10

How Computer Networks Can Become Smart

Ricardo Bagnasco and Joan Serrat
Universitat Politécnica de Catalunya
Spain

1. Introduction

In the past, computers had a simpler life, few applications had network capabilities and users only wanted to connect to Internet in few specific situations as to check emails, view some mostly plain-text webpage or to play few games with their friends using a "not very wide" bandwidth. But several aspects have changed nowadays, we have better connections, more content available on Internet (the biggest computer network ever known), there are new devices able to connect allowing the presence of more network elements than before and, in addition, the rapid advances in real-time supported applications that are expected, make it difficult to manage them as has been done. Today's network applications require extensive human involvement in management but we need a more independent (smart) network so research community has been developing lot of work in the last few years trying to ease this issue and to introduce autonomic behaviour in computer and telecommunication networks[1]. Manage a network is a complex task where the purpose is to control the behaviour of their elements in order to fulfil some high level goals. The automation of networks and systems thus far has been applied to simplified problems. Typically, we can find some works about improving routing[2]; configuring a network element to better support some application; or controlling the compliance of contracts between peers. Among the different enabling technologies for automate network communications the policy based management is one of the most representatives. This paradigm allows segregating the rules that govern the behaviour of the managed system from the functionality provided by the system itself (Boutaba & Xiao 2002). This is particularly useful in environments where it is necessary to dynamically change goals and add new services or resources. The most well developed implementations of this paradigm rely on pre-programmed rules based on logic. Nevertheless, to be called autonomic, a system must show a degree of flexibility to self adapts to changes and this is hardly achievable by means of static policies. Then, it emerges the need of learning and to figure out the mechanism that will lead to new policies.
Another important aspect of a network is that it has high and low level goals so it would be important to understand the whole vertical structure from sending/receiving bits till more complex application tasks in order to combine and enforce the best action choice. One

[1] Our focus is on computer networks but the concepts we mention here are valid for both kind of networks and probably they will merge in a near future.
[2] Routing: how to select paths to send network traffic.

useful methodology to address these challenges is using multi-agent derived frameworks, especially those that acknowledge the existence of other cooperative/competitive agents in the same environment. In this chapter we are going to study the most important characteristics of a network environment and we will show recent approaches which will establish the basis of the future advances we are going to experience in our communications.

There are many different topics and theories around this subject ongoing today so it is quite difficult to present all of them in a comprehensive and coherent manner still covering enough depth. We adopted here a tutorial-like organization where we introduce the concepts, frameworks and general theories and then present topics in the specific domain of computer networks. The remainder of this chapter is organized as follows: first, we explore the dimensions of a network that will affect any proposed solution; then we continue with the modelling techniques and math involved in current approaches to achieve the goal of a self-managed network; later we analyze typical scenarios and most interesting functions that would be important to manage in an autonomic fashion and finally, recognizing that to enable full automation of the networks and systems there are still many unsolved problems and open issues, we are going to mention at the end of this chapter some hot topics waiting for solutions coming from future research.

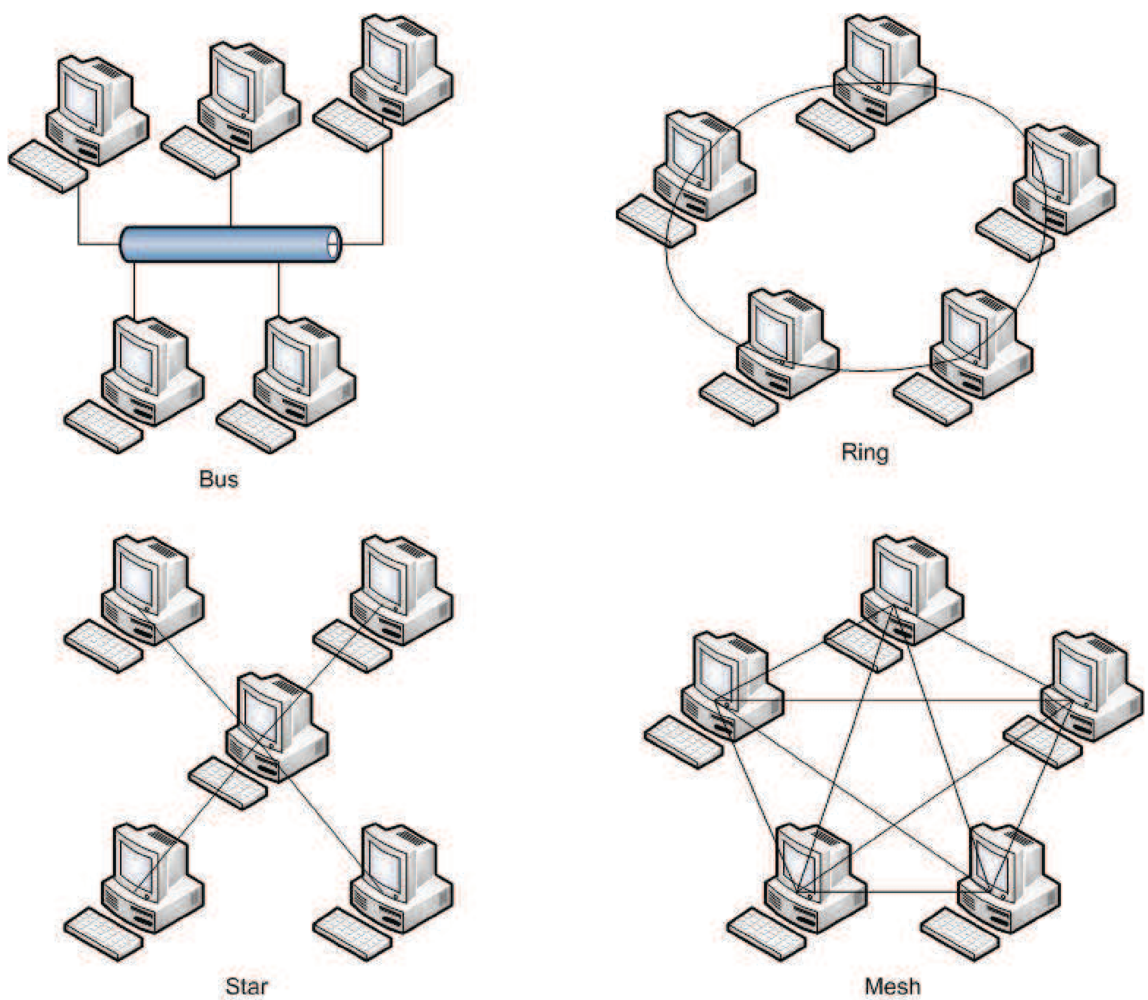


Fig. 1. Example of network topologies

2. Dimensions that affect the solution in a network environment

Besides the functional aspects, there are several features of the network itself that can influence the solution design. We should be aware of the scenario's characteristics in order to choose wisely how to approach the situation. Here we will enumerate some network related orthogonal dimensions that could be used to define the taxonomy of those scenarios.

- Network element characteristics:
  - Internal Resources: memory, processing power (CPU), storage, etc;
  - Relevant external features as screen size if it is a video device;
  - Technologies, protocols and "languages" it can understand.
- General network characteristics:
  - Size: from two to near infinite. It affects the scalability of the algorithm.
  - Topology affects the way agents can communicate (see fig.1):
    - Bus: they all share same line of communication;
    - Ring: they are connected in a circular sequential way;
    - Star: there is one element that concentrate the communications;
    - Mesh: they are all fully connected.
  - Organization: Affects the coordination.
    - Brief history of management in centralized networks:
      To manage a centralized network the Simple Network Management Protocol (SNMP) was developed in response to the growth of Internet devices. It is a protocol for retrieving information, with the use of Management Information Base semantics (MIB[3]) is possible to define what can be communicated. This framework and protocol have been widely implemented in commercial products, and even become the de facto standard for network management. Due to the maturity of SNMP, many device manufacturers include support for this protocol in their products, and many SNMP management solutions exist off-the-shelf for common operating systems and programming languages. However it is still needed to write the management code by control system programmers. This was not considered sophisticated enough to cope with the approach undertaken by the International Standardization Organization (ISO) within the Open Systems Interconnection (OSI) framework. Because of that, the telecommunication community defined a common set of standards which could apply to both worlds. Later these standards were used for telecommunications management network (TMN) defining five functional areas called FCAPS (Fault, Configuration, Accounting, Performance and Security Management). Nowadays they are working on a newer model to replace the aging TMN. The next solution was Common Information Model (CIM) plus eXtensible Markup Language (XML), this solution provides open schema to describe objects and also enables application interoperability without APIs. Within a single organization a hierarchically structured distributed management system can be appropriate. Operators can manage the automated managers effectively. The management system can be partitioned

---

[3] MIB: it is a database that contains information about each device in a network. It is used by network protocols to control performance and productivity.

functionally with dedicated manager components performing configuration, fault or security management.
As in any organization, the centralized approach is excellent in order to coordinate and to achieve goals.

- Peer-to-Peer (P2P): Decentralizing the network.
The centralized approach is not so good when we scale up the amount of integrants. The number of emitting nodes producing information can easily exceed a certain critical value making the centralized server unavailable. Moreover if dynamic information is stored it would be important to update a huge database and to manage it properly. The P2P approach adds flexibility, scalability and survival in the presence of some problems that would be not offered by any centralized approach. On the other hand we loose the potential to achieve fast agreements and hinder the coordination in order to fulfill the network goals because now we need to conciliate and interact more than in the previous approach. Depending on the organization we can find a centralized initially and then decentralized P2P protocol with a central server performing some operations at the beginning of the relation (for example: Napster), or a purely decentralized with no central coordination activity (for instance Freenet) or partially centralized with some nodes acting as supernodes with local central indexes (for example Kazaa). The content can be distributed in a structured way, where network topology is tightly controlled and files are placed in precise locations (Chord, CAN, PAST, Pastry) or loosely structured where locations are affected by routing hints but are not completely specified (Freenet) or unstructured where data is distributed randomly over peers and broadcasting mechanisms are used for searching (Napster, KaZaa). Some common problems in P2P environment are: how to discover resources; how to stimulate the sharing; how to select from different sources; how to assure the presence of all the pieces; and how adapt to change in the location distribution where both agents and targets are moving.
- Hybrid: the best approach is probably a hybrid solution combining the advantages of both architectures in accordance to the particular situation involved. P2P may be useful between cooperating distributed managers but a hierarchy can provide a single point of access to the network and allow a better balance between independent coordination of the network elements and the need of administrative control. In summary, P2P is better to survive while centralized is better to achieve goals.

3. Modelling and learning techniques

We can define a good model as the one that allows us to express in an efficient way the characteristics that are important for the functionality we want to offer. There are several theories and models that describe network scenarios but actually they are somehow incompatible so we are still looking for a unified theory (if it exists).
It is a common agreement that solution complexity depends on the adopted modelling, therefore the designer must be careful about choices he makes because they will have a big impact in the future system. That implies he should have a good understanding of both the task and the network environment. Probably the first decision to be made is to choose the

model's scope, some researchers decide to limit it to the network system while others also include the final users and/or the context of execution. The next step is to define the control variables that will be our inputs and which will be the measures to be considered as outputs. The last task is to choose the best technique to find the solution to the situation. Define an architecture is a common way to face the modelling task so there has been several tries to design one that could assure the functionality of the system in a network environment. IBM defined the autonomic computing architecture (IBM, 2001) similar to the agent model from artificial intelligence, with sensors to capture the state, actuators to affect the environment and an inner loop to reason about. The term "autonomic" comes from the autonomic nervous system that acts as the primary conduit of self-regulation and control in human bodies. Later, Motorolla redefined it to some extent to include explicitly the compatibility with legacy devices and applications, creating the FOCALE architecture (Strassner et al., 2006). For the "choose a technique to build a solution" task there are basically two tools: logic and maths. With logic we can express our knowledge about the world and with math we can classify, predict and express a preference between options. In particular two branches of math, utility based (there is a function to maximize) and economical approach (mostly market based and game theory) are common models used by research community.

We are looking for methods that allow learning by improving the performance so we need a measure to quantify the degree of optimality achieved, because of that we are going to focus on mathematical techniques. The simplest expression of a process model is some function giving us an output from the inputs in the form of *f (input_1,… , input_i) = (output_1,… , output_j)*. If we have an analytic model for the system we can use it and try to solve the equations. But if we do not know the relationships between inputs and outputs we should select a method that is capable of dealing with uncertainty such as Bayesian networks that learns them. We are interested in the kind of tasks that are so complex that there is no such analytical model or it is hard to solve it so creates the need of new kind of solutions. Learning then becomes a mean to build that function when we lack of tools to solve the maths involved or we have no perfect knowledge about the relationships between variables.

Among several possible kind of tasks to be learned (classification, regression, prediction, etc) we will mention the research in several areas but explaining better the planning one. The task to be learned will be how to choose actions in order to follow the best path of functioning during the life cycle of the system.

There are basically three main approaches to learning: supervised, unsupervised and reward-based (also known as reinforcement learning). In supervised learning the critic provides the correct output; in unsupervised learning no feedback is provided at all; in reward-based the critic provides a quality assessment (reward) of the learner's output. The complexity of the interactions of multiple network elements make supervised learning not easy to apply to the problem because they assume a supervisor that can provide the elements with many different samples of correct answers which is not feasible in network scenarios. We are dealing with dynamic environments in several ways; sources of changes can come from variations in network resources, user requirements (including the pattern of user requests), service provision and network operational contexts (Tianfield, 2003). Therefore, the large majority of research papers in this field have used reward-based methods as those modelled by a Markov Decision Process because they have the appealing characteristic of allowing the agents to learn themselves in an unknown dynamic environment directly from its experiences. Other great advantage of this model is that do not need to memorize all the process, it assumes that the consequences of choices only depends on the previous state (not all history, just the previous state matters). In this section we will mention some important characteristics of Markov Process and few variants that make different assumption about the environment. Then we will analyze the dynamics that emerges depending on if we are in a cooperative or competitive environment and we will mention few approaches that already exist to give some insights about the current development stating pros and cons.

3.1 From single-agent to multiagents modelling

If we are in a situation where we ignore the exact dynamic but it is known that it is statistically stationary (probabilities of transition between states do not vary) we can in average learn to act optimally. Borrowing ideas from the stochastic adaptive control theory has showed to be of great help in network management so we will start briefly mentioning some variants of the single-agent approach of Markov based techniques. Generally we can say that value-search algorithms (like q-learning) have been more extensively investigated than policy search ones in the domain of telecommunications.

Modelling the situation as a Markov Chain means that we assume that the probabilities of state transitions are fixed and that we do need to memorize all the past because the transitions only depend on the previous state. The application of that model to processes is called Markov Decision Process (MDP). In its simplest version it assumes perfect information but there are two extensions to MDP, one that it is useful when we can not see all the parameters of the real states but some others related, called Hidden Markov Model. It models the situation where states are not directly observable. And the second variant called Partially Observable Markov Decision Process (POMDP) that allows modelling the uncertainty or error in the perception of the world where states are partially observable.

Returning to MDP, the most well developed solution in machine learning is Reinforcement Learning (RL) where the agent learns by trial-and-error interaction with its dynamic environment (Sutton & Barto, 1998). At each time step, the agent perceives the state of the environment and takes an action which causes the environment transit into a new state. The agent receives a scalar reward signal that evaluates the quality of this transition but not explicit feedback on its performance, the goal is to maximize the reward of the process. In the case of single agent RL there are good algorithms with convergence properties available.

The sparse nature of many networks seems to indicate that multiagent techniques are a good option to design and build complete solutions in these scenarios. A multiagent system (Weiss, 1999) is defined as a group of autonomous, interacting entities sharing a common environment. A specific work on multi-agent systems for automated network management founded in (Lavinal et al., 2006) where it adapts domain specific management models to traditional agency concept and describes precise agent interactions. Not only assign roles to agents (e.g. managed element, manager) but also all agent-to-agent interactions are typed according to their roles and task dependencies. Other basic approach to multiagents learning is to model the situation with a hierarchical technique where the situation is seen as several independent tasks with the particularity that the integration of each best policy is also the global best option. It seems good but when the task is not easily decomposed hierarchical then we need some different approach. The most extended solution in the machine learning literature is called Multiagent Reinforcement Learning (MARL). This modelling is an extension from the single agent reinforcement learning so it does not require exact knowledge of the system and assumes the environment is constantly changing and hence requires constant adaptation. Also it has the same drawbacks: the learning process may be slow; a large volume of consistent training experience is required; and it may not be able to capture complex multi-variable dependencies in the environment. Furthermore, challenges not presented in the single-agent version appear in MARL like the need of coordination, scalability of algorithms, nonstationarity of the learning problem (because all agents are learning at the same time) and the specification of a learning goal. The last topic is related to the fact that in single agent version we had only one reward but now we could have many involved. If we were in a fully cooperative environment we could add the rewards and maximize the sum. But in a competitive environment we need a different approach and a new kind of goals appears, as to arrive to a Nash equilibrium[4] or convergence to a stationary policy as stability measure. However some concerns have been raised against its usefulness because the link between convergence to equilibrium and performance is unclear; in fact sometimes such equilibrium corresponds to suboptimal team behaviour. To find the right goals in MARL algorithms (convergence, rationality, stability and/or adaptation) is still an open field of research.

The generalization of the MDP framework to the multiagent case is called stochastic game, where it appears the joint action set combining all the individual actions. As it is stated in the survey of MARL techniques done by (Busoniu et al., 2008) the simplest version of a stochastic game is called static (stateless) game where rewards depend only on the joint actions. Mostly analyzed with game theoretic focus, often the scenario is reduced to two competitive agents with two actions in zero-sum[5] or general-sum games in order to limit and control the complexities involved.

The taxonomy of MARL algorithms depends on which dimensions we choose to classify them. We can focus on:

- The degree of cooperation/competition;
- How each agent manage the existence of others (unaware of them, tracking their behaviour, modelling the opponent);
- The need of information, sometimes agents need to observe the actions of other agents and in other algorithms they need to see in addition their rewards;
- Homogeneity of the agent's learning algorithms (the same algorithm must be used by all the agents) vs heterogeneity (other agents can use other learning algorithms);
- The origin of the algorithms (Game Theory; Temporal Difference –RL-; Direct Policy Search) but it is important to notice that there are many approaches that actually mix these techniques.

The coordination is another topic itself, to explicit coordinate agent's policies there are mechanisms based on social conventions, roles and communication that could be used in cooperative or competitive environments. In the case of social conventions and roles, the goal is to restrict the action choices of the agents. Social conventions impose an order between elections of actions. It dictates how the agents should choose their action in a coordination game in order to reach equilibrium. Social conventions assume that an agent can compute all equilibrium in a game before choosing a single one. And, to reduce the size

[4] A joint strategy such that each individual strategy is a best response to the others, no agent can benefit by changing its strategy as long as all other agent keep their strategies constant. The idea of using Nash is to avoid the learner being exploited by other agents.

[5] A situation in which a participant's gain or loss is exactly balanced by the losses or gains of the others participants. One individual does better at another's expense.

of action set in order to reduce the expense of the calculation, it is useful to assign roles to agents (some of the actions are deactivated). The idea of roles is to reduce the problem to a game where it is easier to find the equilibrium by means of reducing the size of action sets. But, if we have too many agents we still need a method to reduce the amount of calculation and that is the utility of extra structures as the coordination graph appeared in (Guestrin et al., 2002). It allows the decomposition of a coordination game into several smaller sub-games that are easier to solve. One of the assumptions here is that the global payoff function can be written as a linear combination of many local payoff functions (each involving few agents).

Another important issue in multiagents is communication; we can define direct or indirect communication. Examples of the first include shared blackboards, signalling and message-passing (hard-coded or learned methods). Indirect communication methods involve the implicit transfer of information through modification of the world environment. For example leaving a trail or providing hints through the placement of objects in the environment. Much inspiration here comes from insects' use of pheromones.

In addition, in complex environments it is not realistic to assume complete information then ignorance needs to be taken into account. Ignorance could lead to incomplete or imperfect information. Incomplete information means that the element does not know the goals of the other elements or how much do they value their goals. Collaboration is a possible mechanism to overcome this. Imperfect means that the information about the other's actions is unknown in some respect. The ignorance could manifest in the form of: uncertain information; missing information; or indistinguishable information. Other sources could be measurements with errors or unreliable communication. Bayesian networks and mathematical models as POMDP have been used with some degree of success to try to fix this problem because they explicitly model the ignorance but on the other hand they are difficult to use and do not scale well. Actually it could be even more complicated, because in POMDP there is no good solution for planning with an infinite horizon. The problem of finding optimal policies in POMDP is not trivial, actually is PSPACE-complete (Papadimitriou & Tsitsiklis, 1987) and it becomes NEXP-complete for decentralized POMDPs (Bernstein et al., 2000).

Finding an exact solution with all those problems mentioned above is still infeasible so sometimes it is used some metaheuristic (biological inspired are probably the most well known ones), they are approximate algorithms of the search process (Alba, 2005), this means that they are no guaranteed to find a globally optimum solution and that, given the same search space, they may arrive at a different solution each time are run. But because they work empirically with some decent performance they are of particular interest for networks where the size of the search space over which learning is performed may grow exponentially.

The multi-agent learning area is still in development and each algorithm makes its own assumptions in order to cope with specific options so the solution designer must find the algorithm that fits better whereas the researchers should develop new more advanced mechanisms with less assumptions.

3.2 Cooperative or competitive learning?

The presence of other agents introduces the question: are they friends or enemies? We will focus first on some approaches that assume the cooperation and good behaviour of all the agents (friends) in the system because they are the most developed algorithms. This is not a complete survey but it tries to illustrate the state of current research.

In the extreme of fully cooperative algorithms all the agents have the same reward function and the goal is to maximize it. If a central approach is taken then the task is reduced to a MDP.
In (Tan, 1993) they extend Q-Learning to multi-agent learning using joint state-action values. This approach is very intensive in the communication of states and actions (every step) and do not scale well. A similar approach is to let the agents exchange information like in the Sparse cooperative Q-learning (Kok & Vlassis, 2006). It is a modification of the reinforcement learning approach which allows the components not only learn from environmental feedbacks but also from the experiences of neighbouring components. Global optimization is thus tackled in a distributed manner based on the topology of a coordination graph. Maximization is done by solving simpler local maximizations and aggregating their solutions. Other approach is to restrict each agent to use only the information received from its immediate neighbours to update its estimates of the world state (as a contra it could result in long latency and inconsistent views among agents). In MARL there are basically states, actions and rewards so the approaches differ in what the agents share and what is private (what is social and what is personal). The cost of communications should also be taken into account, a framework to reason about it can be found in (Pynadath & Tambe, 2002) with the name of communicative multiagent team decision problem. Another option to reduce complexity is to try to reduce the universe of possible policies. To achieve it we can use a different level of abstraction that allows to construct near-optimal policies as in (Boutilier & Dearden, 1994) or to use explicitly a task structure to coordinate just at the high level of composed policies as in (Makar et al., 2001).
It is extremely difficult to find some algorithm in the literature which is guaranteed to converge to global optimum even in the reduced case of a fully cooperative stochastic game (one example is Optimal Adaptive Learning in (Wang & Sandholm, 2002)). Because of that, we will repeat that heuristics are still welcome to introduce some guide in the policy search (Bianchi et al., 2007) or to bias the action selection toward actions that are likely to result in good rewards.
If we take a look at the competitive learning models, the interest of agents is now in conflict and we can not assume they will do anything to help each other. Not surprisingly, there is a great influence here of game-theoretic concepts like Nash equilibrium. We are limited by now by the low development of the current techniques and many algorithms are only able to deal with static tasks (repeated, general sum games). In the presence of competing agents we hardly have guaranties in scenarios more complex than two agents with two options. If it were not bad enough, competition can result in cyclic behaviours where agents circle about one another due to non-transitive relationships between agent interactions (like rock-scissor-paper game). And, when scaled up to hundreds of agents in stochastic environments with partially observable states and many available actions, all current existing methods will probably fail. Because of that, it is fundamental to research more in this area where we have many open questions still.

4. Some scenarios and current research

Here we will show some developments that introduce improvements in common network tasks including some interesting approaches, even when some of them are not strictly related to learning and few of them are single agent versions. It could be good to notice that managing a complete network involves several topics so we could find useful related work in different areas of research. Our focus is in communication systems but, for instance in the multi-hardware configuration area and electricity networks we can find analogous problems too.

4.1 Specific challenges of multiagent solutions for network scenarios

The usual scenario for multiagents in machine learning is a group of robots doing some task like playing soccer; grabbing cans; or cleaning some rooms. In section 2 we have mentioned some characteristics and how they can affect the possible algorithms, but there are some extra issues that make network a different and interesting environment to apply multiagents techniques. For instance, communication between agents is an important topic in multiagent solution but in a network there is an extra problem: there will be a trade off between coordination messages vs bandwidth available for the system itself because administrative messages will reduce the available resource. In addition, as computer networks are highly heterogeneous environments with different device capabilities, it could be better suited to develop heterogeneous approaches or if we choose to use homogenous algorithm then to take special measures and abstractions. Privacy also would be of great value in scenarios such of those with presence of different telecommunication carriers while they are collaborating. In addition, some agencies are proposing changes in current network regulation and legislations that would change some of our current assumptions. And last but not least, new business models should be introduced by enterprises to survive. Because of all that, we can only try to imagine the difficulties will be posed to these autonomous network management scenarios. We will next mention some of the existing solutions to illustrate how far we are still to fulfil the mentioned challenges.

4.2 Classic network scenarios

The use of learning techniques has been introduced in the network management field several years ago with some initial scenarios of interest related to fault management and allocation tasks in server. The fault management case has two stages: fault detection and fault mitigation. It is important to detect the kind of fault in order to choose the best way of mitigate it or, if we can not mitigate it then to announce it clearly to the system administrator. One of the difficulties to identify the threat is the big amount of alarms that could be triggered by the same problem, so one important issue in this task is the correlation of alarms. It has been solved usually by classification techniques from the datamining field; neural network or belief networks are some options where some engine might find out the most probable cause for a given sequence of alarms. It can be found in (Steinder & Sethi, 2004) a solution to fault localization using belief networks.
We can also find the use of Reinforcement learning applied to automatic network repair in (Littman et al., 2004) where they study its feasibility and efficiency.
Some low level but critical functions which are worth mentioning are how much data send, when to do it, how to route it and how to control the access of new integrants to the network. In the routing approach of (Boyan & Littman, 1994) Q-routing had better results to non-adaptive techniques based on shortest path; it regulates the trade-off between the number of nodes a packet has to traverse and the possibility of congestion. The problem with routing algorithms is that in general do not scale well, usually make some assumptions to improve their performance or if not take millions of trials to converge on a network of just few nodes. There is a trade-off between proactive and reactive routing protocols, one

improves the delay but increase the control overhead or it can decrease the overhead but at the price of not improving as much as before. Although there are many works about learning in routing there is still a constraint in how much time does it takes to converge against the dynamic of the system (the learned path can became inefficient quite fast) so there are several_deterministic techniques with "enough" performance that seems to make unnecessary the overhead of any learned approach in most situations. One of those heuristics is called AntNet (Caro & Dorigo, 1998) and uses swarm intelligence (based on animal behaviour, ants in this case). Their routing scheme adapts to changes in availability and quality of network resources. Each node keeps a routing table with probabilities of reaching known destinations through each outgoing link. Periodically each node sends an agent to a random destination. When the agent returns updates the routing table at each node on the path based on the measured time to reach the destination (latency). A different approach, based on market, could be found in iREX (Yahaya, 2006), an inter-domain scheme for data needing special quality of service on the Internet. Each domain independently advertises and acquires network resources in deploying an inter domain path while decides a price for the links that it owns. The cheapest path with quality assurance is preferred. Another interesting approach, even when there is no learning involved, is the Greedy Perimeter Stateless Routing (GPSR). It is a geographical routing scheme that improves times sending the data to a specific geographical location instead of a destination IP address (Karp & Kung, 2000). This is more efficient because geographical information is more relevant to choose between paths than just IP address.

Some higher level classic problems include task allocation and resource management. They have some similarities because in the end task allocation also is a request for resources. In (Bennani & Menascé, 2005) they propose a solution using analytic queuing network models combined with combinatorial search techniques. They defined a cost function to be optimized as a weighted average of the deviations of response time, throughput and probability of rejection and use predictive multiclass queuing network models, then the global controller algorithm executes a combinatorial search technique over the space of possible configuration vectors where the utility function will guide the choices. The data centre tries to maximize a global utility which is a function of the local utility functions of the various application environments. In the case of resource management tasks chosen measures usual include average job processing times, minimum waiting time for resources, resource usage and fairness in satisfying clients. With an economic approach, the same problem is presented in (Rodrigues & Kowalczyk, 2007) where they propose a price that is adjusted iteratively to find equilibrium between a set of demands and a limited supply of resource (this method was first applied by (Everett, 1963)) they suggest a mechanism to learn the negotiation. In (Abdallah & Lesser, 2006) they create an algorithm that mixes game theory with a gradient ascent learning algorithm for two players with two actions. Its reading is enough to show the hardness on the theoretical proof of this kind of scenarios where any analysis gets too complex too fast.

4.3 Quality of Service

Not all applications running on a network need the same amount and kind of resources to function so, in order to introduce the possibility to express the demand of different services it has been introduced the concept of Quality of Service (QoS) that states the restrictions that are required for a service to work properly. It allows the decoupling of different services and the definition of service oriented architectures where it is possible to choose between providers allowing market based approaches. Going one step further in the concept of quality measures there has been new proposals working with perceived quality instead of the previous model based on low level parameters and it has been coined the concept of Quality of Experience (QoE). For instance in (Ruiz et al., 2004) they introduce a genetic algorithm related to a model of the user-perceived QoS allowing the application to select the new combination of settings which maximizes the user's satisfaction for that particular network conditions. They states that the best for the user could differ if we only focus on the low level parameters, for example there is not a linear relation between the bandwidth and the user-perceived QoS, so the problem becomes now how to model user's perception.

4.4 Service Level Agreement

The negotiation of QoS can be well established between different networks by means of a service level agreement (SLA[6]) so it is worthwhile to study the improvement of this negotiation. For instance (Cosmin et al., 2010) introduces intelligent strategies for contract negotiation based on Bayesian framework.

4.5 Wireless

It is clear we should not limit to wired networks, the research community is also working in wireless sensor and vehicle networks. Those different kind of networks have also introduced some special needs, for instance scheduling data transmission is important because a path may not always be available, it is vital also to know how to construct hop-by-hop connectivity and besides, the network elements have to control the power they consume because they are mostly battery powered so it is a scarce resource. So, in addition to other goals, "survival" is an extra one that needs to be taken explicitly into account. In (Vengerov et al., 2005) they try to solve the power control problem by working in hybrid approaches using fuzzy rules to bring some flexibility in the expressivity of the reinforcement learning.

Historically we have been using the seven layers model from IBM to describe a network from the lower layer of sending/receiving bits to highest application layer, so it is natural to find works like (Legge & Baxendale, 2002) where they design a solution assigning one agent in charge for each layer. Otherwise, a new "wave" proposes that, for wireless scenarios, breaking that modular definition is actually better and some researches have embraced the Cross-Layer point of view (Lee & al., 2007). The idea behind this is that sharing parameters from different layers can increase the efficiency of the algorithms at the price of loosing the modularity of the IBM's model.

The resource management is not a simple task either, in (Shah & Kumar, 2008) they show the trade off between personal vs social interest and propose mechanisms to balance it aligning individual's utility with the global utility. Another related topic can be found in (Wolfson et al., 2004) where they study the dissemination of information about resources and propose an algorithm that attacks this issue. They developed an opportunistic dissemination paradigm, in which a moving object transmits the resources it carries to encountered vehicles and obtains new resources in exchange using economic models to incentive collaboration in the propagation by virtual currency involved.

---

[6]Part of a service contract where the level of service is formally defined.

5. Interesting future research and open questions

At this point we hope it is clear for the reader that finding the best policy with thousand of network elements in a dynamic environment in less than one hour is not possible (with our present techniques) so we need to continue the development with new ideas to solve the current open issues in multi agents learning techniques: how to coordinate, how to scale, and how to manage partial or incomplete information. Some possible options to explore are to define some information that allows us to process the data quicker in order to find a solution in a limited time; or to reduce the complexity and the heterogeneity of the agents. Other approach could be to leave the goal of finding the global optimal policy and use some near optimal concept with a trade-off measure between time to learn and error introduced in the best solution achieved. All that implies that to find a solution to a big problem we need to identify and add meta-information to drive the transformation from the original problem to a less complex task, maybe we could find a way to introduce time restriction in our algorithms. Another path to explore is the fact that existing MARL algorithms often require some additional preconditions to theoretically guarantee convergence. Relaxing these conditions and further improving of various algorithms in this context is an active field of study.

Another issue is that any realistic network will have multiple, potentially competing goals. Then it becomes a multi-objective problem and we need to find members of the Pareto-optimal set, where no single parameter can be improved without a corresponding decrease in the optimality of another parameter (Lau & Wang, 2005). Multi-objective optimization is a well-studied topic with good results for distributed multi-objective genetic algorithms (Cardon et al., 2000). There is still much work to do improving the selection between points of the Pareto frontier and developing more learning algorithms that do not depend only on genetic ones. And, to further complicate things, not only we have multiple goals but they can also be in different levels (or granularity). As it is mention in (Kephart & Das, 2007) an important issue is how to translate from resource level parameters to high level goals such as performance metrics (response times and throughputs) or availability metrics (recovery time or down time). It is key to manage the relationship between those metrics and the control parameters. Their work is interesting and relies on utility functions (although that approach also has some drawbacks at the time to express preferences with more than two parameters). That implies we need to improve the expressiveness of the goal function if we want to continue using utility based approaches. One possible approach is to reduce complexity, maybe less parameters is better than many if we need to reduce the time of learning, so how to choose which are the most important parameters to do the reduction may be another question to research.

Finally, because the great amount of tasks presented in a network environment we need a definition of a common benchmark to test different existing (and future) approaches. If every designer continues defining a new special scenario where his algorithm performs perfect then we will never be able to compare them. We acknowledge that this is not an easy task because the benchmark needs a definition of a “typical” network whereas a network could vary a lot about “typical” characteristics as load and user’s request, being a very heterogeneous scenario. If it is not feasible to define only one benchmark we could create few illustrative scenarios with a clear list of assumptions and justification of their existence.

In this chapter we have stated several tasks that are important to manage a network in autonomous fashion; we have collected disparate approaches to multi-agent learning and have linked them to different networks tasks showing current advances and open questions. We believe significant progress can be achieved by more exchange between the fields of machine learning and network management. From one side, new features of future Internet (and networks in general) will introduce more demands and will boost the development of better algorithms so we expect the appearance of new techniques and from the other side the new techniques developed by machine learning field will allow new network functionalities that will improve its utility and our user's experience in the near future.

6. References

Abdallah, S. & Lesser, V. (2006). Learning the task allocation game, *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, AAMAS'06, pp. 850-857, 2006Hakodate, Japan

Alba, E. (2005). Parallel metaheuristics: a new class of algorithms. John Wiley & Sons, Inc., ISBN: 978-0-471-67806-9, NJ, USA

Bennani, M. & Menascé, D. (2005). Resource allocation for autonomic data centers using analytic performance models. *IEEE International conference on autonomic computing*, Seattle, Washington, USA

Bernstein, D.; Zilbertein, S. & Immerman, N. (2000). The complexity of decentralized control of MDPs, *Proceedings of UAI-2000: The sixteenth conference on uncertainty in artificial intelligence*, 2000, California, USA

Bianchi, R.; Ribeiro, C. & Costa, A. (2007). Heuristic selection of actions in multiagent reinforcement learning. *IJCAI'07*, 2007, India

Boutaba, R. & Xiao, J. (2002). Network Management: State of the art. *Communication systems: The state of the art (IFIP World computer congress)*, Kluwer, B.V., pp. 127-146, ISBN:1-4020-7168-X , August 2002, Deventer, The Netherlands

Boutilier, C. & Dearden, R. (1994). Using abstractions for decision-theoretic planning with time constraints", *Proceedings AAAI-94*. 1994, Washington, USA

Boyan, J. & Littman, M. (1994). Packet routing in dynamically changing networks: a reinforcement learning approach. In *Advances in Neural Information Processing Systems,* Vol. 7 (1994) pp. 671-678

Busoniu, L. ; Babuska, R. & De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning, *IEEE Transaction on systems, man and cybernetics – Part C: Applications and reviews*, Vol. 38, No. 2 (2008) pp. 156-172

Cardon, A.; Galinho, T. & Vacher, J. (2000). Genetic algorithms using multi-objectives in a multiagent system. *Robotics and Autonomous systems*, Vol. 33, No. 3 (2000) pp. 179-190

Caro, G. & Dorigo, M. (1998). AntNet: distributed stigmergic control for communications networks. *Journal of Artificial Intelligence Research*, 9, pp. 317-365, 1998

Cosmin, G.; Dan Şerban, L. & Marius Litan, C. (2010). A Framework for Building Intelligent SLA Negotiation Strategies under Time Constraints. *Economics of Grids, Clouds, Systems, and Services 2010*, Vol. 6296 (2010) pp. 48-61, DOI: 10.1007/978-3-642-15681-6_4

Everett, H. (1963). Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, Vol. 11, No. 3 (1963) pp. 399-417

Guestrin, C.; Lagoudakis, M. & Parr, R. (2002). Coordinated reinforcement learning, in *Proceedings of the 19th International Conference on Machine Learning (ICML-02),* pp. 227-234, 2002, Sydney, Australia

IBM (2001). Autonomic manifesto, www.research.ibm.com/autonomic/manifesto

Karp, B. & Kung, H. (2000), GPSR: Greedy perimeter stateless routing for wireless networks, *MobiCom2000*, August, 2000, Boston, USA

Kephart, J. & Das, R. (2007). Achieving self-management via utility functions, *IEEE Computer Society*, 2007, Vol. 1, Issue 1 (2007) pp. 40-48, ISSN: 1089-7801

Kok, J. & Vlassis, N. (2006), Collaborative multiagent reinforcement learning by payoff propagation, *Journal of machine learning research 7*, pp. 1789-1828, 2006

Lau, H. & Wang, H. (2005). A multi-agent approach for solving optimization problems involving expensive resources, *ACM Symposium on applied computing*, 2005, New Mexico, USA

Lavinal, E.; Desprats T. & Raynaud Y. (2006). A generic multi-agent conceptual framework towards self-management. *IEEE/IFIP Network Operations and Management Symposium*, 2006

Lee, M. & Marconett, D. & Ye, X. & Ben, S. (2007). Cognitive Network Management with Reinforcement Learning for Wireless Mesh Networks, *IP Operations and Management, Lecture Notes in Computer Science*, Vol. 4786 (2007) pp. 168-179, DOI: 10.1007/978-3-540-75853-2_15, 2007

Legge, D. & Baxendale, P. (2002). An agent-based network management system, *Proceedings of the AISB'02 symposium on Adaptive Agents and Multi-Agent Systems*, 2002, London, England

Littman, M. ; Ravi, N. & Fenson, E. et al. (2004). Reinforcement learning for autonomic network repair. *IEEE 1st International Conference on autonomic Computing*, 2004

Makar, R.; Mahadevan, S. & Ghavamzadeh, M. (2001). Hierarchical multi-agent reinforcement learning. *Atents'01*, 2001, Montréal, Quebec, Canada

Papadimitriou, C. & Tsitsiklis, J. (1987). The complexity of markov decision processes. *Mathematics of Operations Research*, vol. 12, nº. 3 (1987) pp. 441-450, USA

Pynadath, D. & Tambe, M. (2002). The communicative multiagent team decision problem: Analyzing teamwork theories and models, *Journal of Artificial Intelligence Research*, Vol. 16 (2002) pp. 389-423

Rodrigues, E. & Kowalczyk, R. (2007). Learning in Market-based Resource Allocation. *6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007)*, 2007, pp.475-482, ISBN: 0-7695-2841-4

Ruiz, P.; Botía, J. & Gómez-Skarmeta, A. (2004). Providing QoS through machine-learning-driven adaptive multimedia applications, *IEEE Transactions on systems, man, and cybernetics – Part B: Cybernetics*, Vol. 34, No. 3 (2004)

Shah, K. & Kumar, M. (2008). Resource management in wireless sensor networks using collective intelligence, *Intelligent Sensors, Sensor Networks and Information Processing, 2008. ISSNIP 2008.* pp. 423-428

Steinder, M. & Sethi, A. (2004) Probabilistic fault localization in communication systems using belief networks, *IEEE/ACM Transactions on Networking*, Vol. 12, No. 5 (2004) pp. 809-822

Strassner, J.; Agoulmine, N. & Lehtihet, E. (2006). FOCALE: A Novel Autonomic Networking Architecture. In *Latin American Autonomic Computing Symposium (LAACS)*, 2006, Campo Grande, MS, Brazil.

Sutton, R. & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press, 1998

Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. *In Proceedings of the Tenth International Conference on Machine Learning*, pp. 330-337, 1993

Tianfield H. (2003). Multi-Agent based autonomic architecture for network management, IEEE, 2003

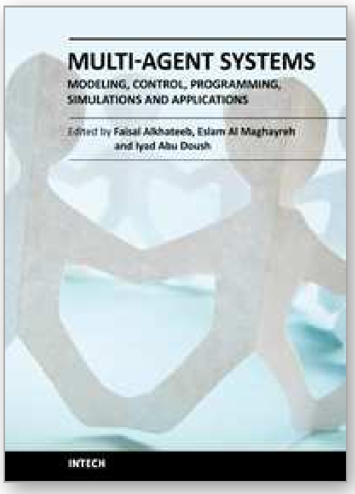
Vengerov, D.; Bambos, N. & Berenji, H. (2005). A fuzzy reinforcement learning approach to power control in wireless transmitters, *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, pp. 768-778, ISSN: 1083-4419, August, 2005

Wang, X. & Sandholm, T. (2002). Reinforcement learning to play an optimal Nash equilibrium in team Markov games, *in Advances in neural information processing systems*, 2002

Weiss, G. (1999). Ed., *Multiagent Systems: A modern approach to distributed artificial intelligence*. Cambridge, MA: MIT Press, 1999

Wolfson, O.; Xu, B. & Sistla, A. (2004). An economic model for resource exchange n mobile peer to peer networks, *Proceedings of the 16th International conference on scientific and statistical database management*, January, 2004

Yahaya, A. (2006). iREX: Efficient inter-domain QoS policy architecture, *in Proceedings of IEEE Globecom*, 2006



Multi-Agent Systems - Modeling, Control, Programming, Simulations and Applications
Edited by Dr. Faisal Alkhateeb

ISBN 978-953-307-174-9
Hard cover, 522 pages
Publisher InTech
Published online 01, April, 2011
Published in print edition April, 2011

A multi-agent system (MAS) is a system composed of multiple interacting intelligent agents. Multi-agent systems can be used to solve problems which are difficult or impossible for an individual agent or monolithic system to solve. Agent systems are open and extensible systems that allow for the deployment of autonomous and proactive software components. Multi-agent systems have been brought up and used in several application domains.