♦ Intelligence-Ready Network Infrastructure: An Ecosystem to Control Third-Party Intelligence Distribution Close to Nomadic Users

Paolo Fogliata and Marco Torquato Mussini

The continuous creation of new services that require customized traffic relationships—blending different services into a combined, coordinated customer experience—is driving the emerging architecture proposals for today's converged networks. In the networking business model, relationships between the roles of network operator and service provider are increasing in complexity and flexibility because of evolving regulation. This paper analyzes these issues and describes the architectural guidelines for a "blended" operational model, where the network provider maintains an ecosystem feeding an intelligence-ready communication infrastructure that can host mobile service provider–designed software modules offering personalized functionality to individual subscribers. The network provider can use its knowledge of user location to distribute and run service provider software modules close to end users. The architecture decouples the tasks of designing software for personalized functions and controlling software distribution across the network to users accessible through a variety of wired and mobile infrastructures. © 2008 Alcatel-Lucent.

Introduction

News, weather, and traffic reporting services have often been cited as examples of the emerging mobile computing paradigm where location awareness (in the terminal) and user profile knowledge (in the network or in the application servers) make it possible to deliver customized content [3]. Although from a user's perspective such services are unlikely "killer" applications, to work, they nevertheless require cross-layer information exchange and some distributed logic if the service is to be provided by the network infrastructure [6]. Today, this type of service can be implemented quite effectively by relying mostly on terminals and application servers and using the network infrastructure only as a dumb bit pipe as shown in **Figure 1.** Another even more challenging example is the delivery of journey-relevant information, such as news reports and advertisements, to vehicles on the basis of not only their position, but also their direction and final destination. Again, this can be achieved by relying on the onboard navigation system and, to some extent, on radio broadcasts with the network serving as a dumb bit pipe.

While many value-added applications can be designed to offer services on top of the network infrastructure [14], it definitely makes sense for the network to host some service-specific logic [10]. This is

Bell Labs Technical Journal 13(2), 105–116 (2008) © 2008 Alcatel-Lucent. Published by Wiley Periodicals, Inc.
Published online in Wiley InterScience (www.interscience.wiley.com) • DOI: 10.1002/bltj.20306

Panel 1. Abbreviations, Acronyms, and Terms

CPU—Central processing unit
HTTP—Hypertext Transfer Protocol
MADK—Mobile actor development kit
MCG—Managed computing grid
Opcodes—Operation codes
OPNDC—Open platform for network-distributed computing
PDA—Personal digital assistant
PERL—Practical extraction and report language
SIP—Session Initiation Protocol
TDM—Time division multiplexing
VoIP—Voice over Internet Protocol
XML—Extensible Markup Language

particularly beneficial when the service is not limited to asynchronous, best-effort delivery of small text or voice messages to a limited number of subscribers, and under the following conditions:

- A service has a large number of subscribers and "rich" content, leading to massive aggregated bandwidth requirements, and the need to rationalize content distribution to exploit network resources better, especially when there are significant bottlenecks in link capacity. This content may have components that are statistically delivered to multiple users, creating caching opportunities [8].
- In a business scenario the network infrastructure provider and the service content provider are decoupled organizations that need to share location, user, or session-related information to deliver profiled content to a mobile end user [11]. This coordinated information sharing will give the end user a better experience: user profile– and session-related information creates the opportunity for blending together elementary services with proactive optimization and proximity-based pre-fetching of content that might be required by the user, on the basis of the user's location-related information.

Certain devices and optimization schemes adopted in current networking products and applications prove that efficient exploitation of proximity and knowledge of the user profile and network topology can improve performance, provide a tailored service, and make better use of available resources [7]. A Web site can deliver its content cached by the end user location,

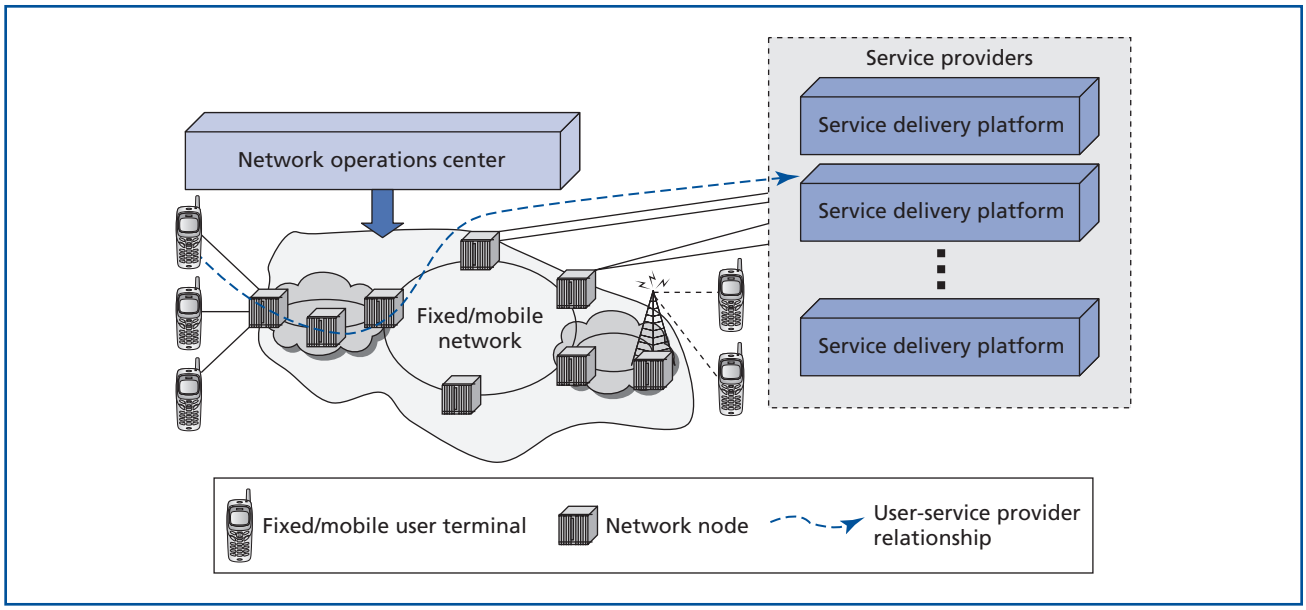


Figure 1.
Classical reference architecture for relationships between users and service providers.

and, when a limited-capacity terminal device such as a mobile phone or personal digital assistant (PDA) is identified, the content can be delivered in an adapted form [13]. Network management systems employ delegation schemes for efficient collection of performance measurement data or remote logs from network equipment [2]. Instead of contacting each information source directly from a central system, intermediate nodes that are topologically closer to the data source can be delegated to correlate different types of information. Local procedures may perform data fusion and then report the overall result to a client.

A comparison of terminal-oriented and network-oriented distribution of intelligence shows that placing computation and logic in the network [11], rather than merely considering the network as a set of fast and reliable bit pipes, can provide significant advantages for some types of services. From this perspective, our paper is structured as follows:

- We analyze the impact generated by a clear separation between the network provider and service provider when these distinct entities need to distribute service applications to the network. The roles of the network stakeholders may change, and a network provider can enhance its own business model by enabling service providers to run their applications directly on network nodes and by relying on local information provided to personalize content. Building a general business model based on these concepts requires an ecosystem spanning from the business to operational aspects, to the technological architecture, applications, and tools required to implement it. In particular, from the technological point of view, this leads to the vision of a network evolving from a set of bit pipes to a "network platform," behaving as an ecosystem for the deployment and execution of service applications.
- The key point of such an ecosystem is the introduction of an open, distributed processing platform in the network, supporting the need to share information between the network infrastructure (where most of the connection-, location-, and user-related information can be disseminated) and the application modules designed and deployed by the service provider. Clearly, all of this implies a paradigm shift, affecting, in particular, how and where service software is deployed and run.
- Focusing on the technological requirements of secure and orderly software execution in this context, we address the issues and risks of introducing such a platform and analyze possible solutions.
- Finally, we describe an experimental project implementing the managed computing grid (MCG), the proposed technical architecture that addresses these issues using existing technologies to support the required ecosystem.

A Network Platform for Computing as a New Arena for Service Applications

A total organizational decoupling of the responsibilities of the network infrastructure operator and the service provider is key to the definition of our operational business model. A network operator running an "open platform" for network-distributed computing (OPNDC) may offer several competing service providers an opportunity to deploy their software modules, loading the desired service logic directly onto network equipment or network management system nodes. Such a separation of roles allows both the network operator and the service provider to focus on its respective core mission and makes it faster and easier to deploy new network-intensive services packaged as plug-ins independent of network infrastructure upgrades. The abstraction provided by the OPNDC decouples the service application plug-ins from the underlying network node hardware, making their evolution cycles independent and reducing the total development investment for service providers because of the portability of the implementation.

In this alternative "teamwork" approach, the network infrastructure operator continues to own the bit pipes but also administers and sells access to its OPNDC, enriched with network-based information ready to be used by service applications. The resulting secondary market of value-added services, provided by a constellation of smaller and dynamic partner companies through the deployment of plug-in software for the standard platform of network machines, can expand the offer of new value-added services, relying on network-based information to create a common user context across the services.

It goes without saying that as in any network, for a "value network" scenario to work well, a clear definition and proper design of interfaces between the OPNDC and the service applications are of critical importance. From a developer's point of view, the OPNDC "is" precisely the interface exposed by the network infrastructure operator to the software modules deployed by the service providers.

Although in an industry characterized by a totally different business model, a similar, beneficial paradigm shift successfully occurred long ago in the personal computing arena, where originally the hardware makers also provided the operating system and almost all software applications for their proprietary-architecture machines. Over time, hardware architectures became open and standard, creating a market for competing operating systems. The operating systems in turn exposed well-documented, stable, and standard application programming interfaces, creating a flourishing market for software applications. In today's personal computer market, it seems natural and beneficial that hardware, operating systems, and software are provided by different actors, with innovations introduced at asynchronous times but generally in a backward-compatible way. The key role of specialization, layering, and open interfaces is evident in this process.

Replicating this evolution in network infrastructure technology may ultimately pave the road to a new market where service software providers compete to offer the best plug-in modules for deployment on a common platform, just as today a number of competing phone services are "deployed on" the traditional telephony network. This evolution would be beneficial for the end user, who would enjoy a broader choice of value-added and innovative services; for the network infrastructure operator, who would see increased business; and for the network infrastructure vendors best positioned to offer the best-performing and most stable computing platforms consistently.

In particular, the role of the network operator will evolve to become the provider of an OPNDC ecosystem which, from the software deployment and execution point of view, functions as a managed computing grid both to the OPNDC owner and to the software developers. The MCG is roughly equivalent to a "distributed server farm." However, unlike the outcome with conventional Web server farms, which evolved into a commodity, the distributed nature of this particular computing platform, with its inherent coupling with the network nodes, makes it inextricably bound to the transmission network, enforcing the competitive position of the infrastructure provider as the sole access gateway to deploy network-based software modules for providing services.

The Network Has the Answers

Blending and customizing pieces of services require a means for service plug-in software developers to obtain information about user location, terminal capabilities, user profiles, and user session data. Making this data readily available, through a standard, easy to access, and well-documented interface, is a common requirement for many different services and can become a network facility. Fast-reacting, context-driven activation is another area where network intelligence is beneficial. Service logic that selects customized content and initiates its delivery should be triggered when certain conditions are detected regarding "who, where, when, what" questions. And in a distributed architecture, the service logic should be activated on the node that is most convenient for topological or computational reasons. Topology considerations may typically suggest selection of the node closest to the information required, or to the user or terminal involved. Computational reasons may exclude nodes with insufficient resources to accomplish the task.

In the case of roaming users, a problem similar to the handoff issue in cellular networks arises. Mobile session handoff is required to ensure stable, uninterrupted connections at the physical or data link layer, whereas service-oriented network platforms would approach the handoff issue from the perspective of optimizing performance and resource usage on the basis of proximity opportunities and the requirements of the application layer. From this point of view, a service layer handoff mechanism could rely on distributed code and even on mobile agents to ensure uninterrupted service operation for mobile users—and

optimized resource usage for the network provider. Implementing these two processing paradigms requires an OPNDC that provides the necessary facilities—such as message passing, rendezvous, and code relocation—in a standard way that makes it simpler to implement node-independent service logic software.

Without an ecosystem such as the OPNDC that is neatly decoupled from the basic network equipment software, it is much harder to implement efficient high-bandwidth, profile- and location-based services on the network. And without an enabling technology such as the MCG which provides the opportunity to administer software on network equipment and to gain insight into the network's embedded software environment, both network operators and independent service providers would have no option but to view the network as a mere connectivity provider, and to design their service entirely on top of it, as seen in the Internet model.

Issues in an Open Platform for Network-Distributed Computing

With the evolution of network equipment architecture, and the increase in available computing power, memory, and storage resources, the software environment is changing from a severely limited, application-tailored, "closed" embedded system, to an open-standard system with well-known application programming interfaces and a range of built-in facilities that include modern and powerful scripting languages (such as those available on a Linux* system). These computing platforms are hosted in network nodes or integrated as pluggable extensions close to network nodes. Protocols to control the overall network behavior are running on top of the standard computing platform, but, of course, the access to these services is embedded in the network management architecture of the operator. Normally, other parties are not allowed to use them or are forbidden even to try to add/delete applications running on a network node.

The OPNDC mission is to create a generic open platform running software modules that implement service plug-ins developed by third parties, hosting them on the computing platform of network nodes. This new approach creates significant issues, mainly in three areas:

1. Security,
2. Resource control, and
3. Software management.

These issues must be addressed by the managed computing grid, the enabling technology on which the OPNDC vision is built.

MCG Security Aspects

Security is the most obvious risk in this architecture because "alien" software would be installed and executed directly on network equipment [5]. Without protection, third-party software may, as a result of either programming errors or intentionally malicious design, interfere with other processes, corrupt data, and alter configuration files. Moreover, if the platform provides facilities for replicating and migrating code to neighbor nodes to support the mobile agent paradigm, and there is no mechanism to limit this behavior, then there is the risk that the plug-ins may spawn out of control like a notorious "Internet worm," quickly saturating memory, as well as the central processing unit (CPU) and storage resources of the nodes.

Therefore, a clear separation between service-layer software plug-in modules and the standard management and control features must be ensured, also considering that their correct operation is the responsibility of two different categories of actors, although it is ultimately the common interest of both. A spectrum of solutions for achieving this functionality exists, ranging from simple permission-based protection, to a lightweight "sandbox approach" in interpreter-based execution, to total encapsulation based on virtualization technology [1].

Security considerations also require that any plug-ins to be installed were actually developed by trusted entities. The "publisher authentication" problem can be solved by using digital certificates to sign code before installing it on network nodes: without a valid signature applied with a certificate issued by a recognized authority, the platform will reject installation and execution. In the case under analysis, in order to maintain complete control of the situation, the certificate authority may be the network infrastructure operator itself.

If communication between plug-ins is required for their operation, and plug-ins from many competing service providers are operating simultaneously on the network, possibly even running on the same network nodes, then it is necessary to protect these communications from eavesdropping and other threats. The architecture supported by the MCG is focused on enabling the control of the communication channels that are set up between the planned entities (e.g., between the distributed plug-ins and the centralized servers of the same service provider), whereas the security of flow within the channel resides with the plug-in applications.

Control of Network Node Computing Resources

The issue of controlling the consumption of resources, for example, the computing load, is closely related to the security issue. Third-party software modules may be flawed or poorly optimized, entering infinite loops or wasting CPU cycles in poorly designed algorithms. System resources such as network connections, file handles, and memory areas may be consumed to a point where system stability is compromised, affecting the orderly operation of basic functions of network equipment or the timely response of critical algorithms.

The MCG approach is based on frameworks for mobile agents that rely on mechanisms such as sandboxing to allow control of resource usage by hosted applications [4], as represented in **Figure 2.**

In addition to this "hard," preemptive way of enforcing certain limitations on resource usage by the actor, a "softer" method may be used whereby each application declares the required resources prior to starting execution, and the MCG verifies its compatibility with the resources available in the candidate node for execution.

Third-Party Software Management

If third-party software is to be deployed on network equipment (possibly on hundreds or thousands of nodes), the availability of powerful facilities for software management is crucial for the network provider. Obtaining an inventory of installed and active plug-in modules is essential in order to monitor the situation properly, to estimate spare capacity that can be made available to service providers, and to plan upgrades.

Upgrades of third-party modules should not affect service operation and should be easy to manage, ideally accomplished automatically or delegated to the service provider. The new software releases can gradually spread over the network, reaching every node by "diffusion" using peer-to-peer file sharing mechanisms, even without a centralized system orchestrating the process.

Besides providing the surveillance and administration needs of the network infrastructure operator, who is concerned with ensuring maximum stability to the overall computing grid, the software management system can provide some degree of monitoring and control directly to the service provider. Prior to carrying out routine maintenance operations on a node which is executing service plug-ins, an administrator might choose to relocate actively running programs to a neighbor node from the remote management console. In situations of system instability or poor performance, administrators need facilities to monitor and control the execution state and resource consumption of software plug-ins and to adjust resource limitations applied by the system.

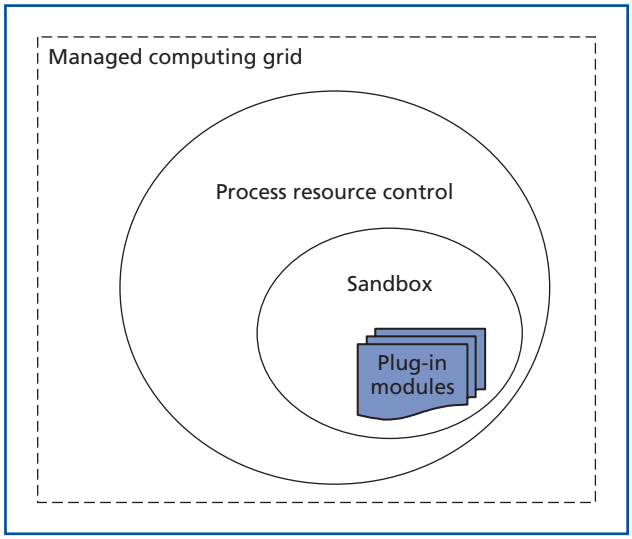


Figure 2.
Layering of process resource control and code sandboxing mechanisms.

Finally, also in the area of software management, the issue of connection control also has significant implications for the fields of security and resource control. Service agents deployed in the network cannot be granted total freedom to open unlimited connections to any address. Rather, for the sake of overall system stability and congestion prevention, a service agent can be characterized by the number of network connections allowed and by the identification of the authorized peer entities for communication.

Localization of these entities is required prior to creating a connection, and the open computing platform should support localization as a service to the application plug-ins. These functions help end-user terminals identify the nearest service module of a certain class. They can provide redirection mechanisms suitable for the particular protocol or technology, for example, Hypertext Transfer Protocol (HTTP) redirect primitives for Web-based traffic or Session Initiation Protocol (SIP) proxy redirect for Voice over Internet Protocol (VoIP). This feature provided by the infrastructure simplifies the work of the service provider, which no longer needs to handle proximity agent localization and redirection, and at the same time it enforces efficient use of network resources by optimizing the connection "distance" between users and service agents.

Figure 3 shows an overview of the advantages of the proposed approach. Service agents are deployed on the transport network nodes, an ideal position for their pivotal role among service users/terminals, the nearest service resource databases (e.g., video servers or advertisement servers), and the relatively few service-support systems that remain centralized, such as the central billing system and the central user profile database.

Leveraging their knowledge of information such as local network topology, bandwidth availability, and user location, the distributed service agents can effectively handle node handoff and connection redirection (possibly with bandwidth arbitration) at the local level. They can offload billing activity by retaining relevant information during the session and they can cache user profile information locally for faster access.

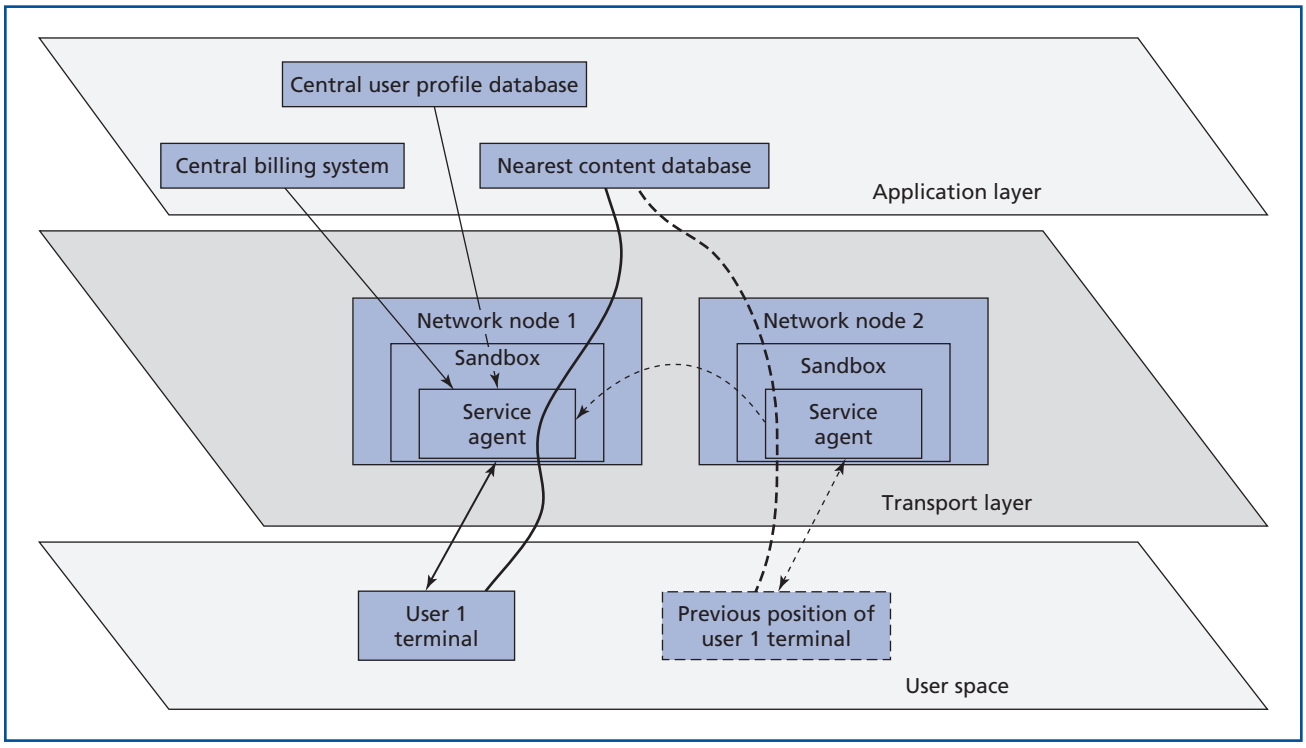


Figure 3.
The application architecture overview.

They can even behave as mobile agents to follow mobile users individually, ensuring continuity of their sessions and ultimately of their service experience. Service agents may also locally perform content profiling, format adaptation, service push, and other computation compatible with local processing power.

The BarleysNew Project

The MCG architecture was studied within an experimental project called BarleysNew, which was named after the Italian town of Orzinuovi, whose castle entrance provides a unique portal to access a number of "services": the church, the gallery, and the castle itself. The project was conducted for service-oriented applications running on next-generation network nodes such as the Alcatel-Lucent 1850 Transport Service Switch (TSS). These nodes host a Linux-based service processor and operate universal switching of both time division multiplexing (TDM)-based and packet-based traffic flows. The following sections describe the solution prototype that addressed the requirements and issues mentioned previously.

The MCG Actors and Playground

The architecture of the software running in each node is designed with three layers as shown in **Figure 4.**

1. The middleware implements an environment for controlled execution with behavioral rules structurally enforced (the playground) and hosts the service application plug-ins (actors).

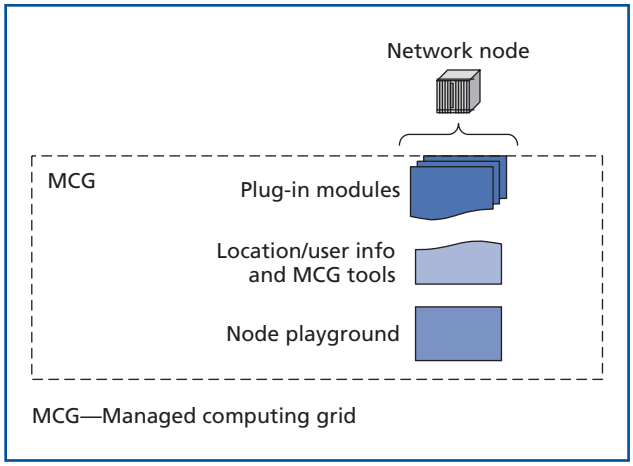


Figure 4.
Architecture of the managed computing grid software running in each node.

2. The extensions (i.e., the tools) refer to a set of software components that implement all the functionalities useful to help the actor, including communication, mobility, and access to local information as made available by the network provider.
3. Hosted application actors are the minimal computational units controlled by the playground. An actor is defined by the triplet: A := (C, B, I) where "C" is a code block that implements the intelligence of the actor; "B" is a dataset called "backpack" which is used by the actor to accomplish its task; and "I," the interface, is a set of additional information that contains data dedicated to the static and dynamic description of an actor (e.g., interface, intention, version, author). It is possible for the site to check this additional dataset to obtain information useful for playground customization. In the BarleysNew project, actors are based on scripting languages for portability, interoperability, easy management and execution control of code at run time, extensibility, and satisfactory performances.

The middleware in the nodes providing the playground is based on the mobile agent–based framework, taking into account the requirements of:

- *Modularity*. Modularity must enable simultaneous operation of numerous actors that, alone or cooperating, accomplish the application tasks. Modularity is also present inside the playground that is composed of several modules dedicated to elementary tasks such as communication services or persistent storage management. The playground must dynamically support the following operations:
  – Adding a new scripting language to the framework,
  – Adding a new system service to help the actors, and
  – Changing security policies for an actor or for a group of actors.
- *Decoupling*. Decoupling exists between the MCG playground and the languages used to implement the actors [12]. This allows the MCG playground to execute and administer every kind of actor. It is only necessary to install the correct interpreter.

- *Encapsulation and layering*. Encapsulation and layering are guaranteed by an architecture composed of several layers communicating through well-defined interfaces and formatted data flow. These policies allow for extensibility in the future.

Tools for Actors

The playground offers the actors a number of services (i.e., the extensions) called tools. These tools are additional functionalities that actors can exploit to accomplish their tasks and to take advantage of the local environment of the network node. Furthermore, the tool set represents a set of actor capabilities that are managed and controlled by the playground to guarantee the respect of security constraints. In the following sections we provide a description of the MCG basic tool set.

Check tool. The check tool allows online substitution of an actor on a playground with a newer version of the same actor. This tool, once called, verifies that a newer version of the calling actor does not exist on the playground. If a newer version is found, the check tool kills the calling actor and loads the newer one. The check tool invocation should be done at a preemptive point of execution. The check tool, if necessary, dynamically substitutes the process image containing the calling actor with a new one, maintaining the queue of messages that will be forwarded to the new actor.

It is important to point out that every actor is digitally signed and versioned in order to prevent malicious code injection in a playground. Digital signatures of the service agents are also the basis of service provider authentication. This is necessary to identify service subscription relationships that exist between the user and the service provider, in order to grant or reject permission to perform a given action.

Rendezvous. The "rendezvous" tool implements a simple publish-subscribe service. Basically, it is possible for every actor to subscribe itself to a defined event. The event could be determined by the unique identifier or by the entity publishing the event. The MCG playground provides a set of pre-defined events, which allow the actors to synchronize with data relevant to location/user information and session status changes.

Closet. Every site has stable storage locally available and every actor has access to this storage through the closet tool. Every actor can specify, for every data item stored, whether its value is public or private (i.e., accessible to every other entity present in that node). To prevent malicious operations, the closet has a fixed maximum size that is dependent on the security profile of every actor. The MCG playground provides a closet of public quasi-static data related to the local environment.

Diary. Every actor can also perform logging into persistent stable storage. The diary tool is dedicated to this task. Unlike for the closet tool, the values logged using the diary are public and the actor is not allowed to delete or modify them. Invocation of selected tools automatically triggers logging activity by the diary tool.

Mobility. The MCG is designed around a weak mobility agent model supported for the actors. When an actor relocates, its destination is declared by the calling actor. In order to declare a relocation, the actor should relay its destination site to the mobility tool. This is accomplished through its identifier, which is retrieved via information provided by the playground using one of the tools described. The actor then creates a backpack containing the data needed at the destination. In the specific case of a service agent "following" the end user, the backpack might contain user profile information or other information required while providing the service. In the BarleysNew project, the relocation tool composes a well-formatted Extensible Markup Language (XML) string representing the actor and sends it as a message to the destination site.

When the message is received, the playground installed in the receiving site will extract the data, save it in stable storage, and start a new process to run the actor inside a sandbox. This process uses the backpack as a parameter.

The MCG Actor's Lifecycle

Figure 5 provides a view of the ecosystem supporting the actor plug-in's lifecycle. This includes

- Design and development, carried out by the service provider,
- Installation management, execution, administration, and termination, carried out by the network provider.

Design and development. Methodologies and patterns to create the application design are available in research literature, but a complete review is beyond the scope of this paper. The key point is that the network provider delivers a mobile actor development kit (MADK) to the application developer, including:

- *The definition of language constraints*. This makes it possible to run the application in the environment of the target network node. In the BarleysNew project, the language is a re-definition of the Practical Extraction and Report Language (PERL) [9], named "ActorPERL." The PERL scripting language has been enriched with MCG extensions, adding methods to interoperate with the playground tools. On the other side, ActorPERL restricts the usable PERL operation codes (opcodes) to allow the generated actor to operate in the MCG playground sandbox. The ActorPERL source code is translated into an intermediate language bytecode. This intermediate language bytecode is in the package delivered to the network provider, which can check the bytecode to verify that the sandbox rules are not violated. ActorPERL takes into account several trade-offs among performance, expressiveness, and security. The same schema can be applied to other languages, extending the MADK without changing the overall process and/or trade-offs.
- *The specification of the dataset used by the MCG check tool*. The dataset identifies the application service, the provider signature, and the reference probe source identifier. The application service identifier is used by the network to resolve the naming between the generic service name invoked by the end user, into the actual nearest actor address offering the requested service. Furthermore, if required, this dataset will allow the network provider to plan the networking resources correctly to build the pipes between the "to-be-distributed" actor instances and the part of the service application, which will still run on the centralized service delivery platform.

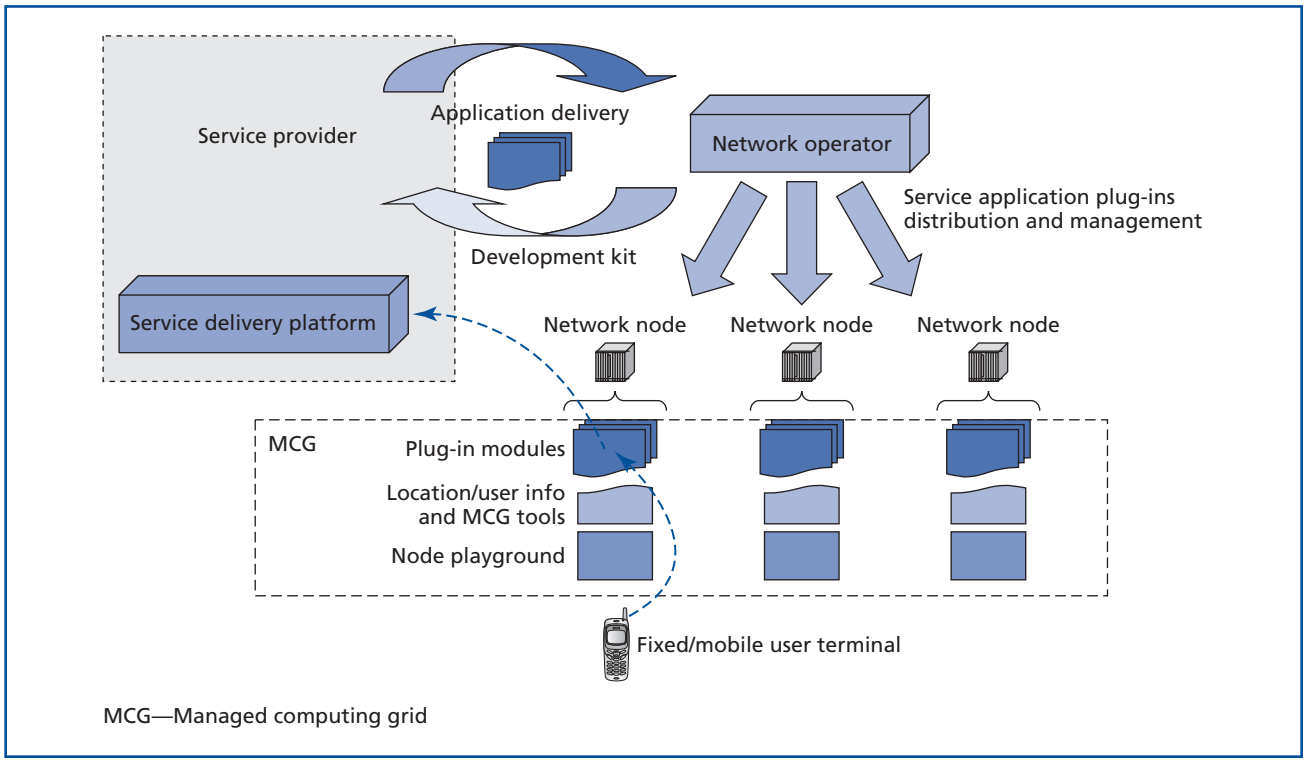


Figure 5.
The ecosystem supporting the actor plug-in's lifecycle.

- *The specification of data and methods to access the MCG playground tools*. This includes information available from the local node and the data required for relocation when following a nomadic user.

The service application, packaged according to the MADK rules, is delivered to the network provider for installation.

Managing the actors. After screening for compliance with the MADK rules, the network provider is responsible for distributing the delivered application into the appropriate nodes. The MCG environment provides the operational management workflow:

- To manage the repository of actor packages and related information about the source provider.
- To handle the actor distribution over the hosting network nodes and control theinstance activation. This process requires binding the actor instance with the actual address to reach it, and entry into the service naming resolution handler owned by the network provider.
- To control an actor backpack relocation when invoked through the MCG mobility tool.
- To control and monitor the MCG diary data.
- To clean up presence data from any actor when its lifetime is over.

Operations management includes the update and control of the playground middleware itself, which is managed as part of the network node software.

The BarleysNew Prototype

It is worth noting that the specific set of plug-in services designed in the experimental phase was related to network management tasks with the assumption that the roaming users actually belonged to the network maintenance team.

The experimental system demonstrated the concept and provided a methodology to distribute, install, activate, monitor, and stop samples of ActorPERL applications running and migrating in their distributed playground on network nodes.

Further work will deal with challenges such as introducing support for other programming languages as already defined in the MCG architecture, investigating performance and load balancing issues, testing complex distributed algorithms on the platform, and interacting with the management information base of network nodes to accomplish network control operations.

Conclusions

This paper described a new model for service development. It provided an analysis of impacts seen when there is a clear separation between service and network provider organizations and presented opportunities for the network providers to expand their role of "dumb bit pipe" provider by enhancing service applications with user location and network-aware information. This set of information could be the glue that blends a new generation of services offered to the end user. This paradigm needs a complex ecosystem (the OPNDC), supported by an architecture (the MCG) which addresses technical issues such as distributed application software generation, certification, and distribution. As proof of concept, we presented the basic choices adopted in an experimental project designed according to the proposed architecture.

Acknowledgements

The authors would like to acknowledge the contributions of Michele Panzeri and Giordano Tamburrelli, who contributed to the BarleysNew project with master papers submitted for their master of science degrees in computer science at the University of Illinois, Chicago, United States, and at Politecnico of Milano, Italy. Special thanks also go to Gus Zimmerman, Janet Pflederer, and Katherine Guo for their help and comments on the manuscript.

*Trademarks

Linux is a trademark of Linus Torvalds.
PERL is a trademark of the Perl Foundation.

References

[1] M. Alfalayleh and L. Brankovic, "An Overview of Security Issues and Techniques in Mobile Agents," Proc. 8th Internat. Federation for Inform. Processing (IFIP) TC-6 TC-11 Conf. on Commun. and Multimedia Security (CMS '04) (Windermere, Eng., 2004), published in IFIP Series Vol. 175 (D. Chadwick and B. Preneel, eds.), Springer, New York, 2005, pp. 59–78.

[2] M. Baldi and G. P. Picco, "Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications," Proc. 20th

Internat. Conf. on Software Engineering (ICSE '98) (Kyoto, Japan, 1998), pp. 146–155.
[3] P. Bellavista, A. Corradi, R. Montanari, and C. Stefanelli, "Context-Aware Middleware for Resource Management in the Wireless Internet," IEEE Trans. Software Engineering, 29:12 (2003), 1086–1099.
[4] W. Binder, J. G. Hulaas, and A. Villazon, "Portable Resource Control in Java: The J-SEAL2 Approach," Proc. 16th ACM SIGPLAN Conf. on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '01) (Tampa Bay, FL, 2001), pp. 139–155.
[5] W. M. Farmer, J. D. Guttman, and V. Swarup, "Security for Mobile Agents: Issues and Requirements," Proc. 19th NIST-NCSC National Inform. Syst. Security Conf. (NISSC '96) (Baltimore, MD, 1996), pp. 591–597.
[6] A. K. Jain, "Intelligent Multiservice Networks," Bell Labs Tech. J., 7:1 (2002), 81–97.
[7] T. Lehman, J. Sobieski, and B. Jabbari, "DRAGON: A Framework for Service Provisioning in Heterogeneous Grid Networks," IEEE Commun. Mag., 44:3 (2006), 84–90.
[8] J. Liu, J. Xu, and X. Chu, "Fine-Grained Scalable Video Caching for Heterogeneous Clients," IEEE Trans. Multimedia, 8:5 (2006), 1011–1020.
[9] R. L. Schwartz, T. Phoenix, and B. D. Foy, Learning Perl, 4th ed., O'Reilly, Sebastopol, CA, 2005.
[10] J. Steinberg and J. Pasquale, "A Web Middleware Architecture for Dynamic Customization of Content for Wireless Clients," Proc. 11th Internat. Conf. on World Wide Web (WWW '02) (Honolulu, HI, 2002), pp. 639–650.
[11] M. Wegdam, J. van Bemmel, K. Lagerberg, and P. Leijdekkers, "An Architecture for User Location in Heterogeneous Mobile Networks," Proc. 7th IEEE Internat. Conf. on High Speed Networks and Multimedia Commun. (HSNMC '04) (Toulouse, Fr., 2004), published in Lecture Notes in Comput. Sci. (LNCS 3079) (Z. Mammeri and P. Lorenz, eds.), Springer, Berlin, Heidelberg, New York, 2004, pp. 479–491.
[12] H. Xu and S. M. Shatz, "ADK: An Agent Development Kit Based on a Formal Design Model for Multi-Agent Systems," J. Automat. Software Engineering, 10:4 (2003), 337–365.
[13] J. Xu, X. Tang, and D. L. Lee, "Performance Analysis of Location-Dependent Cache Invalidation Schemes for Mobile Environments," IEEE Trans. Knowledge and Data Engineering, 15:2 (2003), 474–488.
[14] S. Yadav, S. Bakshi, D. Putzolu, and R. Yavatkar, "The Phoenix Framework: A Practical Architecture for Programmable Networks," Intel Technol. J., 3Q (1999), <http://www.intel.com/technology/itj/archive/1999.htm>.

(Manuscript approved March 2008)

PAOLO FOGLIATA is director of the multi-service



network solutions team at Alcatel-Lucent's Optics Division Lab in Vimercate, Italy. His 20 years of work experience include developing equipment for packet data networks and leading the optical network management applications labs. He holds a degree in electronic engineering from Politecnico of Milan, Italy. His interests are mostly focused on multimedia multi-technology network solutions that rely on high-bandwidth multi-service optical infrastructure.

MARCO TORQUATO MUSSINI is a deployed software



officer in Alcatel-Lucent's Optics Network Management Research and Development in Vimercate, Italy, where he also directed the Element Manager Development team and supervised a number of graduate students working on various software projects. His professional interest mainly focuses on innovation in the software domain: technological prospection, graphical interfaces, architecture and engineering, and development processes and tools. He has been a distinguished member of the Alcatel-Lucent Technical Academy since its foundation in 2001. He holds a degree in electronic engineering from Politecnico of Milan, Italy, and a postgraduate master in information technology from CEFRIEL, Milan, Italy. ♦