18

Selfish Agents and Economic Aspects

Leonid Scharf

18.1 Introduction

Routing algorithms in wireless networks can be developed using different techniques. In a network consisting of selfish nodes, traditional approaches will fail, since nodes are not interested in forwarding foreign data and might refuse to cooperate. In this chapter we discuss algorithmic mechanism design, which provides a possibility to deal with selfishness of network nodes.

Nodes in a network are called selfish, if they are not providing own resources to other nodes for free, which is usually bandwidth for transit data transfers. Forwarding foreign data in a wireless network means for a battery-powered node spending valuable energy, which can be considered as an expense for a node. We will call such expenses costs. A rational node will provide a service when offered compensation for its costs, which could be some kind of virtual currency. The value of such a transaction to a node is the difference between the amount of compensation and the own costs of providing this service. In this chapter we will not consider malicious nodes, for example, nodes with the intent of jamming or tapping the network.

Selfish entities are called agents and are assumed to be rational and responding to incentives when asked to cooperate. For each network model and given routing problem, the aim is to design a cheating-proof mechanism, which computes outcome (the route) and payoff function in polynomial time and minimizes, if possible, the overall route cost.

18.2 Mechanism Design

In Mechanism Design multiple independent agents in a game-theoretic setting are considered. For each given problem, the aim is to develop a mechanism, which computes an output and a payoff vector depending on the agents’ strategies. Properties of the output depend on the system goal. In the case of unicast
network routing the output could be the path with the least energy cost from source to destination.

Consider a system with $n$ independent selfish agents. Each agent $a_i, i \in 1, \ldots, n$ is described by its type $t_i \in T_i$, which is some private information, known only to the agent himself. This information could be, for example, the cost, the agent has to pay to forward data from another agent in the network. Types of all agents define a type vector $t = (t_1, \ldots, t_n)$ in the game. In each round, agents are asked to select their strategies from corresponding sets, i.e. agent $a_i$ selects a strategy $s_i$ from the strategy space $S_i$, which is given to the agent by the mechanism. A strategy is in words a plan or a decision rule, which defines an action of an agent in a given state of the system. Agents’ strategies define a strategy vector $s = (s_1, \ldots, s_n)$. After the mechanism has collected all strategies from the agents, it computes the output $o = o(s), o \in O$ (set of all possible outputs) and the payoff vector $p = p(s), p = (p_1, \ldots, p_n)$.

Preferences of an agent $a_i$ are represented by its valuation function $v_i(t_i, o)$, which computes a monetary value of the output $o$ to the agent. This value can be, for example, the cost of the agent (a negative number) in the case it is selected in the output $o$ and zero otherwise. The utility of the agent is a function $u_i(t_i, o, p_i), u_i : T_i \times O \times \mathbb{R} \rightarrow \mathbb{R}$ and is often assumed to be the sum of valuation function and payment, i.e. $u_i(t_i, o, p_i) = v_i(t_i, o) + p_i$. In other words, utility of an agent is its payment minus its cost and depends on its type, the outcome and the payment. Agents are assumed to be rational, which means that each agent always selects a strategy, which maximizes its expected utility. A strategy $s_i$ is called dominant if it maximizes the utility of agent $a_i$ regardless of what other agents do. A great benefit of dominant strategy is that the agent does not need to take the behavior of other agents into account, which simplifies the agent’s algorithm.

**Definition 18.2.1 (Mechanism).** A mechanism $M = (o, p)$ defines a strategy space $S_i$ for each agent $a_i$, the output function $o = o(s), o : S_1 \times S_2 \times \ldots \times S_n \rightarrow O$, and a payment vector $p = (p_1, \ldots, p_n)$ with $p_i$ being the payment to the agent $a_i$.

A mechanism is called direct revelation mechanism, if strategy spaces consist of all possible types of corresponding agents.

A mechanism is strategyproof or truthful if the agents’ types are part of the strategy space and revealing the true type is always a dominant strategy for each agent [384]. Such a mechanism removes speculations and counter-speculations among agents. The most important class of mechanisms are Vickrey-Clarke-Groves (VCG) mechanisms by Vickrey, Clarke and Groves.

**Definition 18.2.2 (VCG Mechanism).** A direct revelation mechanism belongs to the VCG family if the computed output $o$ maximizes the objective function $g(o, t) = \sum_i v_i(t_i, o)$ which is called social welfare. Furthermore the