG scheme) to achieve truthful and minimum cost revelation.

It should be noted that the authors focus on a single-shot mechanism and left for future research the design of payment schemes in the multiple-session case, whereby the basic mechanism is repeated over time.x

4.4.2 Market-driven bandwidth allocation in selfish overlay networks

In their work [41] Wang *et. al.* focus on selfish overlays as networks formed by autonomous nodes that develop their own strategies to maximize their gain rather than following the prescribed behavior of the overlay creation protocol. [41] propose a market model for bandwidth allocation in selfish overlay networks to understand which overlay links should be included in a service provisioning network to connect all *upstream* nodes (*i.e.,* nodes that have a content to deliver) to all *downstream* nodes (*i.e.,* nodes that want to download a content). Moreover, once those links are established, the focus on how much bandwidth should be assigned to each overlay link in order to satisfy the traffic demands of as many downstream nodes as possible. In their market model, each upstream node has its own specific service price it prefers to charge its downstream nodes, and such a price is dynamically adjusted over time in order to maximize its economic revenue and minimize its empirical loss in the long run.

In their work, Wang *et. al.* provide a practical solution for strategic nodes to gradually solve the pricing game, by modeling them as *reinforcement learning agents* that are capable of incrementally improving their strategies through trial-and-error interactions with the external world. Indeed, the fundamental assumption taken in their work is that they model the market game as a *dynamic sequential move game* with *incomplete information* and *imperfect recall*.

The market model is based on an empirical definition of the utility function associated to each player that has the dual role of service provider and consumer. The utility function is defined as: $u_i(t) = \epsilon_1 log(1 + \frac{\sum_{j\in U_i(t)} b_i^j(t)}{C_i}) + \epsilon_2 log(1 - \frac{\sum_{k\in D_i(t)} b_k^i(t)}{C_i}) - \sum_{j\in U_i(t)} p_j(t) b_i^j(t) + p_i(t) \sum_{k\in D_i(t)} b_k^i(t)$, where $U_i(t)$ and $D_i(t)$ represent respectively the upstream and downstream node set at time t, $b_j^i(t)$ is the rate (bandwidth) assigned to a data flow from node i to node j, $p_j(t)$ is the unit charge for flows directed to node j while $\epsilon_{1,2}$ are two relative parameters that indicate the importance of empirical (*i.e.,* measured) gains or losses in terms of bandwidth as compared to economical gains or losses.

The bandwidth allocation problem is split in two games: the pricing game and the trading game. In the pricing game, upstream nodes select their best strategy that consist in selecting a transmission price. The authors propose a reinforcement learning technique to help nodes in measuring the outcome of the system for a given price and to adjust it based on a feedback signal from the system. Once transmission prices are determined, the downstream nodes select the best upstream nodes and the session throughput of the one-hop flow. Such decisions are made based on the respective utilities brought by the upstream candidates.

The game theoretical framework used in [41] is enriched by a *social choice function* (SCF) similar in its concept to the one introduced in traditional mechanism design (see Section 2.2). However, the SCF is used to assess the properties of the mechanism that derive from the game theoretic formulation of the problem. In particular, given an overlay network, the distribution of data items, and the demands from downstream nodes, the *optimality* of a specific bandwidth allocation scheme is evaluated with two metrics: the percentage of transmission requests accepted by the networks and the total end-to-end throughput in the resulting topology. Although the authors do not clearly point out the performance of their market-based approach to the problem of bandwidth allocation, the proposed methodology constitutes a strong link between game theoretical modeling and traditional mechanism design.

4.5 Selfish Replication and Caching

In this section we outline some applications of game theory to the modeling of *content networks* [23, 24] composed of selfish nodes, i.e., nodes that may cooperate by transferring objects to other nodes upon request, but select the set of locally replicated objects uncooperatively, aiming strictly at minimizing the cost incurred by their local clients (more on this cost in the sequel). The content nodes assume the role of the players in the context of replication games, whereas strategies amount to decisions regarding the set of individual objects selected for replication by each node. The conflict of interest between the different nodes arises implicitly (non zero-sum games) as each one would prefer the implementation of a potentially different global object placement (which is the outcome of the game). These global placements, however, cannot in general be implemented concurrently and therein lies the conflict and the game.

4.5.1 Uncapacitated Replication

Chun et al. [5] have studied the uncapacitated version of selfish replication, in which the basic assumption is that each node has at its disposal as much storage as it wishes to utilize (potentially infinite). They consider an object universe $O = \{o_1, o_2, \dots, o_N\}$ and assign a *placement cost* a_{ij} for replicating object o_j at node v_i. This cost can capture a direct cost for the storage space that the object occupies as well as indirect costs related to maintaining this replica consistent by fetching updates from an origin server. They also consider p_{ij} as the relative preference of the local population of v_i for o_j. Let $m_s(v_i, o_j)$ denote a node that: (1) replicates o_j under the outcome s, and (2) is at minimum distance from v_i according to some (metric) distance function d. Then the cost of v_i is defined to be:

$$c_i(s) = \sum_{o_j \in s_i} a_{ij} + \sum_{o_j \in (O/s_i)} p_{ij} \cdot d(v_i, m_s(v_i, o_j)) \tag{1}$$

The fundamental assumption that nodes have infinite storage capacity eliminates any “friction” between objects of different type and permits treating the problem as if there was only one type of object instead of N distinct ones. Essentially, one has to come up with a Nash equilibrium replication strategy for a basic game that involves only one type of object, and then repeat it it across the different (but non interacting) objects, to obtain a Nash equilibrium for the corresponding multi-type original problem. The *basic game* corresponds to a specific object o_j and involves the following (simpler) cost function.

$$c_i(s) = a_{ij} s_i + p_{ij} d(v_i, m_s(v_i, o_j))(1 - s_i), \tag{2}$$

where s_i simplifies to a binary decision: 1 if v_i replicates o_j, and 0 otherwise. For convenience one can drop the dependence on j and just talk about an object o with placement cost a and consider the even simpler cost $c_i(s) = as_i + d(v_i, m_s(v_i, o))(1 - s_i)$. For the basic game, Chun et al. [5] showed the following:

- Pure Nash equilibria always exist.
- The price of anarchy (*PoA*) of the basic game in general graphs with minimum inter-node distance d^{min} and maximum inter-node distance (diameter) d^{max} is: *PoA*=1 if $a \leq d^{min}$ and at least $O(n)$ when $a > d^{max}$. Tighter bounds are given for specific graphs (clique, star, line, and D-dimensional grid).

One can extend the basic game into a *payment game* that permits nodes to offer payments to other nodes as an incentive to replicate objects on their behalf. For the payment game, Chun et al. [5] showed the following:

- Any pure Nash equilibrium for the basic game is also pure Nash equilibrium for the payment game. Also, the cost of a socially optimal solution is the same for both games since the net payments made cancel out. Consequently the price of anarchy of the payment game is at least as high as the one for the basic game.

- In the payment game, the socially optimal solution can always be implemented by a pure Nash equilibrium, i.e., the optimistic price of anarchy is 1.

4.5.2 Capacitated Replication

Laoutaris et al. [22] have studied the capacitated version of selfish replication in which node v_i has already purchased a fixed amount of storage for up to C_i unit-sized objects and cannot buy more. The consideration of fixed capacity constraints at all the nodes introduces "friction" between different types of objects, in the sense that a global placement for one object type affects the feasible global placements for other object types because the two compete for the common storage at the nodes involved. Therefore, the capacitated version of selfish replication cannot be reduced to the study of single type equilibria, as with the basic game of Chun et al. [5].

In the capacitated version, v_i's strategy $s_i \in S_i$ amounts to selecting C_j objects out of the total N and therefore $|S_i| = \binom{N}{C_j}$. The corresponding cost to be minimized is:

$$c_i(s) = \sum_{o_j \in O} p_{ij} \cdot d(v_i, m_s(v_i, o_j)) \tag{3}$$

Laoutaris et al. [22] showed the following:

- The best response of a node amounts to solving a 0/1 Knapsack problem [35] by selecting objects according to their "excess gain" which is given by the product of an object's popularity and the distance spared by replicating a local copy of the object (as compared to fetching the previous closest copy from some remote node). Due to the unit-size assumption, such a 0/1 Knapsack problem can be solved optimally in polynomial time through a greedy strategy.

- Pure Nash equilibria always exist and can be obtained by having the nodes compute their best responses sequentially.

- Simple distributed solutions for computing such equilibria exist and require the transmission of rather limited amounts of information (each node to advertise only its placement (C_i object identities) as opposed to its entire popularity vector (N pairs of object identity-object popularity)).

The aforementioned capacitated version has also be considered under the view point of on-line algorithms. Specifically, instead of replication, Laoutaris et al. [21]

and Smaragdakis et al. [39] have employed caching (i.e., demand driven temporary storage of objects combined with replacement) and have looked into the problem of resolving contention in a non-stationary environment (where group membership, access distance, and popularity may change).

4.6 Selfish Network Creation

Fabrikant et al. [14] have introduced a game that models the creation of Internet-like networks by selfish agents without central design or coordination. In this game a player v_i is a network node (e.g., an autonomous system (AS)) and its strategy s_i is the set of other nodes to which it chooses to establish direct links, each installed link having cost a. The goal of v_j is to minimize the sum of the cost of the links it purchases and its distances to all other nodes.

$$c_i(s) = a \cdot |s_i| + \sum_{v_j \in V} d_{G[s]}(v_i, v_j), \tag{4}$$

where $G[s]$ denotes the undirected graph that results from the outcome s (the superposition of linking decisions from all nodes). For this network creation game Fabrikant et al. [14] showed the following:

- Computing the best response of a node is an NP-hard problem (shown through a reduction from DOMINATING SET).
- For $a > n^2$ *PoA* is $O(\sqrt{a})$.
- For any $\epsilon > 0$ there exists a Nash equilibrium with *PoA*$> 3 - \epsilon$.

Chun et al. [6] have studied experimentally a generalization of the game of Fabrikant et al. [14] by dropping the assumption that all links cost the same and instead modeling links whose cost is proportional to the distance between two nodes. Using this model and an iterative best response strategy based on randomized local search they study the statistical characteristics of the emerged graphs as well as their resilience to random node failures and intentional attacks.

References

[1] L. Anderegg and S. Eidenbenz. Ad hoc VCG: a truthful and cost efficient routing protocol for manet with selfish agents. In *In Proc. of the 9th ACM Annual International Conference on Mobile Computing and Networking, (MOBICOM)*, 2003.

[2] G. Bianchi. Performance analyisis of the 802.11 distributed coordination function. *IEEE Journal of Selected Areas in Communications*, 18, 2000.

[3] F. Brandt and T. Sandholm. On Correctness and Privacy in Distributed Mechanisms. In *In Proc. of the Agent-Mediated Electronic Commerce (AMEC) workshop*, New York, NY, USA, 2004.

[4] M. Cagalj, S. Ganeriwal, I. Aad, and J.P. Hubaux. On selfish behavior in CSMA/CA Networks. In *In Proc. of the 24th IEEE International Conference on Computer Communications (INFOCOM)*, Miami, FL, USA, 2005.

[5] Byung-Gon Chun, Kamalika Chaudhuri, Hoeteck Wee, Marco Barreno, Christos H. Papadimitriou, and John Kubiatowicz. Selfish Caching in Distributed Systems: A Game-Theoretic Analysis. In *Proceedings of ACM PODC '04*, Newfoundland, Canada, July 2004.

[6] Byung-Gon Chun, Rodrigo Fonseca, Ion Stoica, and John Kubiatowicz. Characterizing selfishly constructed overlay networks. In *Proceedings of IEEE INFOCOM'04*, March 2004.

[7] E. H. Clarke. Multipart pricing of public goods. *Journal of Public Choice*, (11):17–33, 1971.

[8] V. Conitzer and T. Sandholm. Complexity of mechanism design. In *In Proc. of the 18th Annual Conference on Uncertainity in Artificial Inttelligence (UAI)*, pages 103–110, Edmonton, Canada, 2002.

[9] V. Conitzer and T. Sandholm. Application of automated mechanism design. In *In Proc. of the 19th Annual Conference on Uncertainity in Artificial Inttelligence (UAI) workshp on Bayesian Modeling Applications*, Acapulco, Mexico, 2003.

[10] V. Conitzer and T. Sandholm. Automated mechanism design: Complexity results stemming from the single-agent stting. In *In Proc. of the 5th International Conference on Electronic Commerce (ICEC)*, pages 17–24, Pittsburgh, PA, USA, 2003.

[11] V. Conitzer and T. Sandholm. Computational criticisms of the revelation principle. In *In Proc. of the Conference on Logic and the Foundations of Game and Decision Theory (LOFT)*, Leipzig, Germany, 2004.

[12] R. K. Dash, N. R. Jennings, and D. C. Parkes. Computational-Mechanism Design: A Call to Arms. *IEEE Intelligent Systems*, pages 40–47, 2003.

[13] C. d'Aspremont and L. A. Gérard-Varet. Incentives and incomplete information. *Journal of Public Economics*, 11:25–45, 1979.

[14] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. On a network creation game. In *Proceedings of ACM PODC '03*, pages 347–351, New York, NY, USA, 2003.

[15] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based Mechanism for Lowest-Cost Routing. *Proceedings of the 21st Symposium on Principles of Distributed Computing, ACM Press, New York*, pages 173–182, 2002.

[16] J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the Cost of Multicast Transmissions. *Journal of Computer and System Sciences*, 63:21–41, 2001.

[17] J. Feigenbaum and S. Shenker. Distributed Algorithmic Mechanism Design: Recent Results and Future Directions. In *In Proc. of Dial-M*, Atlanta, Georgia, USA, 2002. ACM.

[18] D. Fundenberg and J. Tirole. *Game Theory*. MIT Press, 1991.

[19] A. G. Ganek and T. A. Corbi. Corbi: The dawning of the autonomic computing era. *IBM SYSTEMS JOURNAL*, 42(1), 2003.

[20] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[21] N. Laoutaris, G. Smaragdakis, A. Bestavros, and I. Stavrakakis. Mistreatment in Distributed Caching Groups: Causes and Implications. In *Proceedings of IEEE INFOCOM'06*, Barcelona, Spain, April 2006.

[22] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis. Distributed Selfish Replication. *IEEE Transactions on Parallel and Distributed Systems*, 2006. [in press].

[23] N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis. Joint Object Placement and Node Dimensioning for Internet Content Distribution. *Information Processing Letters*, 89(6):273–279, March 2004.

[24] N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis. On the Optimization of Storage Capacity Allocation for Content Distribution. *Computer Networks*, 47(3):409–428, February 2005.

[25] A. MasColell, M. Whinston, and J.R. Green. *Microeconomic Theory*. Oxford Univ. Press, 1995.

[26] R.P. McAfee. A Dominant Strategy Double Auction. *Journal of Economic Theory*, 56:434450, 1992.

[27] P. Michiardi and R. Molva. CORE: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *CMS'2002, Communication and Multimedia Security 2002 Conference, September 26-27, 2002, Portoroz, Slovenia / Also published in the book : Advanced Communications and Multimedia Security /Borka Jerman-Blazic & Tomaz Klobucar, editors, Kluwer Academic Publishers, ISBN 1-4020-7206-6, August 2002 , 320 pp*, Aug 2002.

[28] P. Michiardi and R. Molva. Simulation-based analysis of security exposures in mobile ad hoc networks. In *EW'2002, European wireless 2002 - February 25-28, 2002, Firenze, Italy*, Feb 2002.

[29] P. Michiardi and R. Molva. Analysis of coalition formation and cooperation strategies in mobile ad hoc networks. *Ad Hoc Networks, Volume 3 N 2, March 2005*, 2005.

[30] P. Milgrom and J. Roberts. Adaptive and Sophisticated Learning in Normal Form Games. *Games and Economic Behavior*, (3):82–100, 1991.

[31] D. Monderer and M. Tennenholtz. Distributed Games: From Mechanisms to Protocols. In *In Proc. of the Sixteenth National Conference on Artificial Intelligence*, Orlando, Florida, 1999. (AAAI).

[32] R. Myerson. Optimal auction design. *Mathematics of Operation Research*, 6:58–73, 1981.

[33] N. Nisan and A. Ronen. Algorithmic Mechanism Design. In *In Proc. of Games and Economic Behavior*, number 35, pages 166–196, 2001.

[34] M. J. Osborne and A. Rubistein. *A course in game theory*. MIT Press, 1994.

[35] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, New York, 1998.

[36] D. Parkes. *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 2001.

[37] D. Parkes, J. Kalagnanam, and M. Eso. Achieving Budget-Balance with Vickrey- Based Payment Schemes in Exchanges. In *In Proc. of the 17th Intl Joint Conf. Artificial Intelligence (IJCAI),*, page 11611168, Seattle, Washington, USA, 2001.

[38] T. Sandholm. Automated mechanism design: A New Application Area for Search Algorithms. In *In Proc. of the International Conference on Principles and Practice of Constraint Programming (CP)*, 2003.

[39] G. Smaragdakis, N. Laoutaris, I. Matta, A. Bestavros, and I. Stavrakakis. A Feedback Control Approach to Mitigating Mistreatment in Distributed Caching Groups. In *Proceedings of IFIP Networking 2006*, Coimbra, Portugal, May 2006.

[40] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, (16):8–37, 1961.

[41] W. Wang and B. Li. Market driven bandwidth allocation in selfish overlay networks. In *In Proc. of the 24th IEEE International Conference on Computer Communications (INFOCOM)*, Miami, FL, USA, 2005.

[42] W. Wang, X.Y. Li, and Y. Wang. Truthful multicast in selfish wireless networks. In *In Proc. of the 10th ACM Annual International Conference on Mobile Computing and Networking, (MOBICOM)*, Philadelphia, PA, USA, 2004.