Configuration and Optimization in Self-Managing Networks: a Tutorial

Adam D. Ross

University of Northern Iowa
1227 West 27th Street
Cedar Falls, IA 50614
adross@uni.edu

Abstract

One of the key goals of network administration is configuring a network that is always operating at an optimal state. However, this goal is increasingly time-consuming and expensive as networks increase in size and complexity. One path to offsetting the increasing cost is in the implementation of a network that manages itself – known as a cognitive network. This paper will introduce the concepts of self-configuration and self-optimization as they relate specifically to cognitive networks and more generally to the discipline of artificial intelligence from a non-implementation specific view. A possible model for the structure of a cognitive network will be presented as a result of these concepts. A minimal knowledge of network architecture is required to understand the following material. This paper is intended in part to be an introduction to the newly formed concept of cognitive networks.

1. Introduction

The methods of network configuration have remained largely unchanged since networks first needed to be configured. The physical infrastructure has certainly changed since the first telephone network or the first computer network was configured. The interface for configuration of networks has also changed, but one fact has remained: networks depend on human interaction. A network must be configured by a human at its inception and any time a change is required. Because of this reliance on human intervention, the progression of network management and its associated methodologies have stagnated.

This stagnancy is due almost entirely to the fact that network management is an expensive task. Not only does it cost money to hire skilled network administrators and to purchase management software, but it also takes a great deal of time. To add to this is the fact that changing the architecture or configuration of a network can easily become a highly disruptive procedure, especially when changes are made to critical distribution nodes, such as core routers. Anyone who has ever had to make these changes on a busy network can attest to the task's difficulty.

Even the simple task of adding or removing an endpoint node such as a client computer or cell phone can become expensive in terms of money and time when done on a large enough scale. The financial expense is proven by the exponential increase in cell phone towers or wireless access points within the last decade. Expensive physical additions have been made to networks to accommodate increased traffic in a generally inefficient manner. On top of that, in many cases these distribution nodes overlap and interfere with each other, adding to the difficulty of network management.

The new paradigm of cognitive networks offers a solution to these problems, among others. The majority of work and research on cognitive networks has occurred since early 2006, making this a relatively young concept. There is still debate on the definition of what makes a network truly "cognitive". The first part of this paper will offer a brief definition and description of a cognitive network, based on the research of current domain experts. As the concept itself is far too large for this paper alone, the following sections will focus only on the self-configuration and optimization aspects of cognitive networks. To further narrow the information covered, a somewhat conceptual approach to these elements will be given, rather than an implementation-specific, engineering approach. The paper will conclude with an outlook on self-managing network technologies.

2. Definition

The sudden emergence of cognitive network research is due largely to the advancements made in cognitive radio, pioneered by Joseph Mitola. Mitola designed a system of radio communication that transmitted data according to observations made about the current state of the radio spectrum. The client could view the spectrum's usage and transmit only in unused time slots and frequencies (Mitola, 2000). Because of this monitoring and by keeping a "history" of the spectrum usage, the spectrum could be utilized in the most efficient way possible and thus greatly improve the end-to-end throughput of the medium.

If multiple cognitive radios were to communicate within the same physical and spectral region, the result would be a

much faster and more efficient network than a standard radio network – such as those utilizing 802.11 technologies (Wi-Fi). However, means of transmission alone do not qualify a network as *cognitive*.

The actual definition of a cognitive network is still in debate among researchers and varies among implementations and perspectives. Many definitions dwell on implementation specifics such as infrastructure platforms and signal transmissions. However, often times the simplest definition is the best. Therefore, this paper will use the definition formed by (Clark, 2003): "[a cognitive network is one] that can assemble itself given high level instructions, reassemble itself as requirements change, automatically discover when something goes wrong, and automatically fix a detected problem or explain why it cannot do so." The underlying message here is that a network must be able to manage itself – without human intervention such as that of a network administrator.

This definition can be extended by that of (Thomas, 2006): "Ideally, a cognitive network should be forward looking, rather than reactive, and attempt to adjust to problems before they occur." This definition states that a cognitive network should be able to recognize the signs that something detrimental to the operation of the network is about to occur (best case scenario) or is currently occurring and should then be able to remedy the situation. This facet of cognitive networks would prove to be very important in mission critical networks such as that of a factory production network, a hospital communication network, or a battlefield tactical communication network. This is something that the current network architecture paradigm simply cannot accomplish without a degree of intelligence or cognition.

2.1 Requirements

There are a few criteria that a network must achieve to be considered cognitively self-managed (Lu, 2007). The two following criteria form the basis of any cognitive network and can be summarized to the two aspects of a self-managed network that this paper focuses on: self-configuration, and self-optimization, respectively. Summarized, the two core criteria are as follows:

- The network must properly prioritize conflicting policies and/or goals. It then must be able to translate these policies and goals into the configuration of the network without human intervention.
- The network must also be constantly working towards an optimal state of operation. It is not enough for a network to simply "work", but it must be able to either obtain its full operational potential or conform perfectly to predefined business goals. No network is perfect, but many are limited to sub-optimal operation due to lack of time, money, or experience on the part of the administrator.

There is a third criteria, self-healing, which deals with how a network handles purposely incorrect configurations, but it is beyond the scope of this paper. The remainder of this paper will describe in more depth both of these aspects, and how each aspect utilizes artificial intelligence to achieve their respective tasks.

3. Self-Configuration

Network configuration is not a trivial task. As the size of the network increases, the complexity and administration time required to manage the network increases exponentially. This effect is compounded by the administrative differences in the involved platforms. The logical and physical infrastructure of today is vastly different than that of ten years ago due to increases in speed, the addition of wireless access points, increased VPN usage, and a proliferation of "non-PC" clients (such as smart phones, Voice Over IP phones, and even videogame consoles) all accessing the same network. One could argue that due to the flexibility and inclusive nature of standardized network technologies (TCP/IP, Ethernet, Wi-Fi), administrative complexity is inherent in the design of networks.

Disregarding the physical configurations of network devices (since, if they are able to connect to a given network, they are obviously transmitting and receiving similarly), the main administrative differences lie in how a router is configured differently than a switch, or more drastically, how a distribution device is configured differently than the client devices (such as a laptop computer or a smart phone) that access it. Even within similar devices there are drastic configuration differences.

Take, for example, the configuration interfaces for a Cisco router versus a Juniper router - these two vendors offer two distinct and different command sets, via the Cisco IOS and the Juniper Junos OS. If a network administrator wants to configure one or both of these, then it becomes apparent that there are different commands and logical progressions to memorize for each device. The two routers may also have their own proprietary protocols that can be configured, but this incompatibility will limit the operational set of actions when the device is in a mixed environment rather than all-Cisco or all-Juniper. These differences and limitations, if left unchecked, could at best, result in a sub-optimal network or at worst, a network that simply "does not work".

A properly functioning network therefore requires one or more administrators skilled in the configuration specifics of every device on the network. Furthermore, the administrator(s) must be able to learn and master any new technology that becomes integrated into the network – most of which with unforeseen differences. This is simply not a feasible solution with the current networking paradigm. However, artificial intelligence can be of great

help here.

A diverse network would benefit from one or more configuration ontologies. An ontology can "provide a set of formal mechanisms for defining a set of definitions for facts, as well as a rich set of relationships between those facts" (Strassner, 2007). A configuration ontology could therefore provide the mappings between a configuration action, such as disabling an interface, and its respective implementations on various network devices.

For example, a network administrator could issue a command to set up a trunk between two network switches of different manufacturers and with two distinct command-sets (such as the previously stated Cisco and Juniper operating systems). The administrator would issue a high-level command that has a direct match in the configuration ontology, such as `createTrunk(Switch_A, Switch_B)`. The cognitive network then utilizes the configuration ontology to issue two separate commands that correspond to the high-level request – the equivalent IOS command to the Cisco switch, and the equivalent Junos command to the Juniper switch. This results in a trunk between two distinct devices – and the administrator did not have to understand either device's command set. The true potential of a cognitive network can be expressed in this example when the administrator issuing configuration requests is replaced with an intelligent software agent, which will be discussed later.

The concept of a configuration ontology is a key component of a self-configuring network. However, the ontology by itself is useless as it only provides a relationship between high-level instructions and vendor-specific commands. A cognitive network needs to have some knowledge of cause and effect to properly utilize the configuration ontologies. It needs to have a correlation between the high-level commands found in the ontologies and the results of these commands' execution as it pertains to the state of the network. For this, a knowledge base must be created.

The cognitive network's knowledge base will consist of the cause and effect relationships between a configuration command and its effect on the network. For example, a simple rule in the knowledge base would be that if a router interface is disabled, the network load on that device will decrease and CPU cycles will be conserved. A more complex rule may be that if a wireless access point (WAP) detects the presence of other WAPs in its transmission range, it will increase the power distributed to its transceiver in order to decrease range and increase bandwidth speed. Another set of rules may govern how to automatically set up an ad-hoc Bluetooth network when a group of smart phones are within range of each other.

Due to the incredibly large number of actions the collection of devices in a network can perform and the nearly infinite number of situations a device can be expected to perform in, cognitive network knowledge

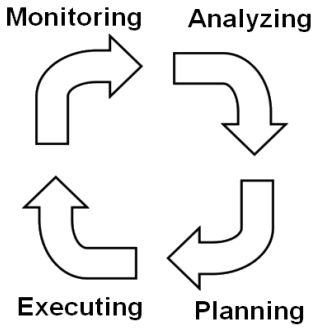

Figure 1. The Cognitive Process

bases can become quite large. One solution to this is to break up the entire network's knowledge base into multiple knowledge bases with one for each device located on that device. These devices would then trade their knowledge with each other in a process known as knowledge routing (Clark, 2003). The knowledge routing process creates its own "knowledge network" that parallels the operation of the communication/data network and is beyond the scope of this paper.

Although how a knowledge base is distributed and stored is one concern, its rule set is of greater concern. How does a cognitive network "know" that its rules are actually beneficial to the overall operation of the network? The previously mentioned example of a router shutting down interfaces to conserve CPU cycles benefits the router itself, but it does not benefit the network if there are nodes relying on those interfaces for communication. Imagine a core router shutting down all of its interfaces because its CPU is at a 99% load! The result would be disastrous – especially in a mission critical network. This is where the next aspect of a self-managing network comes into play: optimization.

4. Self-Optimization

Cognitive networks provide a stark alternative to conventional networks. While the configuration of conventional networks change irregularly, such as with distribution device upgrades and additions, cognitive networks are in a continuous state of change and self-optimization. This is achieved through the use of what is known as a cognitive process, also known as a cognitive cycle.

The cognitive process is the core of the intelligence of a cognitive network. Without it, a network cannot self-manage. The cognitive process can be described as a four-stage loop, consisting of monitoring, analyzing, planning, and executing stages [see figure 1] executed in this order.

The cognitive process gives direction to what is known as the cognitive manager. A cognitive manager is generally implemented as an intelligent software agent that interacts with a series of cognitive components that will be discussed in greater detail later (see Figure 2). More than one cognitive manager can be present in a network at a

time – in fact, this allows for greater control over the network. A cognitive manager will generally manage a single network element, such as a router, a link between two devices, or a VPN. However, there is a certain trade off between the optimality gained from the actions of the manager and the overhead caused from its operation. Because of this, the placement of these managers is of much debate between cognitive network implementations.

In a network performing at an optimal state, the majority of the cognitive process activity can be expected to occur in the monitoring and analyzing stages as no changes are required and the network's optimality needs to be continuously reassessed.

Once the network is considered sub-optimal, the planning stage prepares for a configuration change by checking its knowledge base to see what series of actions it can perform to improve the network's optimality. It then carries out these actions in the Executing stage. This might be as simple as disabling a noisy and unnecessary port on a switch or as complicated as a series of configuration changes to brace against a Distributed Denial of Service (DDoS) attack. These two examples illustrate not only the versatility of the optimization process, but also the incredibly ephemeral nature of a network's optimality. A network that continues to carry out its normal functions in the face of a DDoS attack is by no means operating at the optimal state for which it was designed, but it is doing the best it can for the current situation. Due to situations like these, the cognitive process is presented with opportunities to achieve its next facet of self management, learning.

4.1 Learning

One of the main reasons a cognitive network is continuously striving for optimality is that it never knows what the "optimal" state of the network is. There isn't a set of guidelines saying something like "the network is optimal when the link between A and B is utilizing 37% of its bandwidth and device A is transmitting at 100% of its top speed". None of the agents or knowledge bases ever know what defines the "true" optimal state of the network. They know at most what the business policies or rules consider to be an optimal state.

These rules and policies are handled by the policy manager (Jennings, 2007). The policy manager can best be viewed as an entity that determines the desired goal of a network as pertaining to the current situation by accessing a policy database. The policy manager for a small home network may consider reliability and low power consumption of distribution nodes to be its normal policy/goal, but whenever someone wants to set up a video conference with someone on the Internet, the policy will switch to maximum throughput from the client out to the Internet, regardless of power consumption. Another example would be an ISP that provides a Service Level Agreement (SLA) to a business which states that they

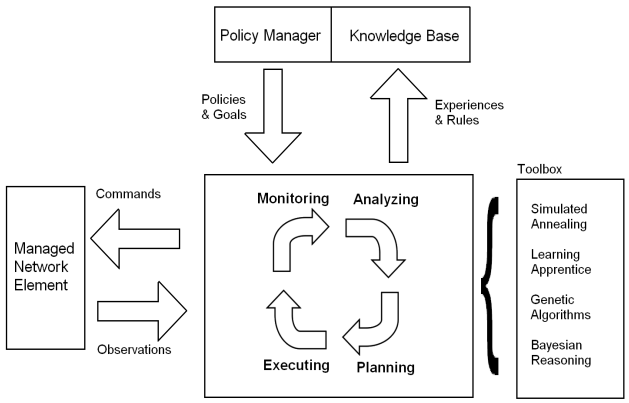

Figure 2. The components of a cognitive manager. Adapted in part from (Mahonen, 2006).

a lower bandwidth in case of an abnormality or disaster.

The policy manager provides the goal that the network's cognitive process should constantly be working towards. However, in order to reach that goal, the cognitive process needs to understand which actions in its knowledge base actually help it get closer to that goal when executed. To do this, the network must be able to learn.

At the core of the learning process of a cognitive network are entities known as sensors (Clark, 2003). Sensors provide observations about the network element they are managing, such as a router or trunk between two network switches. Sensors do all of the work of the monitoring stage of the cognitive process. They monitor and relay a set number of attributes of a managed element back to the cognitive process, such as interface utilization, CPU utilization, operating state of the interface (up or down), neighbor nodes, and many others.

With this enormous amount of data about the network in hand - gathered from all of the sensors – the cognitive manager compiles this information into a current state of the network. That state is then compared against the global policies and goals from the policy manager to determine what needs to be changed in order to reach an optimal state.

The cognitive manager then sets out to determine what actions from its ontologies would bring the current network state to the optimal state by first checking for any rules in the knowledge base. If a rule is found, the necessary command(s) are pushed out to entities known as actuators (Clark, 2003), that reside on the managed element in order to perform the executing stage of the cognitive process. The cognitive process then restarts by having the sensors check the new state of the element and relaying it back to the cognitive manager. You will notice that the learning process here is synonymous to the cognitive process. This should not be surprising as the cognitive network is in a continuous state of learning.

This is a relatively straight-forward process if a rule can

be found in the knowledge base. However, if there are no rules for the given situation, the cognitive manager will have to find a way to create one itself. This can be done in a handful of ways, with the learning apprentice method (Dietterich, 2007) arguably being the best for knowledge bases with few rules.

The learning apprentice, also known as an adaptive interface, is a system where the cognitive manager "watches" a human network administrator manage and configure a network. It does this by presenting the administrator with a list of possible configuration commands for a given situation. The administrator can then choose the best option, or reject all of them and execute an entirely different command. Much like a human apprentice, after watching enough configuration tasks, the cognitive manager should be able to perform administration without the aid of a human.

This of course implies that the human truly knows the best options and consequently sets a bar for the cognitive manager's capabilities. A cognitive manager trained by an inept human will itself be inept at best. The learning apprentice system isn't too far off from explicitly writing out rules into the knowledge base. Therefore, both the explicit creation of rules and the learning apprentice system create a limited knowledge base that may not be able to handle anomalous network conditions. The cognitive manager can, however, continue to learn through other methods.

A cognitive manager should have at its disposal a toolbox of algorithms and methods to help it learn new rules (see Figure 2) such as genetic algorithms, bayesian reasoning, and simulated annealing. Simulated annealing can help the cognitive network learn new facts when presented with a limited knowledge base due to its capability in dealing with a large set of variables, such as those returned by the sensors (Mahonen, 2006). By utilizing simulated annealing, coupled with a weighting system that considers one network state better than another, a cognitive manager issues different configuration commands, checks the resulting state, and adds successful rules to the knowledge base accordingly.

As the knowledge base increases in size, so does the versatility and speed at which a network can achieve its goals of optimality. This section covered the learning aspects of the optimization of a cognitive network, but these methods could also be applied to other cognitive network aspects, such as creating a configuration ontology using the learning apprentice or (somewhat awkwardly) simulated annealing.

5. Conclusion

The concepts in this paper are mostly just that – concepts. However, a few projects have begun to explore the implementation of these concepts, such as the FOCALE (Foundation, Observation, Comparison, Action, and Learning Environment) system created by Motorola Labs (Strassner, 2007). At the time this paper was written, the FOCALE system is the closest any network has come to being fully cognitive. The major entities now working on cognitive networks are Motorola Labs, universities, and the ANA (Autonomic Networking Architecture) group. The ANA group has also created a detailed specification of the routing, monitoring, and resilience aspects of their Autonomic Network Architecture (Tschudin, 2006), which has provided the basis for many cognitive network research projects.

This paper provided a definition for self-managing networks known as cognitive networks. The concepts of self-configuration and self-optimization were described in terms of their associated intelligent aspects.

Although the concept of a cognitive network is still in its infancy, the technologies behind it will provide a drastic change for the current paradigm of network architecture or at the very least will forever change the way networks are configured.

References

Clark, D., Partridge, C., Ramming, J., & Wroclawski, J. (2003). A Knowledge Plane for the Internet. *ACM SigComm 2003.*

Dietterich, T., Langley, P. *(*2007). Machine Learning for Cognitive Networks: Technology Assessment and Research Challenges. In Q. H. Mahmoud (Ed). *Cognitive Networks: Towards Self-Aware* Networks. (pp. 97-119). Chichester, West Sussex: John Wiley & Sons Ltd.

Jennings, B., Meer, S., Balasubramaniam, S., Botvich, D., Foghlu, M., & Donnelly, W. (2007). Towards Autonomic Management of Communications Networks. *IEEE Communications Magazine, Vol. 45, No. 10.* 112-121.

Lu, J., Egashira, R., Fujii, K., Yahaya, A., & Suda, T. *(*2007). Adaptive Networks. In Q. H. Mahmoud (Ed). *Cognitive Networks: Towards Self-Aware* Networks. (pp. 53-76). Chichester, West Sussex: John Wiley & Sons Ltd.

Mahonen, P., Riihijarvi, J., & Wellens, M. (2006). Cognitive Wireless Networks: Your Network Just Became a Teenager. *IEEE InfoCon 2006.*

Strassner, J. (2007). The Role of Autonomic Networking in Cognitive Networks. In Q. H. Mahmoud (Ed). *Cognitive Networks: Towards Self-Aware* Networks. (pp. 23-52). Chichester, West Sussex: John Wiley & Sons Ltd.

Thomas, R., Friend, D., DaSilva, L., & MacKenzie, A. (2006). Cognitive Networks: Adaptation and Learning to Achieve End-to-End Performance Objectives. *IEEE Communications Magazine, Vol. 44, No. 12.*. 51-57.

Tschudin, C., Jelger, C., Bouabene, G., Leduc, G., Lorenzo, P., et al. (2006). ANA Blueprint. http://www.ana-project.com