Pursuing an Always-on Connected Network Architecture through INTERA

Leandro Carvalho, Alexandre Passito, Saulo Jorge, Laércio Péricles, Edjard Mota[1]

[1]Departamento de Ciência da Computação
Universidade Federal do Amazonas (UFAM)
Av. Gen. Rodrigo Octávio Jordão Ramos, 3000, Coroado I
69077-000 – Manaus – AM – Brazil

{galvao,passito,sjbq,lpbj,edjard}@dcc.ufam.edu.br

Abstract. *It is expected for the next years that any nontrivial device will contain some degree of embedded processing and communications capability. In this new environment, the focus shifts from individual devices to an "environmental user interface," acting as a contextual user access and information delivery engine across multiple interconnected devices. As an intermediate step to reach this goal, we present the Intelligent Routing Architecture (INTERA), whose core is a logical router responsible for checking network status and determining the best route(s) for packet forwarding. From INTERA concept, we present some challenges to be tackled in order to achieve fully dependable, scalable, and ubiquitous computer systems.*

1. Introduction

Consider a business person (either from a private company or government), participating in a webcast video conference with her/his computer device (PC, laptop, blackberry, or another one) equipped with two or more network interface cards (NIC). Let's call it a *node*. As the conference goes by she walks around the building, leaves one hot spot connection to another and sometimes the connected NIC fails but not the service. She decides to go home taking company transport and points to the running aplication (perhaps using command voice), what device she will be using next. She gets her Internet Tablet over her table, keep participating in the web event while she leaves the other device on stand by and go to the parking laud. All other open services (email, a text document, etc) also "migrate" to her tablet. She arrives at home, seats at the living room balcony to have a tea while she follows the last questions and even herself addresses one. As she closes the application she starts now getting messages related to her home business (bills, community meetings, family issues, etc.).

All this time she was, according to our view, *always-on connected* to her business no matter which NIC the application was using, where she was located and even which device she was handling. As a matter of fact not even to which business she was connected since there was an immediate context switch when she logged off from the conference. Despite the many issues this visionary networking raises, at least one is certainly most fundamental and supports all other services to allow this kind of always-on connected network: how to keep a given application service (like `http`, `ftp`, or `sip`) running with no interruption whenever its correspondent NIC goes down (because of link failure, or because the user is actually wandering around or even changing device) and there is at

least another operational NIC which can maintain this service? In other words, what do we have to do in order to seamlessly maintain a given session of the application layer when the physical and link layers are switched?

Behind this question lies many challenging issues that are currently under study in many areas of knowledge. Basically, the intention is to design context-aware devices and networks capable of seamlessly, ubiquitously, and autonomically change their configuration according to changes in the environment or user needs. Although a collaborative and multidisciplinary task forces is needed to integrate solutions to all challenges, computer networks must be provided with intelligent and distributed capabilities, based on flexible hardware and operating systems, continuously and autonomically tuning themselves in order to not be noticeable to their users.

In this paper, we introduce the Intelligent Routing Architecture (INTERA), as means to achieve that high-level goal. If we regard this "logical router" as an agent, i.e. "a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives" [Wooldridge 2002], we have to provide it with self-management features (aka self-configuration, self-optimization, self-healing, and self-protection). On the other hand, as every router requires a public IP address, if every node is likely to have an agent that behaves as such, a scalability problem grows at the very first glance. Furthermore, there is the routing table creation/update and consulting problem, whose execution time increases as the number of nodes grows, and the problem of maintaining the user context as it is moving along different and diverse network providers.

Our intention in presenting INTERA is two-folded. First, we think that high level architectures, designed to offer a real user-centric service, must count on reliable routing, and this means that modules of the operating system should have to pro-actively work on the behalf of its user. Second, based on the discussion about INTERA, we shall present and comment the main challenges to be tackled to go further in the quest for reaching "dependable, scalable and ubiquitous systems".

The rest of the paper is structured as follows. In Section 2, we describe the Linux Ethernet Bonding Driver [Davis et al. 2007], a kernel module responsible for gathering multiple ethernet interfaces into a single logical interface. By extending the logical interface functionalities so it can act as a logical router, we conceived the Intelligent Routing Architecture, presented in Section 3. In Section 4 we address other challenges that along with INTERA are the cornerstones of the always-on connected network in the scenario described above, and we sketch a likely roadmap to overcome this grand challenge. Finally, our concluding remarks and some directions for future work are drawn in Section 5.

2. Aggregating NICs – Related Work

The Linux Ethernet Bonding Driver (Bonding Driver for short) was initially proposed by Donald Becker's and is reported in [Davis et al. 2007]. The basic idea behind it is to provide node's NICs as slaves for a logical interface. The Bonding Driver presents two ways of detecting failures on a link of a given slave NIC: Address Resolution Protocol (ARP) and Media Independent Interface (MII).

The ARP monitor philosophy is similar to that of a link state routing protocol. It sends ARP queries to one designated peer systems on the network and uses the response

as an indication that the link is operating. In turn, MII only monitors the carrier state of a slave NIC. The updating of carrier state can be provided in one of the following alternatives: by device driver, by querying the device's MII registers or by making an *ethtool* query to the slave NIC.

The Bonding Driver also offers three policies to controlling frame transmission by slave. The first one is *Round-robin*, in which the packets are transmitted in sequential order, from the first available slave through the last one. The second is *Active-backup*, where just one slave is used until a failure happens and another slave will be used if, and only if, the current slave fails. Finally, there is the *Broadcast* policy, in which every frame is transmitted on all slaves. A comprehensive understanding of the techniques above described as well as more details about Bonding Driver can be found in [Davis et al. 2007].

With respect to our visionary networking scenario, in spite of Bonding Driver can be configured "for high availability" in a given node, as suggested by its manual, it fails with respect the following reasons. Firstly, it works only when all NICs belong to the same Autonomous System (AS). Also, if a new NIC is inserted into the node, it is necessary to (manually) configure the Bonding Driver to communicate with it. Secondly, the Bonding Driver treats all NICs evenly, not considering peculiarities of each medium channel for an optimum packet forwarding. Thirdly, the Bonding Driver can only tell to its node whether a given NIC or link is down, but not if the next hop is unable to forward the packets. Finally, the Bonding Driver can not compute whether the cause of a link being down because is due to a physical failure or to a denial-of-service attack, among many other security reasons.

Initiatives similar to the Bonding Driver have emerged, such as Multi-radio Unification Protocol (MUP) [Adya et al. 2004] and Wireless Bonding (WBond) [Kim and Ko 2007]. However, in general, they are only throughput driven, i.e., they are concerned with maximizing the throughput by making use of multiple NICs and do not solve the problem which we have presented too.

Multi-radio Unification Protocol (MUP) was proposed to optimize local spectrum usage via intelligent channel selection in a multihop wireless network. MUP periodically selects the best NIC among those available for forwarding data. The criterion to select the best NIC is the round trip time (RTT), by which a MUP-enabled node sends a probe message to its neighbors. After receiving a probe message, neighbor nodes send back an acknowledgement packet containing their timestamp. When receiving acknowledgment messages, the node that sent the first probe message can calculate the RTT and then determine which one of its NICs has better quality. The MUP has the limitation that the maximum throughput is bounded to the maximum throughput of the best NIC.

In [Kim and Ko 2007], authors point some limitations of current IP-based network architecture such as: the maximum throughput of one application is limited to the throughput of one NIC and a busy NIC can be selected to transmit traffic even if other NICs are not busy. The authors also argue that this situation can go on unless the connection is lost. In order to overcome these limitations, Kim and Ko [Kim and Ko 2007] proposed the Wireless Bonding (WBond) architecture.

According to [Kim and Ko 2007], WBond is located below the IP layer and above layer 2 of a node. It creates a virtual interface that connects and manages the multiple

NICs. Thus, a WBond-enabled node is set with only one IP address for multiple NICs and applications. Assuming that two nodes of a network have a set of NICs whose channel currently in use is same, the WBond distributes packets from an application by all available interfaces of a node. A NIC with better channel quality is more likely to be selected. Thus, the maximum throughput achieved in WBond is the maximum throughput of a node, which is the throughput achieved by using all available interfaces.

Similarly to Bonding Driver, both MUP and WBond can not ensure uninterrupted services as we described it in Section 1. Despite MUP coordinates the operation of multiple network cards, it focuses on optimizing spectrum usage of a local node and, consequently, MUP is limited to wireless NICs, in particular those which are compliant with IEEE 802.11 standards. WBond, in turn, presents the same limitations of Bonding Driver because it is essentially based on this solution, as pointed out by [Kim and Ko 2007].

Next, we present our Intelligent Routing Architecture (INTERA), which extends Bonding Driver functions to perform layer 3 tasks, such as packet forwarding and routing. This brings new concerns beyond Bonding Driver, like routing protocols, network addressing and cross-layer design, which are addressed in Section 4.

3. Extending the Bonding Driver – the Intelligent Routing Architecture

Increasing the abstraction level towards the scenario described in the Introduction, we need to build an intelligent agent inside computer devices, capable of performing network layer tasks. It should be aware of both MAC/PHY layer status and applications needs, and use this knowledge to accomplish high-level objectives, such as maximizing throughput, saving power, and, above all, allowing an alays-on connected service. In the present section we describe the Intelligent Router, the core of the Intelligent Routing Architecture. It focuses the relationship among OS, hardware, application and environment, giving support for service-oriented autonomic architectures that can be built over it.

3.1. The Intelligent Router

In order to achieve the always-on connected network, we should have a Bonding drive-like agent endowed with intelligent routing capabilities. It would act as an intelligent and virtual router embedded within the node hardware. In this case, the extended Bonding Driver, would not only be capable of checking link status, but also to computing route costs and quality, behaving pro-actively in order to keep its node always-on-line anywhere, anytime. Additionally, this intelligent router, should be able of routing packets among NICs belonging to different autonomous systems (AS), which is not performed by the Bonding Driver itself.

One may think about this intelligent router (IR) as being a software process placed between the Operating System and the NICs, as depicted in Figure 1. It extends Bonding Driver functions, because it can determine routes, check their status and redesign routes in case of failures. So, we define a “node” as being a hardware device equipped with multiple NICs (wired or wireless) and running multiple applications over an Operating System. As virtualization is a common place in today’s Internet, our node may or may not have virtual machines running in it.

Figure 1 shows the evolution of our concept from a node with bare NICs (a), to a node having a simple Bonding Driver aggregating the multiple NICs (b), and finally to a

node capable of determining its own routes to its target destinations (c), independently of the status of underlying access networks.

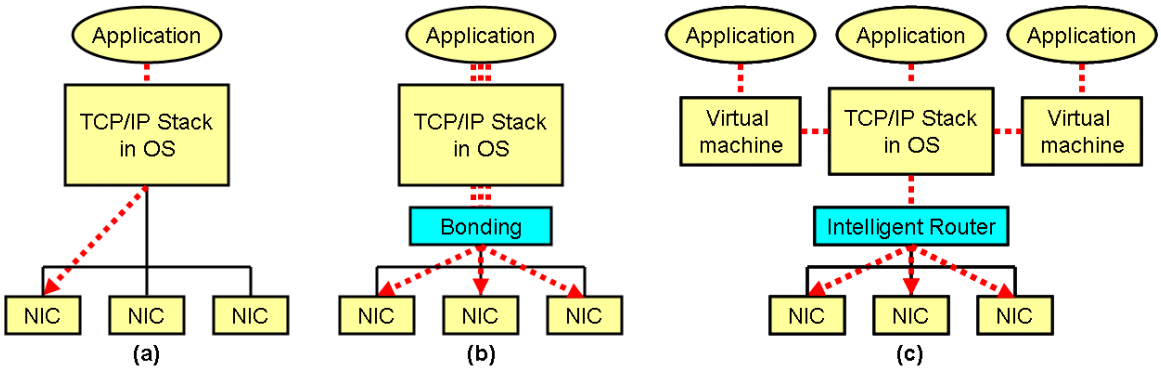

Figure 1. A comparison among (a) a traditional multi-interface architecture, (b) a bonding layer inserted in the architecture, and (c) the proposed intelligent routing architecture.

The diagram sketched in Figure 1(c) introduces a new vision of networking. On one hand, we have a small network virtualized inside one physical device. On the other hand, as every node has a router within itself, all the network can be regarded as an *hybrid Mesh network*. Thus, we should be able to benefit from Mesh network research experience to implement routing protocol schemes inside the IR.

The overall picture is that we have several nodes equipped with different access technologies (WiMax, Wi-Fi, cellular, ethernet, sensor networks, and so on). The NICs of the same physical device are aggregated by an intelligent router (IR), whose role is to keep applications running in the device always connected. A given node can be fixed or mobile, can be a dedicated router or a single client with an embedded IR. From this perspective we can say that nodes equipped with IR form a hybrid Mesh network.

However, we can not claim that INTERA is only a Mesh network regarded in another way. INTERA also proposes the remodeling of the Operating System (OS) so all NICs can really be managed as one resource: network connection, regardless the PHY/MAC layers. Having several decentralized nodes performing routing tasks on their own in a worldwide scale will require a huge management effort and extra bandwidth for control messages exchange. Initiatives such as Autonomic Computing [Kephart and Chess 2003] and Autonomic Communications [Dobson et al. 2006] are taking their first steps to specify a fully self-managed system. A self-managed system is understood as presenting all self-configuration, self-optimization, self-healing and self-protection properties [Kephart and Chess 2003].

Within Autonomic Communications context, the node OS have to be remodeled to support the above self-behavior properties, so network protocols can work efficiently and securely accomplish their goals *through* the OS, and not *besides* the OS. Thus, extending the Bonding Driver is only the first modification to be performed in the OS. Intelligent agents must be designed and implemented inside the OS to support cross-layer message exchange and deal with foreign intelligent agents.

3.2. Why intelligent routing

As computer devices are increasingly present in our everyday life, and we depend on computer systems to perform many tasks, from paying bills to controlling power plants, they must always be available and fault free; they must function as specified and be scalable and safe [Brazilian Computing Society 2006]. Autonomic Communications is a holistic, goal-orientated approach to designing computer networks and/or network services for accomplishing such high-level requirements [Lin et al. 2005]. By "goal-orientated and holistic approach" we refer to the fact that Autonomic Computing *is not a conventional computer systems project* which can be planned, designed and implemented entirely by a single developer, but a visionary approach where a collection of existing technologies has to be grouped together to accomplish a common goal [IBM 2006, Lin et al. 2005].

Considering that, we situate Intelligent Routing Architecture as a supporting step toward dependable, scalable and ubiquitous computer systems. Embedding IRs inside computer nodes will enable the deployment of autonomic communication services with hardware and OS support. Without INTERA concept, many autonomic computing projects, such as *Autonomic Service Architecture* (ASA) [Cheng et al. 2006], and *Foundation, Observation, Comparison, Action, and Learning Environment* (FOCALE) [Strassner et al. 2007], will reach a bottleneck at the hardware/software interface, because they are service-oriented, and do not focus the relationship among OS, hardware and user applications. In other words, their solutions focus on high-level layers, assuming that lower machine levels and network layers are working properly.

Another reason for adopting intelligent routing is because it aggregates the connection potentialities of multiple interfaces under an intelligent agent that acts on behalf of the applications running in the node. Its routing functionality allows it to evenly negotiate with internetworking devices in order to minimize the effects of congestion and link failure to the end nodes.

Consequently, this infrastructure paves the way for the implementation of protocols that really gives support to fault tolerance, permanent availability and scalability. The whole network will present these characteristics only if its devices and nodes are ready to switch traffic among its interfaces while keeping connectivity. Several criteria can be used for choosing the best interface(s): throughput maximization, minimal connectivity level, the least power consumer, and so on.

As Autonomic Communication does not specify which types of technology have to be used to achieve pervasiveness and self-management, INTERA (and any other autonomic proposal) is able to drive solutions based on a merging of partial results of existing fields and most of their future achievements. Areas like artificial intelligence, distributed systems, Mesh networks, peer to peer applications, operating systems, cross-layer design, security, handover, and so on [Lin et al. 2005]. As it will be seen in the challenges addressed in the next section, INTERA aims to convene and foster collective research efforts of such different areas to achieve actual dependable, scalable and ubiquitous computer systems.

4. Challenges to Tackle

Now, we shall set some remarks about the challenging cornerstones that need to be investigated in order to endow the present Internet with autonomic capabilities, considering

the suggestion of bringing the router inside a node as an intelligent agent of the Operating System, as stated above in the description of INTERA. Latter in this section, we sketch a likely roadmap to overcome these challenges and achieve fully dependable, scalable, and ubiquitous computer systems.

Addressing: it is very well known that IPv4 addressing scheme has scale problems. A short-term solution was the development of Network Address Translation (NAT) protocol, but it also imposes scale limits to private networks, and many applications and services have problems to traverse it gently, as is the case of handover procedure. On the other hand, IPv6 [Deering and Hinden 1998, Forouzan 2007] has not been adopted by the majority of network operators, although it has extraordinarily increased the availed network address range. Considering that INTERA main purpose is to make every node act pro-actively by taking over routing functions, and that routing protocols try to build forwarding tables based on unambiguous network addresses, addressing number range problem is brought once again to the spotlight.

There are many proposals in the literature to solve this challenging issue, such as IPv6 [Deering and Hinden 1998] and clean-slate design [Calvert et al. 2007]. For instance, the later one suggests no naming or addressing hierarchy, and no central naming authority. [Calvert et al. 2007] suggests that an user should be located by an EID (Endpoint IDentification), an end-channel identifier by which an user can send and receive packets. By this approach, an EID identifies an application entity independent of its location, and should be randomly selected from a large space.

On the opposite side of clean-slate design, we have the evolutionary research paradigm [Dovrolis 2008]. It does not present an addressing or routing scheme, but it shows some design guidelines, arguing that core protocols play the role of "evolutionary kernels", meaning that they are conserved so that complexity and diversity can emerge at the lower and higher layers.

Route discovery: this is an important challenge faced, for example, by sensor and wireless mesh networks. Following the scheme proposed by [Calvert et al. 2007], route discovery problem should be separated from route-selection. Different levels of abstraction could to be provided by means of *channel based routing*, i.e., only channels are named and nodes remain anonymous. Where necessary, nodes can be identified either as individuals (physical nodes) or as clusters of many levels (conceptual nodes). Figure 2 shows an example of three different abstraction levels. The best view of the topology (whether blurry or detailed) depends on some factor such as physical distance between IR and observed nodes, or logical fragmentation for trouble-shooting purposes.

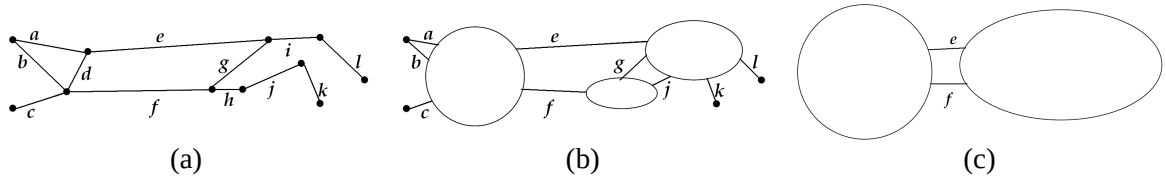

Figure 2. Node names can be abstracted in different levels, depending on the resolution (detail) required by the IR: (a) no abstraction (b) one level of abstraction (c) two levels of abstraction. Source: [Calvert et al. 2007].

Path selection, in its turn, is based on IR policy, which should pursue a balance between application needs (e.g. low delay for multimedia services) and available routes (e.g. for multimedia sessions, routes composed of terrestrial links should be preferred over those ones based on satellite links). Constraints such as battery-power level and/or account/billing issues should be also taken in account as boundary conditions in path selection.

Moreover, as long as the trend for the next years is the coexistence of many and diverse mobile devices with embedded processing and communications capability, merges and splits of self-managed networks will be a common place. Additionally, nodes will not have the same routing protocols, processing capabilities and application requirements. Therefore, the IR should be designed to perform selection among of the most appropriate routing protocol depending on environment (e.g., what protocols are used by neighbor nodes, or how dynamically the network topology changes) and application requirements (multimedia sessions are not latency-tolerant but can withstand some packet loss, unlike instant message service) [Jun and Julien 2007, Legendre et al. 2004].

Routing-table look-up: since traditional hierarchical routing is weakened by INTERA, address aggregation isn't efficient. Mesh routing protocols could be used within some neighborhood from a given node and outside this range, hierarchical routing could be applied.

Self-configuration: Developing self-management capabilities is not a straight-forward road. So INTERA development will have to be done in incremental steps, where the autonomic potential will be increased as long as the maturity level increases [Parashar and Hariri 2007, IBM 2006]. Particularly for self-configuration, the IR shall obtain capabilities that enable to adapt itself to unpredictable conditions by automatically changing its configuration, such as adding or removing a NIC, or installing a new patch of implemented routing protocols, in accordance with high-level policies representing what is desired to keep INTERA working properly.

Self-healing: the IR shall be enabled to automatically discovering, diagnosing, circumventing, and recovering from issues that might cause service disruptions. For instance, switching the packet flow from a NIC in failure to a better NIC is a first step in this direction. But it is not enough to provide a full self-healing system. Research effort among AI and computer network management is needed to design and implement such systems.

INTERA components will detect, diagnose, and repair localized problems resulting from bugs or failures in network software and hardware. Using knowledge about the system configuration, a problem-diagnosis component would analyze the information from network log files. The component would then match the diagnosis against known software patches (or alert a human programmer if there are none), install the appropriate patch, and retest.

Self-optimization: IR shall be enabled to continuously tune itself pro-actively to improve on existing processes and reactively in response to network conditions. The actions of monitoring channel quality and distributing traffic among NICs to increase throughput are some of self-optimization behaviors that the IR can be implemented with.

INTERA components will continually seek ways to improve their operation, identifying and seizing opportunities to make themselves more efficient in perfor-

mance or cost. The components will monitor, experiment with, and tune their own parameters and will learn to make appropriate choices about keeping functions or outsourcing them. They will pro-actively seek to upgrade their function by finding, verifying, and applying the latest updates.

Self-protection: IR shall be enabled to detect, identify, and defend itself against viruses, unauthorized access, and denial-of-service attacks.

Self-protection could also include the ability for INTERA autonomic elements to protect themselves from physical harm, to defend themselves against large-scale, correlated problems arising from malicious attacks or cascading failures that remain uncorrected by self-healing measures. They also will anticipate problems based on early reports from sensors and take steps to avoid or mitigate them [Parashar and Hariri 2007].

Cross-layer design: as INTERA autonomic elements must keep track of information from different network layers or, even, promote the communication between them, its important to INTERA protocols to be designed with cross-layer methodology, i.e., creating new interfaces between layers, redefining the layer boundaries and joint tuning of parameters across layers, in order to improve the performance of INTERA components. There are some approaches in literature to utilize cross-layer information to to attain some of the above self-behaviors [Razzaque et al. 2007, Sadler et al. 2005, Gu et al. 2005].

Operating System kernel support: it is not our ambition to contribute to micro-kernel $\times$ monolithic debate and propose improvements approach for operating systems (OS) design. However, any autonomic system, such as INTERA, will need to exchange messages with the OS kernel, so it can act pro-actively. Part of such need will be supplied by the cross-layer design, where layers implemented in the OS kernel will provide these messages. Another part of this need will be supplied by an intelligent agent as a component of the OS, which will have to adapt its behavior to changing runtime conditions. Researches have been carried out to investigate the benefits of applying Aspect Oriented Programming (AOP) in the implementation of system kernels [Engel and Freisleben 2005, Yanagisawa et al. 2006]. The suitability of using AOP to provide better OS configurability without increasing the costs of CPU usage and memory overhead was demonstrated to be promising by [Lohmann et al. 2006, Lohmann et al. 2007].

An AOP implementation of an OS brings better evolvability to the OS which is a basic property to support autonomic systems, because changes at kernel source code level by refactoring methods can be avoided. However, some challenges still have to be tackled like current AOP language limitations to deal with low level implementation of concerns which requires additional control over the resulting machine code semantics. Another path to go is how to integrate dynamic aspect approach with a system structure. This may lead to a need to use a microkernel-based structure rather than a monolithic kernel.

As far as INTERA is concerned we plan to experiment the use of a kernel level aspect implementation as most NIC do interact at this level. The use of KLASY [Yanagisawa et al. 2006] seems a natural way to go.

Application support: web browsers, softphones, video applications and so forth should have an agent embedded to their code in order to communicate with OS and/or IR agents and benefit from their support for self-* capabilities.

Context handover: some key terms, such as *ambient intelligence* [Ed Thompson 2006] and *omnivalent computing* [Brazilian Computing Society 2006], have been coined in the last years to cover all concepts that relate ubiquity, dependability and evolution of all kinds of computational systems. The main idea behind these concepts is to design user-centric systems, where services can be provided seamlessly across the changes of networks and/or terminal devices, allowing users to maintain access to their contents even while moving, or changing terminal devices and network providers [Kawarasaki et al. 2004].

Traditionally, the term "handover" refers to the change of the *physical* channels involved in a call while the state of the call is supported. However, the challenge here is the seamless continuation of content viewing. For instance, the IR, as proposed in Section 3, intends to perform handover at layer 2/3; Session Initiation Protocol (SIP), in turn, provides seamless session mobility across terminal/network changes. But it does not suffice to reach an actual intelligent and omnivalent environment.

For example, in the case of secure access (e.g., internet banking) or billing, a context handover is needed. The challenge here is to keep an user session seamlessly while its device changes the environment or even though the user changes her/his device. This corresponds to a shift from designing individual interfaces for devices to creating a proactive user interface framework for the environment [Ed Thompson 2006]. Similarly to IR, applications will have to implement one or more autonomic agents in order to keep context while the user moves among different networks or even devices.

Authentication, authorization, accounting (AAA): INTERA must implement functions which provide protection and access control of the network resources. These architecture functions must verify the identity of an agent entering the network and manage the trust and privacy among stranger users that are interacting at some moment (authentication) [Dobson et al. 2006], must verify whether a requesting agent will allowed to access to a resource provided by INTERA (authorization) and collect information on resource usage for the purpose of capacity planning, auditing, billing and cost allocation of the architecture (accounting).

It would be attractive if each of these issues could be dealt with independently, but unfortunately they are closely related [Dobson et al. 2006]. For example, routing involves measuring the network (as also as self-healing and self-optimization), interacting with untrusted nodes, querying authorization and accounting permissions, and rely upon well-tuned Operating System kernel and cross-layer stack. Thus, a coherent solution can only be achieved by weaving efforts in each research issue, aiming an optimum performance of the whole architecture.

As consequence, the roadmap to fullfil a ubiquitous scenario as that described in the Introduction can not be linear, but rather branching, where the branches should be interconnected according to milestones. Figure 3 depicts this idea. Self-* capabilities and cross-layer design are shown as driver philophy behind and among designing every piece of software/hardware in charge of some network function (all layers). Suporting every layer, we have autonomic operating systems across different devices which the user may access. A given group of devices composes the environment where the user is placed in. Above all, we have the context handover, which can make use of underling resources to

provide an always-on connected experience.

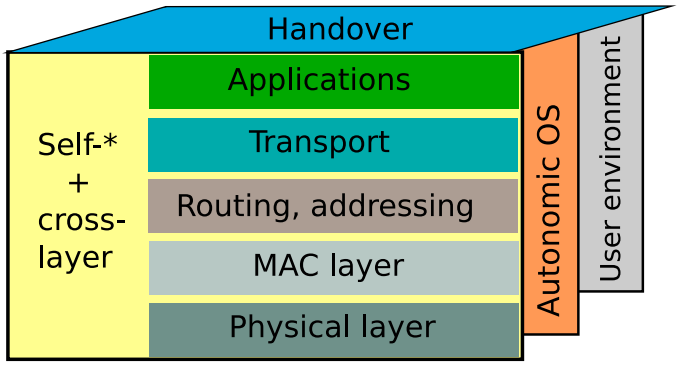

Figure 3. Relationship among the described cornerstones.

5. Concluding Remarks

The notion of dependable, scalable and ubiquitous systems arose as a response to the popularization of small, cheaper and multiple-interface computer devices. Many approaches in the literature try to reach this notion by focusing only protocol deployment or system redesign, neglecting the interaction of network protocols with Operating System, attaining to a comparted vision of the overall computer system. The present work proposes an holist vision of the problem through the Intelligent Routing Architecture (INTERA).

INTERA goes deeper in decentralized philosophy of the Internet: the own nodes (computer devices equipped with multiple NICs) can perform routing tasks, working proactively on behalf of themselves or neighbor nodes in order to maintain seamless and ubiquitous services, at least from application layer perspective. This vision makes it possible to develop more robust and intelligent systems, where the users can move from one network to another, or from one physical device to another, without losing connection.

Additionally, taking INTERA as an intermediate step, we have the notion of context handover, which really provides a ubiquitous and always-on connected environment for the next-generation computer systems. Its deployment will not follow a linear schedule, but rather a branching one, according to milestones defined by current technology's state-of-art. To accomplish this major goal, both academia and industry will have to work in a multidisciplinary way in order to tackle several challenges, such as addressing, routing policies and protocols, self-behavior properties, Operating System kernel support, context handover, AAA, and so on.

References

Adya, A., Bahl, P., Padhye, J., Wolman, A., and Zhou, L. (2004). A Multi-Radio Unification Protocol for IEEE 802.11 Wireless Networks. *IEEE International Conference on Broadband Networks (Broadnets)*.

Brazilian Computing Society (2006). *Grand Challenges in Computer Science Research in Brazil – 2006 - 2016*. Available at `http://sistemas.sbc.org.br/ArquivosComunicacao/Desafios_ingles.pdf`.

Calvert, K., Griffioen, J., and Poutievski, L. (2007). Separating Routing and Forwarding: A Clean-Slate Network Layer Design. *Proceedings of the Sixty-Ninth Internet Engineering Task Force*.

Cheng, Y., Farha, R., Kim, M. S., Leon-Garcia, A., and Hong, J. W.-K. (2006). A generic architecture for autonomic service and network management. *Computer Communications*, 29(18):3691–3709.

Davis, T., Tarreau, W., Gavrilov, C., Tindel, C. N., and Girouard, J. (2007). *Linux Ethernet Bonding Driver mini-howto*. Available at `http://www.kernel.org/pub/linux/kernel/people/marcelo/linux-2.4/Documentation/networking/bonding.txt`.

Deering, S. and Hinden, R. (1998). Internet Protocol, Version 6 (IPv6) Specification. *Request for Comments (RFC) 2460*.

Dobson, S., Denazis, S., Fernández, A., Gaïti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., and Zambonelli, F. (2006). A survey of autonomic communications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(2):223–259.

Dovrolis, C. (2008). What would Darwin think about clean-slate architectures? *SIGCOMM Comput. Commun. Rev.*, 38(1):29–34.

Ed Thompson (2006). *Strategic Technologies for 2006 and 2016*. Gartner Group presentation, Available at `http://www.slideshare.net/guest8e5bf1/plenum-edmund-thompsonpdf/`.

Engel, M. and Freisleben, B. (2005). Supporting autonomic computing functionality via dynamic operating system kernel aspects. In *AOSD '05: Proceedings of the 4th international conference on Aspect-oriented software development*, pages 51–62, New York, NY, USA. ACM.

Forouzan, B. (2007). *Data Communications and Networking*. McGraw-Hill, 4th edition.

Gu, X., Fu, X., Tschofenig, H., and Wolf, L. (2005). Towards self-optimizing protocol stack for autonomic communication: initial experience. *Springer Lecture Notes in Computer Science (LNCS)*, 3854(1):183–201.

IBM (2006). *An Architectural Blueprint for Autonomic Computing*. Available at `http://www-1.ibm.com/industries/government/doc/content/resource/thought/278606109.html`, 4th edition.

Jun, T. and Julien, C. (2007). Automated Routing Protocol Selection in Mobile Ad Hoc Networks. In *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC)*, pages 906–913, Seoul, Korea.

Kawarasaki, M., Ooto, K., Nakanishi, T., and Suzuki, H. (2004). Metadata Driven Seamless Content Handover in Ubiquitous Environment. *2004 Symposium on Applications and the Internet (SAINT'04)*, 00:287.

Kephart, J. O. and Chess, D. M. (2003). The Vision of Autonomic Computing. *IEEE Computer Society*, 36(1):41–50.

Kim, S.-H. and Ko, Y.-B. (2007). Wireless Bonding for Maximizing Throughput in Multi-Radio Mesh Networks. *Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops(PerComW'07)*.

Legendre, F., de Amorim, M. D., and Fdida, S. (2004). Some Requirements for Autonomic Routing in Self-organizing Networks. In *First International IFIP Workshop on Autonomic Communication (WAC 2004)*, volume 3457, pages 13—24.

Lin, P., MacArthur, A., and Leaney, J. (2005). Defining Autonomic Computing: A Software Engineering Perspective. In *ASWEC '05: Proceedings of the 2005 Australian conference on Software Engineering*, pages 88–97, Washington, DC, USA. IEEE Computer Society.

Lohmann, D., Scheler, F., Tartler, R., Spinczyk, O., and Schröder-Preikschat, W. (2006). A quantitative analysis of aspects in the eCos kernel. *SIGOPS Oper. Syst. Rev.*, 40(4):191–204.

Lohmann, D., Streicher, J., Spinczyk, O., and Schröder-Preikschat, W. (2007). Interrupt synchronization in the CiAO operating system: experiences from implementing low-level system policies by AOP. In *ACP4IS '07: Proceedings of the 6th workshop on Aspects, components, and patterns for infrastructure software*, page 6.

Parashar, M. and Hariri, S. (2007). *Autonomic Computing: Concepts, Infrastructure, and Applications*. CRC Press.

Razzaque, M. A., Dobson, S., and Nixon, P. (2007). Cross-Layer Architectures for Autonomic Communications. *Journal of Network and Systems Management*, 15(1):13–27.

Sadler, C., Chen, W., and Kant, L. (2005). Cross-Layer Self-Healing in a Wireless Ad-Hoc Network. *U.S. Patent #11/114.846*.

Strassner, J., Raymer, D., and Samudrala, S. (2007). Providing Seamless Mobility Using the FOCALE Autonomic Architecture. In *Next Generation Teletraffic and Wired/Wireless Advanced Networking, 7th International Conference (NEW2AN)*, pages 330–341, St. Petersburg, Russia.

Wooldridge, M. (2002). *An Introduction To Multiagent Systems*. John Wiley & Sons, Chichester, England.

Yanagisawa, Y., Kourai, K., and Chiba, S. (2006). A dynamic aspect-oriented system for OS kernels. In *GPCE '06: Proceedings of the 5th international conference on Generative programming and component engineering*, pages 69–78, New York, NY, USA. ACM.