A System Perspective on Cognition for Autonomic Computing and Communication

Arjan Peddemors[♦], Ignas Niemegeers[*], Henk Eertink[♦] and Johan de Heer[‡]

[♦]*INCA group* / [‡]*AIMS group, Telematica Instituut, The Netherlands*
{Arjan.Peddemors, Henk.Eertink, Johan.deHeer}@telin.nl

[*]*WMC group, EWI Faculty, TU Delft, The Netherlands*
I.Niemegeers@ewi.tudelft.nl

Abstract

In this paper we present a conceptual view on the incorporation of cognitive processing capabilities in future generation computer systems. We argue that cognition is at the heart of autonomic behavior, and therefore a necessary ingredient for autonomic computing and communication. We introduce a bio-inspired cognitive engine that interacts with and has control over major operating system components, and showcase, based on scenario descriptions, how communicating applications take advantage of this setup by adapting and autonomously reacting to changes in heterogeneous and widely varying network resources. As a conceptual work, this paper does not cover experiences with the implementation of the introduced concepts, nor does it describe experimental results.

1. Introduction

In the emerging domains of autonomic computing and communication, many characterizing features such as self-awareness, self-(re)configuration, striving for optimization, self-recovery and healing are hard to realize with the current technologies we use for building computer systems and software. In fact, the characteristics that are distinctive to autonomic computing are characteristics that we know from biological organisms only. In nature, they are almost always related to *cognition*, i.e. the use or handling of knowledge including such aspects as sensation, perception, awareness, reasoning and judgment. For biological organisms, cognitive capabilities are prerequisite to capabilities such as problem solving, self-awareness and attention – many of the characteristics we also wish to realize in autonomic computing systems. So, taking this prerequisite as a hint from nature, we most likely need cognitive processing capabilities to implement autonomic behavior in computer systems. This makes artificial cognitive processing functionality a central element for autonomic computing.

In nature, higher order cognitive processing takes place inside the human brain as well as brains of certain animal groups. Much research has been dedicated over the years on how to mimic parts of the functionality of the brain with Artificial Neural Networks (ANNs). In terms of computer architecture, neural networks depend on massively parallel processing by simple interconnected nodes, which is quite different from the Von Neumann model used by the majority of contemporary computers. Limited attention has gone to integrating ANN technology with computer operating systems. In theory, an ANN may do all processing on a computing device, but we consider this a too dramatic departure from the computing model we are currently used to. However, we also realize that bio-inspired cognitive processing on a computer system requires an ANN runtime.

In this paper, we therefore propose a hybrid approach, with an ANN-based cognitive engine that is part of a more traditional computer system and that has control over most parts of the operating system. We expect that this system perspective with cognitive processing in the centre supports us in realizing autonomic computing applications.

The motivation for this work lies with our objective to understand and support the adaptation of applications on mobile devices towards changing network resources – the kind of dynamic environment typical for mobile devices using heterogeneous network technologies. We realize, however, that the conceptual view presented here has a much broader scope and applicability and covers multiple research domains. We believe that it provides the groundwork for building autonomously adapting applications, not relying on limited sets of static adaptation rules but

Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA'05)
1529-4188/05 $20.00 © 2005 **IEEE**

capable of optimizing and learning even in situations not anticipated at design time.

This paper is organized as follows. In section 2 we discuss a number of aspects relevant to system support for cognition. Section 3 lists the guiding principles we use for the formulation of our system perspective, and section 4 provides the outline of this cognitive system perspective. In section 5, we show a number of network related application scenarios that benefit from our approach. We wrap up with conclusions and future work.

2. Towards System Support for Cognition

When taking biological organisms as inspiration, we have to find the basic principles and mechanisms that build cognition in nature and map them, in a suitable manner, to the substrates and tools we use for building computer systems. This does not necessarily mean mimicking, with silicon and software, every physical process taking place inside a brain cell, it 'merely' means the identification of those functions and processes that are necessary for cognition. As with heavier-than-air flight: our current – successful – aircraft technology relies on the principle of aerodynamic lift, but does not mimic every other common aspect of animal flight such as wing flapping.

Biological cognition is based on the *connectionist* principle, where mental activity can be represented by activity inside networks of interconnected units (neurons). It is generally accepted, although to date not shown to realize higher levels of mental capabilities in artificial systems, that the algorithm describing the operation of a single neuron is relatively simple, while the network of neurons as a whole can express very complicated, even complex, behavior. This 'complex behavior with simple rules' is in stark contrast with the way we currently build computer applications (as illustrated in [6], although the topic itself is older). Any non-trivial software program needs many lines of code, and the implementation effort, i.e. the programming of software, intuitively is proportional to the complexity of the program's functionality. This means that highly complicated functionality, such as higher order cognition, is very complicated to build using traditional software engineering techniques. With neural networks, a part of the effort is shifting from *rule making* (programming) to *learning*; typically, a neural network only shows the desired behavior after the network is sufficiently trained. Indeed, the training effort may be substantial, possibly even comparable or higher than the coding effort. So, by applying the connectionist principle to computing machinery, we believe it is possible to realize complicated functionality with simple rules, at the expense of learning.

Cognitive capabilities only seem to develop and grow in the event of rich sensory input combined with sufficient actuator capabilities given resource and processing limitations and environmental constraints. To learn about the nature of an artifact, for instance, it is important that an animal or human can pick it up, observe it from different sides, feel, smell, etc.: the actuators position the artifact in such a way that the sensors detect relevant data (see also [3]). Likewise, cognitive computer systems must make sure that the cognitive processing parts receive input that is meaningful for the tasks at hand, and can steer or influence actuators in such a way that the input is enhanced, i.e. becomes more meaningful. Obviously, sensors and actuators in computer systems may be quite different from those found in nature. For instance, to learn about the availability of wireless networks at the current location, the system may instruct the wireless network interface to start scanning for networks with certain intensity, and decrease the intensity, after a while, when the list of known networks does not change.

The research on ANNs has traditionally focused on relatively small networks, by far not approaching the billions of neurons (10^{11}) found in the human brain. We believe, however, that size is highly important, especially when aiming at higher order cognitive functions. So far, practical limitations have made it difficult to build ANNs in silicon and simulate them with software. In many cases, it is harder to build or simulate large networks when the individual neurons more precisely mimic the operation of brain neurons. These may turn out to be conflicting requirements: on one hand the individual neurons must mimic their biological counterparts in enough detail to reach a cognitive process, on the other hand building or simulating large networks may force neuron models to be as simple as possible.

Large and multi-dimensional ANNs may show, similar to their biological counterparts, an organization that has many cyclic connections between sets of neurons. These recurrent networks show a dynamic behavior that is mostly determined by the feedback loops, much in the same way as with traditional control systems (see [3]). This means that the overall organization of interconnects may influence the ANN characteristics more than the characteristics of single neurons.

In more recent years, a new neuron model that explicitly takes into account the timing of inputs has gained increasing attention. These so called Spiking Neural Networks (SNNs) represent input and output as series of spikes or pulses, similar to the biological

neuron. One advantage of this model is that neurons do not have a continuous output that needs to be propagated to others with every time step of, say, a simulation, and that spiking, generating a pulsed output, can be relatively sparse. These characteristics make them suitable for large size network simulations on standard PC equipment [2]. Developments in this direction, together with the increasing processing capabilities of standard computer equipment indicate that the incorporation of large ANNs in computing systems may be feasible, now or in the near future. Ultimately, ANNs will run much more efficiently, perhaps with much larger sizes and less power consumption, on dedicated silicon [5]. This would mean a departure from the Von Neumann architecture, at least for the cognitive processing part of the computer system.

3. Guiding Principles

Considering the discussion in the previous sections, we adhere to the following guiding principles when formulating our cognitive system architecture:

- We take a bio-inspired approach when it comes to the implementation of artificial cognition. This ultimately means running very large ANNs on massively parallel hardware, but for the short term means applying moderate size ANNs with a runtime (simulated) environment on general-purpose hardware.
- Given the large paradigm shift when going from a Von Neumann architecture to a connectionist architecture for computing, see [4], we prefer a hybrid system that builds on our existing knowledge and experience while applying new principles to open problems. This offers the possibility to adjust proportions depending on the tasks and depending on future characteristics of then available hardware. For instance, at some time in the future, massively parallel hardware may be readily available which can support the ANN in taking over many functions executed by traditional hardware. An additional benefit comes from the observation that the different architectures are complementary: it's hard for a human being to quickly calculate 7251 times 4.89, while it's hard for traditional computer systems to anticipate on change, make correlations, and deal with unexpected situations. Hybrid systems are capable of merging functionality from both domains.
- The traditional part of the hybrid system must be setup in such a fashion that the cognitive processing part is capable of receiving information on the operational status of as many subsystems and components as possible and capable of controlling and influencing the operation of these elements. The cognitive processing part should have rich sensory input and plenty of means to adjust the sensor characteristics or influence through other actuators the data reaching the sensor. Given limited processing resources and the cost of processing associated with obtaining sensory data, it must be capable of reducing the sensing and sensory input processing when not necessary for current tasks. As indicated in the previous section, sensors and actuators on computer systems can be radically different from the ones found in nature. We expect that the design of the sensors and actuators contribute to a large extend to the overall quality and level of the cognitive capabilities of a system.
- Detailed input on the current operational status of the system allows the cognitive processing part to watch other processes on the system performing tasks. This should enable learning by observation, for instance, learning from a traditional application that is communicating with external entities, watch how it uses system components such as network stacks and network interfaces. Note that this is not the same as invention, where the cognitive processing part tries out, more or less randomly, to steer system resources until something useful comes out. Obviously, humans often learn by watching others.

4. Contours of a Computer System with Cognitive Features

This section provides the outline of a computer system architecture that incorporates facilities supporting cognitive processing. Being a hybrid architecture, it consists, on the hardware side, of elements found in common contemporary computer systems such as a CPU, memory, hardware for I/O. Additionally, it may be equipped with hardware means for massively parallel processing, to support connectionist tasks. The architecture's central entity is the *cognitive engine*, offering an environment for the cognitive processes that are active (see Figure 1). The cognitive engine is a large ANN that controls all tasks taking place on and executed by the system.

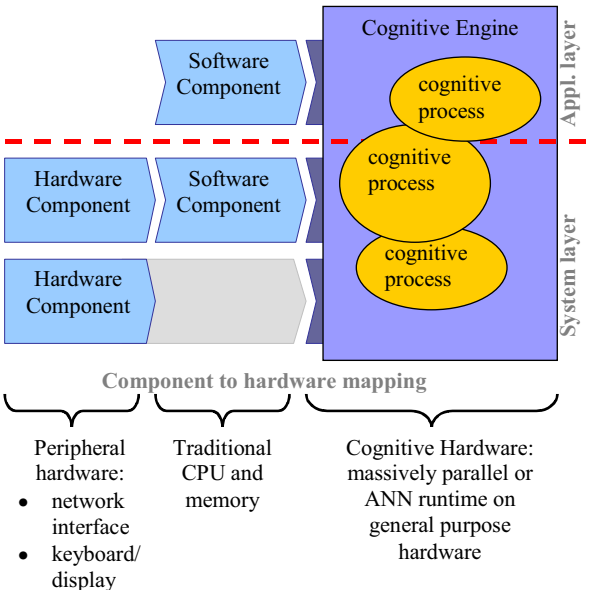

Figure 1: subsystems and components of a computer system with cognitive features

The cognitive engine interacts with the traditional system components, which could be software as well as hardware components. The example in figure 2 shows a media player that projects a video stream on the display of the computer. The cognitive engine obtains information on the operational status of all components that are relevant to the player, through interfaces between the 'clocked' world of the traditional components and the 'continuous' world of the ANN. Furthermore, it influences the operation of the components, for instance by tuning UDP, IP, and network interface parameters to reduce packet loss.

When new tasks are assigned to the system, and it is extended with extra components (software or hardware), the cognitive engine will first have to adjust to the new configuration. Different tasks, or rather different cognitive processes, may overlap and influence each other.

5. Experiments and Applications

In this section we consider two application scenarios, both dealing with data communication: one with a focus on network usage from a single computing device, the other dealing with interaction between systems within the network infrastructure.

In the domain of autonomic computing, an interesting interaction takes place between the computer system and the data network it is using for communication with other parties. Both the computer (or perhaps, applications running on the computer) and the network may show autonomic behavior and potentially influence each other's adaptation. Furthermore, clusters of computer systems may form autonomic systems themselves, using the data network for internal communication. Indeed, the network infrastructure itself can in many cases be regarded as consisting of computer system nodes cooperating to deliver data from one end point to another.

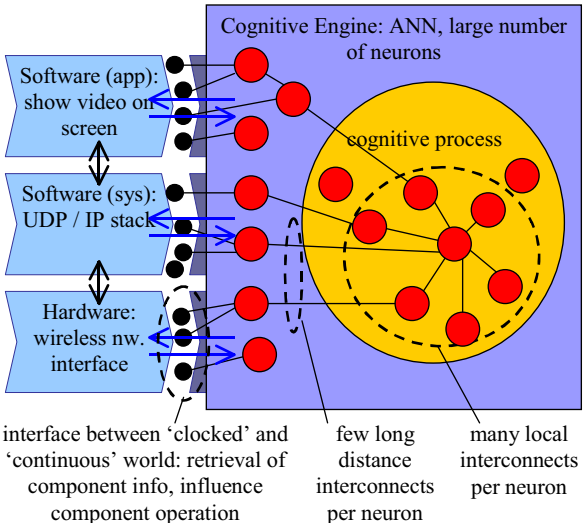

Figure 2: interaction between cognitive engine and traditional system components

The *first application* is an audio streamer, for instance as used in a Voice-over-IP setting, that runs on a mobile device (computer) and learns to direct the outgoing media stream over the best network path. The device is equipped with multiple wireless network interfaces, which offers multiple communication paths to the application, depending on availability of networks. For the sake of the experiment, the receiving endpoint supplies feedback to the sender in the form of a quality indication at application layer. The outgoing audio stream is artificial: it has elements corresponding to different frequency domain: low, middle, and high. The quality indication metric is artificial as well, although based on actual human perception: mid tones contribute more to the overall quality than the low and high tones.

The streamer is now learned to optimize its own operation, using the feedback of the receiver. It has control over all elements having a part in the communication: application decisions such as dropping low or mid tones, system settings such as UDP and IP parameters, as well as hardware settings such as 802.11 WLAN parameters: a true cross-layer optimization.

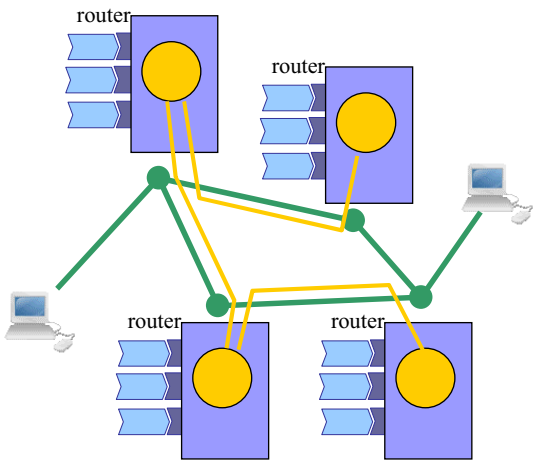

Figure 3: cognitive engine interaction between network routers

The *second application* deals with the Knowledge plane [1]. Here, the routers in a network infrastructure are computer systems based on the architecture presented in the previous section. They are configured and learned in such a way that the cognitive engine exchange information about the overall status of the network infrastructure (see figure 3). Together the individual routers form an artificial organism responsible for keeping the data network in the best possible condition.

6. Conclusions and Future Work

We have presented a view on the incorporation of cognitive processing capabilities in a computer system in order to support autonomic computing and communication. Our perspective is based on biology inspired artificial cognition running as an ANN controlling all other system components.

We realize that many aspects have not been covered or specified in detail. Amongst those are:

- So far, large ANNs showing higher order cognitive capabilities have not been build. The examples from nature show it can be done, but we're not there yet.
- It is hard, at this point in time, to estimate the efforts necessary to learn the ANN as presented in our setup.

Offering a conceptual view, this paper does not cover implementation aspect. We plan to fill in details of our architecture, supported by experience from running systems. Additionally, we want to apply the system in situations where applications need to be adaptive towards changing network resources.

7. Acknowledgement

This research has been supported by the Dutch Freeband Communication Research Programme (AWARENESS project) under contract BSIK 03025.

8. References

[1] D. Clark, C. Partridge, J. Ramming, and J. Wroclawski, "A Knowledge Plane for the Internet", In *Proceedings of the ACM SIGCOMM*, August 2003

[2] A. Delorme and S. Thorpe, "SpikeNET: An Event-driven Simulation Package for Modeling Large Networks of Spiking Neurons", *Network: Computation in Neural Systems*, Vol. 14, No. 4, pp. 613-627, 2003

[3] H. Ritter, J. Steil, C. Nölker, F. Röthling, and P. McGuire, "Neural Architectures for Robot Intelligence", *Reviews in the Neurosciences*, Vol. 14, No. 1-2, pp. 121-143, 2003

[4] L. Stein, "Challenging the Computational Metaphor: Implications for How We Think", *Cybernetics and Systems,* Vol. 30, No. 6, pp. 473-507, 1999

[5] P. Tosic, "A Perspective on the Future of Massively Parallel Computing: Fine-Grained vs. Coarse-Grained Parallel Models", In *Proceedings of the ACM Conference on Computing Frontiers*, April, 2004

[6] S. Wolfram, "A New Kind of Science", Wolfram Media, 2002