# HiMang: Highly Manageable Network and Service Architecture for New Generation

Taesang Choi, Tae-Ho Lee, Nodir Kodirov, Jaegi Lee, Doyeon Kim, Joon-Myung Kang, Sungsu Kim, John Strassner, and James Won-Ki Hong

*(Invited Paper)*

*Abstract:* **The Internet is a very successful modern technology and is considered to be one of the most important means of communication. Despite that success, fundamental architectural and business limitations exist in the Internet's design. Among these limitations, we focus on a specific issue, the lack of manageability, in this paper. Although it is generally understood that management is a significant and important part of network and service design, it has not been considered as an integral part in their design phase. We address this problem with our future Internet management architecture called highly manageable network and service architecture for new generation (HiMang), which is a novel architecture that aims at integrating management capabilities into network and service design. HiMang is highly manageable in the sense that it is autonomous, scalable, robust, and evolutionary while reducing the complexity of network management. Unlike any other management framework, HiMang provides management support for the revolutionary networks of the future while maintaining backward compatibility for existing networks.**

*Index Terms:* **Autonomic management, cloud-based virtual networks, cognitive mamanagement, network management architecture, policy continuum, smart networks.**

## I. INTRODUCTION

The Internet is one of the most successful human inventions of the 20th century. However, since it was not originally designed to accommodate a massive number of users or for commercial use, it has fundamental architectural and business limitations. To name a few, processing, handling, storage, transmission, and operation limitations [1] and the inability to match business needs to network services and resource offerings are some such limitations. Scientists and researchers from various companies and research institutes worldwide have been working toward understanding these architectural and business limits, so as to progressively determine principles that will drive the future Internet architecture in order to adequately meet the above-mentioned challenges.

In this paper, among these many limitations, we focus on one particular issue: The lack of manageability. Ironically, even though people acknowledge that manageability is a critical aspect of network and service design, it has not been considered as an integral part when actually designing new networks and services.

In the current Internet, network infrastructure is instead developed and deployed first and management and security are separately added later. Thus, functions that need management cannot be designed or tested until management is added. Our proposed architecture, highly manageable network and service architecture for new generation (HiMang), addresses this issue. We argue that our design reduces the complexity of network management architecture and provides autonomy, scalability, robustness, heterogeneity, and backward compatibility.

The future networking environment is expected to be composed of a much increased number of managed entities, including services and devices, and management tasks due to newly introduced services (e.g., cloud computing, green networking, and Internet of things (IoTs)), new types of managed resources (e.g., virtual machines, virtual networks, and virtual elements), dynamic process management, support of heterogeneity, and stringent but optimal resource utilization. These requirements will introduce unforeseen complexity in the management of this environment. To make the matters worse, consumers always want more for less. Network equipment vendors, service providers, and network operators have responded to this challenge by building smarter, feature-rich devices; at the same time, network operators have been offering new types of access methods based on technological advances. Such changes, coupled with the emergence of mobile Internet, have shifted the traditional operator-centric focus on reliability and quality of service (QoS) to user-centric operations that demand rapid innovation in applications and the integration of pervasive communication into the fabric of everyday life.

This inevitably leads to demands for more local autonomy over the management of resources. Five main sources of complexity are listed in [2]; they are as follows:

- Separation of business, technology-neutral, and technology-specific information that relates to different perspectives of the same entity. This is necessary in order to ensure that the needs of different stakeholders are represented.
- Difficulty in harmonizing network management data that is

Manuscript received September 28, 2011.

This research was supported in part by Korea Communications Commission (KCC), Korea, under the "Novel Study on Highly Manageable Network and Service Architecture for New Generation" support program supervised by the Korea Communications Agency (KCA) (KCA-2011-10921-05003) and the World Class University (WCU) program through the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (project no. R31-2008-000-10100-0).

T. Choi, T.-H. Lee, N. Kodirov, J. Lee, and D. Kim are with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea, email:{choits, kthlee, nodir, jkilee, kdy}@etri.re.kr.

J.-M. Kang, S. Kim, and J. W.-K. Hong are with the Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Republic of Korea, email:{eliot, kiss, jwkhong}@postech.ac.kr.

J. Strassner is with the Division of IT Convergence Engineering, Pohang University of Science and Technology (POSTECH), Republic of Korea, email: johns@postech.ac.kr.

1229-2370/11/$10.00 

inherently different.

- Problems in integrating new functionality and new technologies due to the lack of a common management data and programming model used by all components.
- Isolation of management and operational data into separate repositories that are accessed by different protocols and languages, thereby preventing their exchange and reuse.
- Problems in being able to detect and respond to user and environmental changes.

Autonomics is, first and foremost, a way to manage complexity [2]. If an autonomic system can perform manual and time-consuming tasks, operators and administrators can save valuable time and operational costs, enabling them to focus more on higher level cognitive network functions.

Another important capability for managing complexity is scalable management application placement. To manage huge heterogeneous communication networks environment, neither centralized nor fully distributed management solutions may be suitable. We argue that by grouping autonomic components into a hierarchy, the network overhead associated with managing network devices and other resources can be greatly reduced. Further, dissemination of context, propagation of policies, and collaboration between autonomic components can be more efficiently orchestrated, resulting in a more scalable architecture. In addition, the hierarchical structure can be logically mapped to the structure of the organization and its infrastructure, simplifying configuration and management at all layers of the organization.

Robustness or availability of the management system, in addition to the support of scalability, is equally important for dealing with management complexity. The traditional way of providing fault tolerance to the management system may not be sufficient for the above-mentioned future network environment, since it is usually achieved at the cost of scalability. Management complexity alleviation can be best served when robustness can be guaranteed without sacrificing scalability.

Other important factors that increase management complexity are heterogeneity and backward compatibility. A compromise design approach, in comparison to either the incremental or revolutionary approaches, enables new ideas to evolve while simultaneously emphasizing backward compatibility with existing networks. This is very important to certain stakeholders, such as Internet service providers (ISPs), who have invested billions in their equipment and want to leverage those investments. This approach also recognizes that current and future networks and networked applications have vastly different requirements; this implies that a single architecture cannot simultaneously meet these different needs.

Above all, all these management capabilities have to be considered intrinsically during the network and service design phases. HiMang aims at overcoming the legacy management limitations and the growing management complexity of the future networking environment by addressing the above-mentioned design principles. In this paper, we first examine related work in Section II. Given the gap analysis results, we identify core design principles and key features for managing complex future networks in Section III. Then, we present the HiMang architecture, which was designed based on these principles, in Section IV. In Section V, we propose a proof-of-concept system solution and some use cases. Finally, we conclude our effort with potential future work in Section VI.

## II. RELATED WORK

There are many programs and projects underway related to the future Internet. However, the vast majority do not address the manageability aspects of the future Internet. This section provides a small but meaningful sampling of the most important prior art relevant to this research. We attempt to analyze the pros and cons of the projects whose results have led to our design principles and the key features of the HiMang architecture.

### A. Autonomic Internet (AutoI)

The goal of the autonomic Internet (AutoI) project [3] is to develop a self-managing virtual resource overlay that can span across heterogeneous networks and that supports service mobility, security, QoS, and reliability through the concept of five planes: The orchestration plane, service enabler plane, knowledge plane, management plane, and virtualization plane. The knowledge plane offers a subset of the functionality provided by foundation, observation, comparision, action, and learning environment (FOCALE) [4]. The management plane consists of a set of autonomic management systems (AMSs). Each AMS consists of parts operating on the management plane and the knowledge plane, as well as interfaces to a dedicated set of models and ontologies and interfaces to one or more distributed orchestration components (DOCs).

The virtualization approach is novel, and they structured their information model and ontology very well. However, it is doubtful that the orchestration plane can accomplish its goal as they suppose. The AutoI project described interfaces between DOCs and AMSs, but they do not specify how they control and manage network resources to meet the overall goals and policies of heterogeneous networks. In our HiMang architecture, we use the policy continuum [5] to develop mappings between policy rules that correspond to each constituency. This enables context-aware policies to be used to orchestrate behavior for business goals, system-level interactions, and other forms of interactions.

### B. 4WARD

The 4WARD project [6] has developed a clean-slate network management approach called in-network management (INM) that is aimed at the management of large, dynamic networks where a low rate of interaction between an outside management entity and the network is required. 4WARD has developed two related approaches: (1) A new architecture, called networking of information (NetInf), where information retrieval and storage act on the objects themselves rather than on the nodes that contain the objects, and (2) architectural concepts to support INM. The idea behind NetInf is to develop an information-centric architecture in which hosts take a secondary role and information itself is the primary focus. The idea behind INM is that management tasks are delegated from management stations outside the network to a self-organizing management plane inside the managed system. This is facilitated through decentralization, self-organization, embedding of functionality, and autonomy.

4WARD has some interesting ideas. However, its information model is not a true information model, but is rather a simple data-oriented mapping of objects. Hence, it does not solve the interoperability problem. We do not believe that devices have the ability to manage themselves; while a router or switch has the physical space to contain such functionality and the logical computing means to support it, this approach is in general not applicable to the vast majority of end nodes that exist, especially those that are mobile-centric and have limited resources.

### C. FAME

The federated, autonomic management of end-to-end communication services (FAME) project [7] addresses the federation of communication networks, as well as autonomic network management. Current research into autonomic communications and management emphasizes the automation of decision making to reduce service providers' operational costs. However, by failing to address the highly dynamic and federated nature of modern service provisioning, this research runs the risk of simply replacing today's network management silos with autonomic network silos. The FAME project structured its research into three strands: (1) Federated communications service management, (2) service monitoring and configuration, and (3) network infrastructure coordinated self-management. The federation of heterogeneous networks requires harmonizing different management data from different networks, such as the resources being managed, the context used in communication services, and the policies used to express governance. The approach and methodology used by FAME are similar to ours. FAME uses information models and ontologies extracted from the directory enabled networks-next generation (DEN-ng) model [8] for the semantic mapping of management data, and the FOCALE model [4] is used for their architecture. FAME concentrates on federation, but we focus more on cognitive control and management of virtualized resources and cloud environments. We use a new FOCALE cognition model for data collection, management, and decision making that enables our system to learn about its own behavior and effectively respond to context changes.

### D. 4D

The data, discovery, dissemination, and decision (4D) architecture [9] is based on four planes: A data plane, a decision plane that controls and manages the network, a discovery plane that discovers the network statuses of the data plane, and a dissemination plane that exchanges discovered results with their managers. These four planes cooperatively define a centralized control architecture based on a view of the whole network; thus, the architecture is able to dictate direct control over the various distributed entities to meet the autonomous network-level objectives of mostly routing-related policy enforcement. It sets up a separate dissemination channel for control and management activities through link-layer self-discovery mechanisms. This removes the management and control plane bootstrapping problem and provides a basis for self-configurable networks. Although this centralized solution provides various improvements in routing control, it may also have some drawbacks in terms of scalability. Since the discovery and dissemination planes depend on network-wide broadcasts, the huge broadcast domain in large-scale networks may create a performance bottleneck.

### E. CONMan

The complexity oblivious network management (CONMan) architecture [10] is an extension of the 4D architecture. It reuses the discovery and dissemination mechanisms of 4D and extends the 4D management channel to accommodate multiple decision elements or network managers. Each network manager in CONMan performs different network management tasks in addition to routing management. Thus, CONMan can be considered to be a more general-purpose management solution. The objectives of CONMan are self-configuration, continual validation, regular abstraction, and declarative specification to avoid mainly two sources of complexity: The dependence of management capability on data plane reliability and the ever growing complexity of target networks. Besides providing discovery and dissemination mechanisms, it also implements module abstractions and pipes, which are primary building blocks for realizing network-wide management objectives. Each module exposes a generic complexity oblivious management interface. These extensions hide the complex details of the target managed networks. Thus, CONMan takes a less extreme measure than 4D by centralizing the configurability of the network at the granularity of module interactions rather than centralizing the whole control and management plane. CONMan, however, still has a similar scalability problem to that of the 4D architecture.

## III. DESIGN PRINCIPLES AND KEY CHARACTERISTICS

In this section, we describe the major design principles and key characteristics based on the gap analysis of related research, and limitations of the current communications network and service design described in the above sections.

### A. Autonomic Management Principle

- Management complexity alleviation through autonomicity: The HiMang architecture supports autonomic management, which is a way to manage complexity when managing heterogeneous network devices [2]. Autonomic management is often incorrectly characterized as one or more of the following self-functions: Self-configuration, self-healing, self-optimization, and self-protection. These are benefits of autonomic management, but they are not the defining features [2]. In order to implement any of these four functions, the system must first be able to learn about itself and its environment: What are its capabilities, and what is demanded of it? This is called self-knowledge, and this is the basis for situated autonomic communications [11]. Specifically, the ability to learn and incorporate knowledge at runtime is crucial for harnessing the power of emergent behavior to enable components and systems to dynamically self-organize [4].

  Since we must address both the business as well as the technical aspects of manageability, our approach introduces two important advances in the state of the art: (1) The ability to dynamically adjust the behavior of one or more resources or services in response to changing user needs, business objectives, and/or environmental conditions and (2) the ability

to proactively assist the user in his or her task. Since we still do not have a single programming language, and likely will never have one, we conclude that there will never be a single language for network devices as well. Therefore, our HiMang architecture takes the approach of developing a common internal form for a set of mappings from vendor-specific devices; in essence, this results in a network lingua franca that enables the capabilities of heterogeneous devices to be normalized. We have developed a novel knowledge representation that enables facts as well as consensual meanings and semantic relationships to be defined, which facilitates machine-based learning and reasoning.

- Cognitive management: The HiMang architecture supports cognitive management based on human intelligence models. A cognitive system is a system that can reason about what actions to take, even if it has not anticipated a situation. It can learn from its experience to improve its performance. It can also examine its own capabilities and prioritize the use of its services and resources, and, if necessary, it can explain what it did and accept external commands to perform necessary actions [12]. Minsky developed a model of human intelligence based on simple processes called agents. They interact according to three layers: The reactive (or subconscious), deliberative, and reflective layers [13]. Reactive processes make immediate responses upon the reception of an appropriate external stimulus. They have no sense for what external events mean; rather, they simply respond with some combination of instinctual and learned reactions.

  Deliberative processes receive data and can send commands to the reactive processes; however, they do not interact directly with the external world. They achieve complex goals by applying short- and long-term memory in order to create and carry out more elaborate plans. Reflective processes supervise the interaction between the deliberative and reactive processes. They reformulate and reframe the interpretation of the situation in a way that may lead to more creative and effective strategies. They consider what predictions turned out wrong, along with what obstacles and constraints were encountered. They also include self-reflection.
- Policy continuum: The HiMang architecture supports policy-continuum-based management, which addresses policies and their interaction with multiple constituencies. People play different roles in an organization, but work together on products or solutions. The goals that these people aim to achieve via policies appear to be as diverse as their roles, but these goals and hence their policies are intrinsically linked. The policy continuum provides this intrinsic link.

  The policy continuum offers five different abstractions (business, system, administrator, device, and instance) to address this concern [5]. The business view is employed by business users; therefore, in this abstraction, policies are specified using vocabulary native to business professionals. The business view provides both a business vocabulary, which is composed of both objects and facts that relate to the object, in the language of business professionals and a set of business rules that govern how the objects and facts are used. As a simple example, consider the following business definition for QoS in a communications network: "A customer who purchases an Internet service package with gold-level QoS gets preferred access to and usage of network resources." There is no mention of any technology or any particular device. The system view uses the same objects as defined in the business view, but adds further objects and details that correspond to concepts and terminology that are not appropriate in the business view. For example, QoS as defined at the system view includes detail pertaining to bandwidth, latency variations, and delay information. However, the system view is still independent of the device and technology; thus, a user who defines policy at the system view, such as a systems engineer, does not refer to any particular technology or vendor device. Reference to specific technology is introduced in the administrator view. This abstraction is device independent but technology specific. QoS policies defined by a network architect will include reference to specific QoS technologies such as differentiated service (DiffServ), multi protocol label switching-traffic engineering (MPLS-TE), and other traffic management capabilities. Reference to specific device features is introduced in the device view. Thus, the device view is both device and technology specific. For instance, policy for QoS defined by a network administrator will include references to queuing, scheduling, and dropping algorithms for a particular type of router using a particular version of a vendor's operating system. The instance view includes device- and technology-specific commands to implement device-level policies for a specific vendor device. For example, in Cisco IOS 12.x, one would specify access control lists (ACLs), policy maps, and service policies. The policy continuum represents an important milestone in enabling business goals to drive the services and resources provided by a network.
- Scalability and high availability: HiMang uses hierarchical autonomic network management architecture. Because of the complex requirements for network management and the increasing number of devices, network management has become more complex. Therefore, it is difficult to manage a network with a single network management server. Multiple network management nodes are distributed in the network for both current networks and future networks, and more management nodes will be needed as the size of the network domain gets larger. Each management node exchanges management information with the nodes it is cooperating with, and this could create an overhead in terms of network bandwidth. Introducing a hierarchy to management nodes reduces this overhead, because the managing node only sends information to the nodes it manages [14].

  Besides scalability, high availability of management systems should be another critical design consideration for providing nonstop operability for sustainable, high-quality future network services. For this purpose, HiMang provides a high-availability management mechanism called core library for rapid development of network management systems (NMSs (CORD)). It utilizes a consistent hash function and extensible markup language (XML) to guarantee high availability. The details and its performance evaluation results are described in [15] and [16]. Depending on the complexity of the management domain, an appropriate combi-

nation of hierarchical management and CORD mechanisms can be applied.

### B. Software-Defined Control and Management Principle

The data plane implements per-packet processing as part of the forwarding process. The control plane implements decisions that affect packets, such as those required by routing and signaling protocols. The management plane (which is really not a layer or plane in the true sense of the word, but rather a concept or a set of principles) monitors the network and coordinates the functionality of the control and data planes. Since the current network does not have independent data, control, and management planes, the data plane is used not just to transmit end-user data, but also to transmit control and management data. There are a number of important reasons to separate these three planes. Because of space limitations, we list three:

- Separate control and management planes prevent malicious users from changing the configuration of network devices through the data plane, and hence provide better inherent security.
- In current networks, if there is a problem with the data plane, that problem also may affect the control and management planes. Separation makes it possible to manage the network even when normal data traffic is adversely impacted.
- By separating the planes, computing power can be separated and accounting and auditing functions can also be simplified. This leads to more robust implementations.

The HiMang architecture, therefore, adopts the design principle of separation of the data, control, and management planes. It supports OpenFlow-like, software-defined networking for flow-level, fine-grained control, and global network-wide visibility and also provides the advantages mentioned above.

### C. Heterogeneity and Backward Compatibility Principle

There are three fundamentally different approaches for the design of the future Internet. The first is an incremental approach, which is evidenced by the plethora of solutions being applied to the current Internet that violate its architectural principles, such as network address translators, firewalls, and virtual private networks. The fundamental directive of this approach is to maintain compatibility with the existing Internet. However, the vast majority of these approaches are in fact stovepipe solutions meant to solve one particular problem without regard to other problems, and hence, this approach is no longer sustainable [17]. The second is what is called a clean-slate approach, which eliminates existing commitments, restraints, and assumptions and starts with a new set of ideas [17], [18]. While the first approach evolves a system from one state to another incrementally, the latter enables a radical redesign of the current Internet architecture.

However, a third alternative exists, which is a compromise between the above two approaches [19], [20]. This approach enables new ideas to evolve while simultaneously emphasizing backward compatibility with the existing Internet. This is very important to certain stakeholders, such as ISPs, who have invested billions into their equipment and want to leverage those investments. This approach also recognizes that current and future networks and networked applications have vastly different requirements; this implies that a single architecture cannot simultaneously meet these different needs.

## IV. HIMANG ARCHITECTURE

Fig. 1 shows a conceptual architecture of HiMang. The concepts of a control plane and management plane have been fundamental to the telecommunication model. The control plane focuses on real-time signaling among network elements for service provisioning and traffic engineering, and it implements distributed algorithms to enable data plane functionality. The management plane is a near-real-time approach for managing the network through appropriate external interfaces provided by network devices. However, the management plane itself cannot control the data and control planes, because control mechanisms can be very different in nature for different types of networks, such as those for wired and wireless networks. This means that the management plane must understand the differences in command syntax as well as the underlying semantics of these networks. To make matters worse, neither wired nor wireless networks are able to explicitly convey their semantics, due to limitations in the way that they represent their management data. This lack of self-awareness leads to a lack of self-awareness of the network itself. Self-awareness is the ability of an entity to understand its own behavior as well as that desired of it.

The inference plane is a coordinated set of intelligent decision-making components that represent the capabilities of the computing elements being controlled, the constraints placed upon them using different functions, and the context in which they are being used. The most important difference between the management plane and the inference plane is that the inference plane uses semantic knowledge to orchestrate the behavior of the system using the management plane. Another distinctive feature of the inference plane is that it supervises the optimization and distribution of knowledge, both in itself as well as for use by other planes. Hence, the inference plane uses a combination of local and global knowledge to accomplish its goals. For example, the use of models and ontologies enables the solution of interoperability problems using inference techniques. Enablement, distribution, mapping, and relationship management of autonomic elements (AEs) throughout the planes are novel features in our architecture.

There are three important aspects of the implementation of the architecture: (1) Knowledge representation and organization, (2) FOCALE control loops for monitoring and reconfiguration of the networks, and (3) distribution and interaction of AEs, which collect and analyze data and provide commands to network devices. Detailed explanations are given in the following subsections.

### A. Knowledge Representation and Organization

Knowledge is critical to our approach for two different reasons. First, without a single definition of knowledge, it is impossible to solve the interoperability problem that prevents disparate data produced by different vendors to be shared and reused. This problem is exacerbated by the multiple conflicting standards that are present in the telecommunications world, as well as by artifacts such as private management information

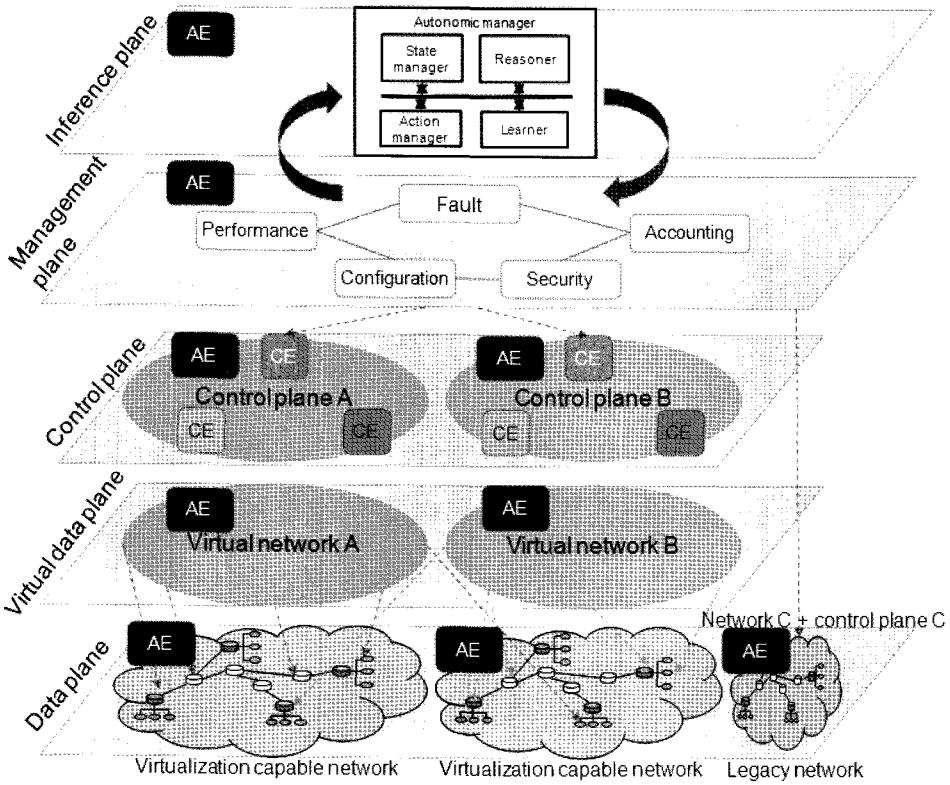



Fig. 1. HiMang conceptual architecture.

bases (MIBs). Second, network management has been limited in its power by having to deal with data, instead of having access to machine-readable semantics. Hence, our approach relies on formal models to represent knowledge that provides machine-readable semantics.

An information model is a representation of the characteristics and behavior of a component or system that is independent of vendor, platform, language, and repository. In contrast, a data model is an implementation of the characteristics and behavior of a component or system using a particular vendor, platform, language, and/or repository. We use a DEN-ng information model [4] and a directory enabled networks-next generation ontology (DENON-ng) ontology model in order to translate between the different data models (e.g., a directory and a relational database) used by different applications. Knowledge is defined once in the DEN-ng information model and then bound to one or more specific forms via one or more specific data models by using ontological reasoning to ascertain and discover inconsistencies and incompatibilities between data models.

While models are important for representing knowledge, they are not sufficient for the implementation of the proposed architecture. They do not contain the constructs necessary to support the definition of knowledge or reasoning about knowledge, which require formal semantic definitions. We augment the knowledge contained in models with a set of ontologies. An ontology provides mechanisms to establish semantic relationships between different pieces of information.

We gain significant benefits by combining modeled and ontological data. Doing so enables us to reason from facts; more importantly, it forms the foundation for mapping between different schemata. It can also be used to strengthen the semantics being inferred. For example, if a hypothesis is formed, we can use additional semantic relationships to gather additional information to help prove or disprove the hypothesis. This, in effect, provides a self-check of the consistency and integrity of the information that has been collected. This is especially important when multiple data from different sources need to be integrated, since they can have different qualities (e.g., accuracy, certainty, and freshness) as well as different formats. Hence, semantic relatedness enables us to solve two important problems: (1) The harmonization of different qualities of data and (2) the alignment of different ontologies to facilitate extraction of diverse data. This, in turn, increases the accuracy and reliability of the proofs that use the ontological data. Our current implementation uses a programmable threshold for the data harmonization that enables us to weigh the contribution of different data sources; for the ontology alignment, we use standard and custom semantic equivalence relationships.

### *B. Extended FOCALE Control Loop*

HiMang is built in accordance with FOCALE [4] control loops; it is also self-governing. The system senses changes in itself and its environment and determines the effect of the changes on the currently active set of business policies. As shown in Fig. 2, the FOCALE control loops operate as follows. Sensor data are retrieved from the managed resource (e.g., a router) and fed to a model-based translation process, which translates vendor- and device-specific data into a normalized form in XML for use as reference data using the DEN-ng information model and ontologies. These data are then analyzed to determine the

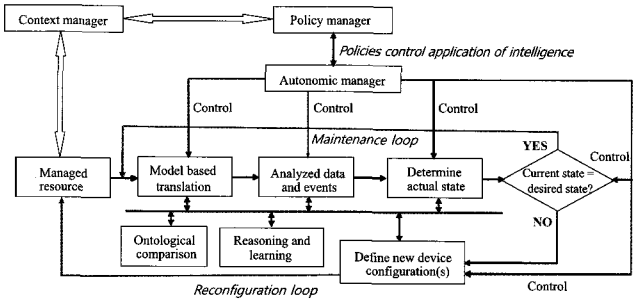



Fig. 2. Simplified version of the FOCALE autonomic architecture.

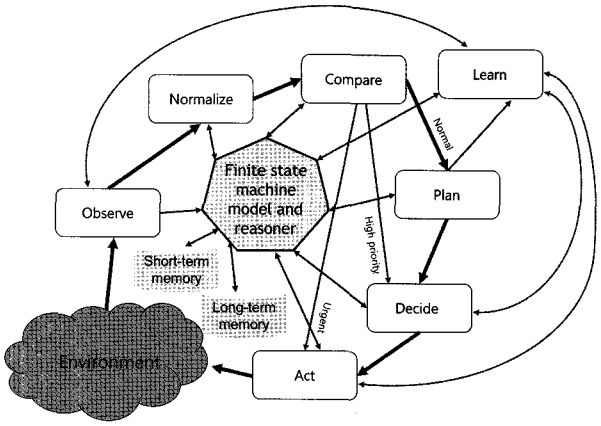



Fig. 3. FOCALE cognitive model.

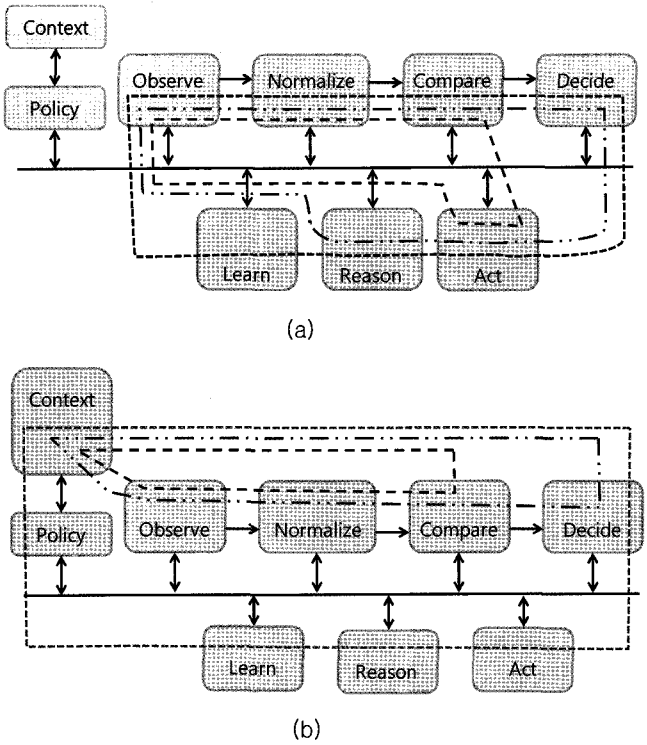



Fig. 4. New FOCALE control loops: (a) Outer control loop and (b) inner control loop.

current state of the managed entity. The current state is compared to the desired state obtained from the appropriate finite state machines (FSMs). If no problems are detected, the system continues using the maintenance loop; otherwise, the reconfiguration loop is used so that the services and resources provided can adapt to these new needs.

In order to strengthen its self-awareness, the new FOCALE cognition model employs a model of human intelligence. The cognition model is built using simple processes, called agents, that interact with each other in accordance with the three layers-the reactive, deliberative, and reflective layers-that we mentioned previously.

The new FOCALE cognition model employs the cognitive processes shown in Fig. 3. Since all processes use the FSM and reasoner, the system can recognize whether an event or set of events has been encountered before. Such results are stored in short-term memory. This reactive mechanism enables many of the computationally intensive portions of the control loop to be bypassed, producing two shortcuts labeled "high priority" and "urgent." The deliberative process is embodied in the set of bold arrows, which takes the observe-normalize-compare-plan-decide-act path. This process uses long-term memory to store how goals are met on a context-specific basis. The reflective process examines the different conclusions made by the set of deliberative processes that are being used and tries to predict the best set of actions that will maximize the goals being addressed by the system. This process uses semantic analysis to understand why a particular context was entered and why a context change occurred to help predict how to more easily and efficiently change contexts in the future. These results are also stored in long-term memory, so that the system can better understand contextual changes in its reasoning to aid debugging.

The new cognition model makes a fundamental change to the FOCALE control loops, as shown in Fig. 4. Instead of using two alternative control loops-one for maintenance and the other for reconfiguration-that are both reflective in nature, the new FOCALE architecture uses a set of outer control loops that affect a set of inner control loops. The outer control loops are used for "large-scale" adjustment of functionality in reaction to context changes. The inner control loops are used for more granular adjustment of functionality within a particular context. In addition, both the outer and the inner control loops use reactive, deliberative, and reflective reasoning, as appropriate.

The inner loops have two important changes. First, the decide function has been explicitly unbundled from the act function. This enables additional machine-based learning and reasoning processes to participate in determining which actions should be taken. Second, the new cognition model has changed the old FOCALE control loops, which were both reflective in nature, to the three control loops shown in Fig. 4, which are reactive, deliberative, and reflective. Fig. 4 shows the new outer loops of FOCALE. The reactive loop is used to react to context changes that have previously been analyzed; the deliberative loop is used when a context change is known to have occurred, but its details are not sufficiently well understood to take an action; and the reflective loop is used to better understand how context changes are affecting the goals of the AE.

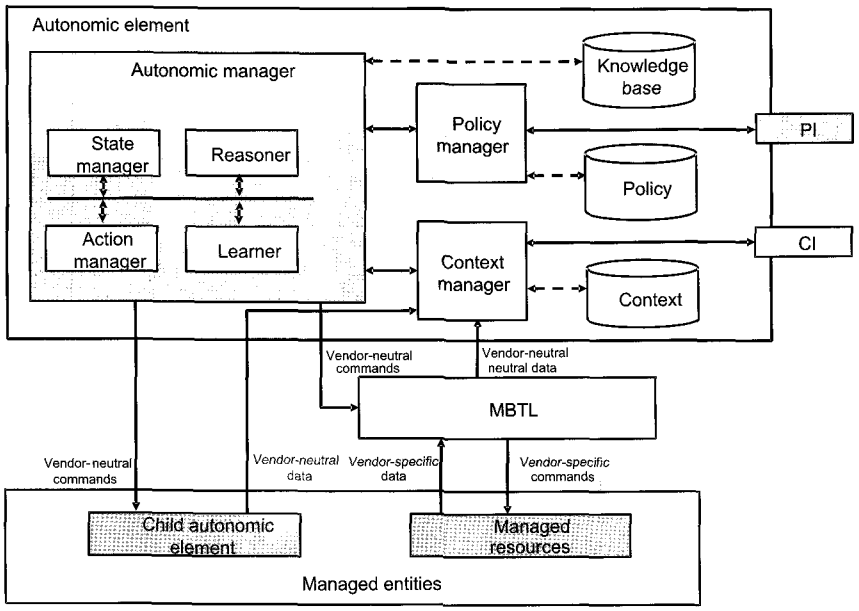



Fig. 5. Block diagram of autonomic element.

### C. Hierarchical Autonomic Management

An AE is an abstraction that allows FOCALE to provide distributed functions such as communication, learning, reasoning, and management. AEs are grouped together in cooperating communities, or clusters. An AE is a managed entity itself and can have a hierarchical relationship with other AEs, just like any other management entity. A parent AE can have several child AEs and other managed entities. In the future, network devices may have more intelligence and be able to manage themselves, so routers or switches could also be AEs. However, current existing network devices have no functionality for managing themselves, and thus, an AE is an agent that manages the existing network devices.

Fig. 5 illustrates a block diagram of an AE. It is composed of an autonomic manager (AM), a context manager (CM), a policy manager (PM), and a knowledge base (KB). An AE can manage a child AE or managed resource (MR) as a management entity (ME). When an AE manages a MR, a model-based translation layer (MBTL) translates vendor-specific data and commands to vendor-neutral data and commands (or vice versa), because MRs can be devices from various vendors. A CM collects data from an ME and stores it in a context repository. Other AEs access context information via a context interface (CI). A PM stores available policies in a policy repository, and other AEs configure policies via a policy interface (PI). An AM decides on appropriate actions for the current context using a state manager, action manager, learner, and reasoner based on the FOCALE cognition control loops. The chosen action is sent to the ME. HiMang selects one of the three control loops (reactive, deliberative, and reflective control loops) based on the current state and collected context.

As discussed above, HiMang uses a hierarchical autonomic network management architecture to reduce complexity. Fig. 6 shows the structure of a hierarchical AE and gives an example of how the hierarchy can be mapped to the physical infrastructure. The autonomic network domain manager (ANDM), autonomic network managers (ANMs), and autonomic element managers (AEMs) in the figure are AEs. We designed this hierarchy based on the DEN-ng information model, which describes a network management hierarchy. A parent AE governs the management decisions of its child AE such as coverage of monitoring and performance metrics to be measured. However, child AEs also have their own autonomic decision-making capabilities. FOCALE cognition control loops play a role in the main AE decision engine, and monitoring data is sent by child AEs or managed entities. A number of AEs should be distributed in the network. When there is only one AE in the network, this model is the same as a traditional client-server model and has similar problems, such as a single point of failure, because all the management information is delivered to a single AE.

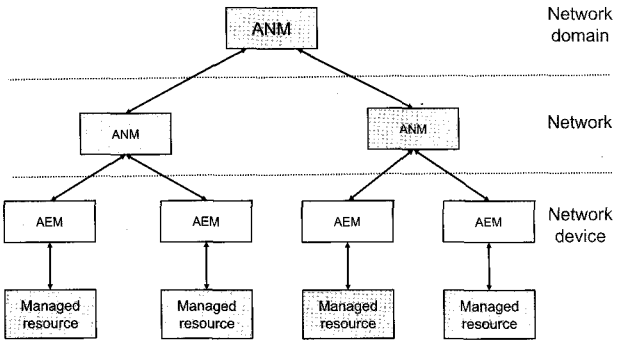



Fig. 6. Example of autonomic element hierarchy.

### D. Implementation Architecture

Based on the design principles and conceptual architecture described above, we designed an implementation architecture

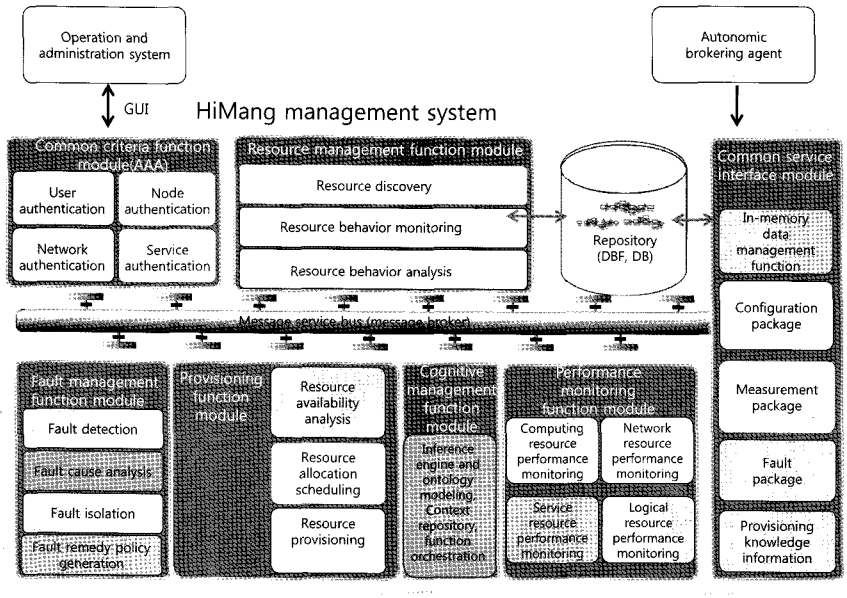



Fig. 7. HiMang implementation architecture.

for verifying the feasibility of this system, as shown in Fig. 7. It consists of eight modules: The provisioning, resource management, performance monitoring, fault management, cognitive management, authentication, authorization, and accounting (AAA) management, common service interface (CSI), and information repository modules. The main criteria of our design choice are scalability, extensibility, efficiency, and reliability of the implementation, as described in our design principles. Thus, we tried to modularize the system's functionality as much as possible and, at the same time, centralize the common information (e.g., CSI) to ensure the system's efficiency.

All the modules share a publish/subscribe-type common message bus. The CSI plays a role of in-memory common management information storage, and the repository serves as its associated persistent storage as well as a database for all management function modules. All the modules depicted in Fig. 7 belong to the management and inference planes in the HiMang conceptual architecture. The cognitive management function (CMF) implements the AEs in the inference plane. AEs are also implemented in the other planes either as a part of the management functions or embedded in the element itself (e.g., router or switch). Since the management functionalities are complex, we applied hierarchical management for scalability and the CORD mechanism to ensure the system's reliability, as described in subsection III-B. The manager modules interact with an autonomic brokering agent to retrieve or configure management data/information or provisioning rules. The brokering agent implements a MBTL, as described in the Fig. 5, to support heterogeneity and backward compatibility.

## V. PROOF OF CONCEPT

In this section, we provide three example use cases demonstrating how the HiMang architecture can be instantiated into realistically demanding management environments. The first case enables automation of management for cloud-computing data

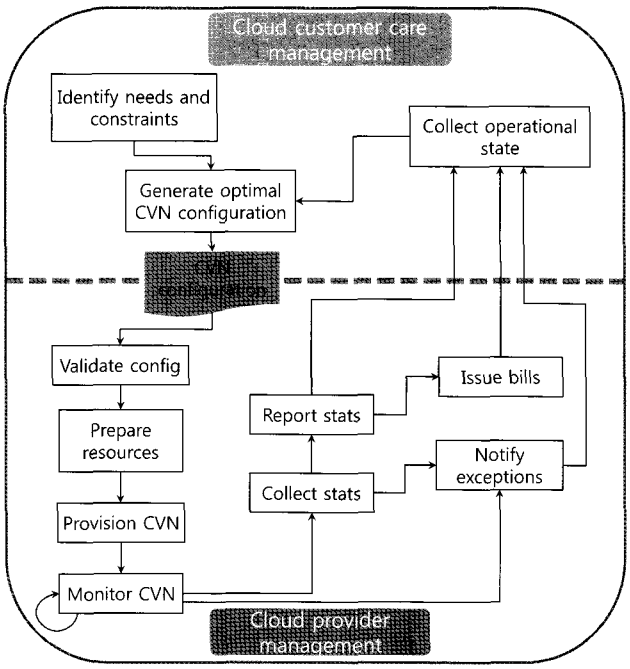



Fig. 8. High-level task flow for CVN management.

centers (CDCs). We assume that, unlike existing CDCs, the CDCs in this example support network virtualization as well as computing resource virtualization. We define such an enhanced CDC as a cloud-based virtual network (CVN). Among various management tasks, this use case focuses on optimal resource provisioning based on real-time accurate resource status monitoring. Our novel contributions in this use case are twofold: Autonomicity-capable CVN management and optimal CVN resource allocation. The second use case describes how HiMang architecture can be applied to the management automation of a mission-critical advanced content distribution network (CDN) called smart network. One of the major network operators in

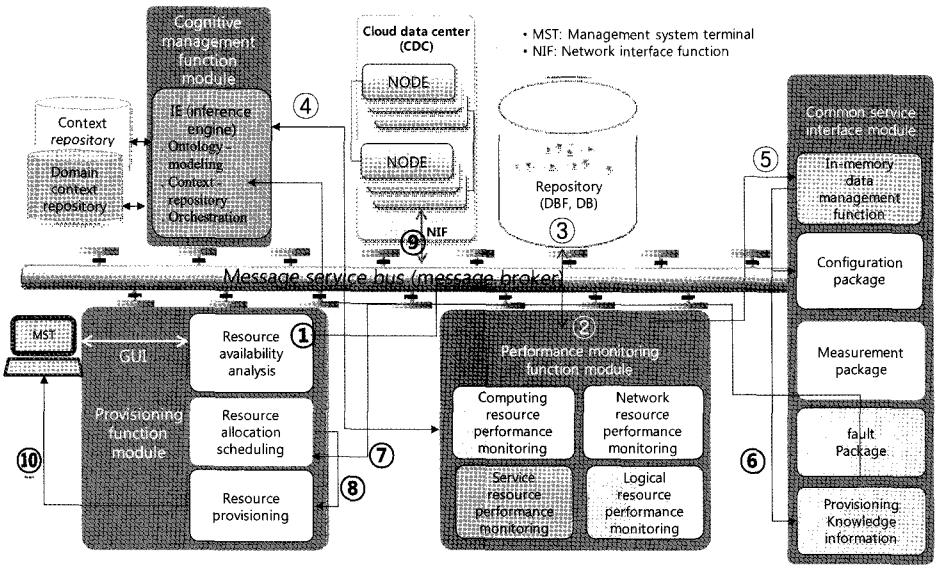



Fig. 9. HiMang implementation mapping from task flow for CVN management.

Korea is preparing a trial of this service, and our use case will be a part of the network management in this trial. Finally, as a third use case, we describe autonomic fault management for legacy enterprise IP networks; this case demonstrates the backward compatibility of our HiMang architecture.

### A. *Autonomic Management of Cloud-Based Virtual Network*

Cloud computing promises to reshape the way IT service is produced and consumed by virtualizing computing resources. As cloud service providers offer virtualized CPU and storage resources via machine virtualization and distributed storage technologies, respectively, they wish to offer to their customers virtual networks running on the cloud. With CVNs, cloud customers (often, corporate customers) can easily build new sites, effectively expanding their enterprise networks into the cloud and thus leveraging all the benefits of cloud computing (agility, manageability, low cost, etc.). Because of the various new features introduced by CVNs, we face significant technical challenges in managing CVNs.

Fig. 8 shows the high-level task flow for CVN management. To create a CVN, a customer first identifies the needs and constraints for the CVN, such as the CVN's size (number of member virtual machines (VMs)), IP-address prefixes to use for the VMs, connectivity to the customer's existing enterprise network or other CVNs, routing information, host names, etc. Then, on behalf of the customer, the provider translates the needs and constraints into a specific CVN configuration instance, issuing a "CVN-creation" request. Given the CVN configuration instance, the CVN provider first validates the configuration. When the configuration passes the validation step, the provider runs a resource-allocation algorithm and chooses the right set of resources that can accommodate the "CVN-creation" request. Finally, the provider deploys and provisions all the needed resources: The provider instantiates VMs, configures the virtual network interfaces of the VMs, assigns IP addresses, sets up the correct performance and reachability policies, etc.. The functional blocks above the bold dotted line are specifically devoted to cloud customer care management. Such management capability can be packaged and provided as a management service as a kind of platform as a service (PaaS) for large-scale enterprise customers who want to have more sophisticated control of their cloud resources.

Fig. 9 shows how such a CVN management task flow can be realized in the HiMang implementation architecture. As the first step, a customer's provisioning request is made through a graphic user interface (GUI). The resource availability function within the provisioning function module verifies that the resources are available by using data collected by the performance monitoring function module (PMFM). The PMFM collects computing, network, service, and logical resource information and stores it in both the CSI for real-time access and database (DB) repository for persistency purposes. During the resource availability check, the cognitive management function module assists in inferring the optimal resource selection. The resource allocation function then calculates optimal resource mapping rules. These are then enforced in the provider's selected computing and network elements. Once provisioned, the AE continuously monitors any abnormalities, including quality degradation, and performs the necessary remedies autonomically with the help of the cognitive control loop function.

In order to illustrate an optimal resource allocation algorithm, we assume that the customer provides a simple network topology, as shown in Fig. 10, to be run on the cloud. These types of requests from the customers are common in infrastructure-as-a-service (IaaS)-type clouds.

In Fig. 10, the vertices represent the nodes that the customer needs, and the vectors inscribed in the circle represent the CPU and memory requirements in MHz and Gbytes, respectively. The numeric line labels represent the bandwidth requirements between the nodes in Gbps.

A typical L2 cloud topology can be viewed as a three-tiered architecture. The bottommost tier consists of the virtual bridges in the computing servers, which have the least capability (i.e., small bandwidth and high latency). The second tier is made up

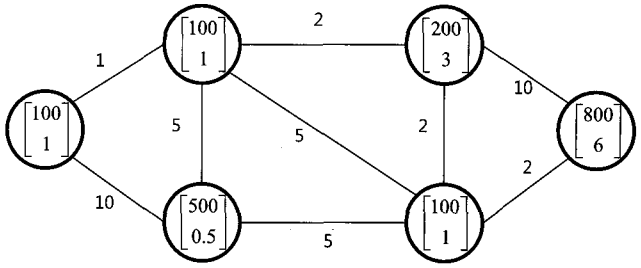



Fig. 10. Virtual network topology.

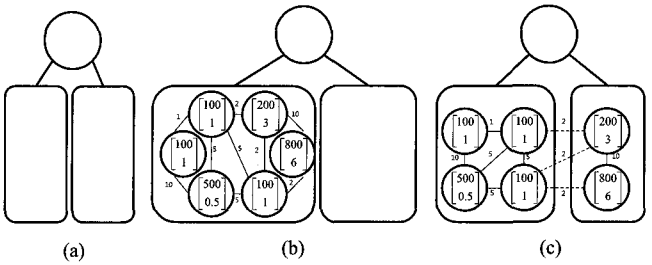



Fig. 11. Simple cloud topology abstraction and provisioning: (a) Base structure of cloud topology, (b) provisioning on a single cluster, and (c) provisioning on two clusters using the minimum graph cut.

of top-of-rack (ToR) switches; the topmost tier is an aggregation switch. The aggregation switch has the most capacity, and the virtual bridges have the least capacity.

As in any hierarchical topology, traffic gets aggregated at the aggregation switch, and this frequently becomes the performance bottleneck. Thus, in our proposed provisioning algorithm, we try to minimize the amount of traffic that gets forwarded to the aggregation link. As an example, consider provisioning the virtual network topology shown in Fig. 10 onto the simple cloud topology shown in Fig. 11(a). In Fig. 11(a), the circle and the two squares could represent either an aggregation switch and clusters attached to the aggregation switch or a ToR switch and computing servers. For the following discussion about the provisioning, let the circle represent an aggregation switch and the squares, clusters.

First, consider solving the provisioning problem when the sole consideration is minimizing the amount of traffic to the aggregation switch. That is, if any other constraints (computing power, memory requirements, etc.) are omitted, the best provisioning scheme is to provision the entire virtual topology onto a single cluster, as shown in Fig. 11(b). In this provisioning scheme (in Fig. 11(b)), no traffic to the aggregation link is generated.

Once the relaxed constraints are introduced back into the provisioning problem, it may become impossible to provision the virtual network topology onto a single cluster because of limited CPU power or limited memory. In this case, an efficient way of distributing the virtual topology onto two clusters is necessary. It is easy to see that to minimize the traffic at the aggregation links (minimize the intercluster communication), the virtual network topology should be partitioned so that the communication between the two partitions is minimized. This partitioning can be achieved by using the minimum cut operation from graph theory using the link bandwidth as the cost. This provisioning is shown in Fig. 11(c).

The next natural problem in the provisioning problem is how to map a partitioned graph within a cluster. Once again, for this problem, the provisioning algorithm tries to place the entire virtual network topology onto a single rack, if this is not possible, then the minimum cut algorithm can be applied to the partitioned graph to distribute it among the racks.

The last phase of the provisioning algorithm is allocating computing servers within a rack to the partitioned graph. Again, if a single computing server can host the partitioned graph, a single computing server will be assigned. If not, the partitioned graph will be partitioned again to distribute the graph into two computing servers. As in previous steps, the idea behind the partitioning is to minimize the traffic to the ToR switch.

In any phase of the provisioning algorithm, the virtual network topology is always assigned to the cluster/rack/computing server with the fewest available resources (i.e., computing power and memory), as was done in [21]. This approach increases the resource use efficiency. Another intuition behind this approach is that by keeping more resources available, they can be used more flexibly later to provision a future virtual network topology request; thus, more requests can be served.

Even though application of the minimum cut operation to the virtual network topology during provisioning reduces the amount of intercluster traffic, the minimum cut algorithm does not provide any information about the intracluster communication bandwidth. The cut-clustering algorithm [22], which can be used instead of the minimum cut algorithm, provides an upper bound to the intercluster communication bandwidth and a lower bound to the intracluster communication bandwidth.

Once a CVN is provisioned, the provider continuously monitors the CVN's state and collects the appropriate operational logs needed for charging and reporting. Upon an occasional exception, the provider immediately notifies the customer of the state. In addition, the CVN customer can also poll the CVN's state on demand.

The types of information a provider can offer to a customer include but are not limited to the following:

- CVN's operational state: Up-time, etc.
- Usage: Traffic volume (per class), storage footprint, etc.
- Performance: Latency, loss rate, etc.
- Failure notification.
- Exceptions: Policy violation incidents, customer-defined alarms, etc.

Fig. 12 illustrates how autonomicity can be enabled for CVN management. It illustrates a CVN data center autonomic management scenario. The device-level manager has an autonomic management function (AMF) that implements a control loop for autonomic device-level management capabilities, such as switch/server configuration, link management, device failure management, and physical capacity management. The network-level manager also has its own AMF that implements a control loop for autonomic network-level management capabilities, such as traffic monitoring, performance management, security management, network-level failure management, usage and cost management, and virtual capacity management.

Network- and device-level AE managers cooperate to enable autonomic management of a CVN data center. The network-

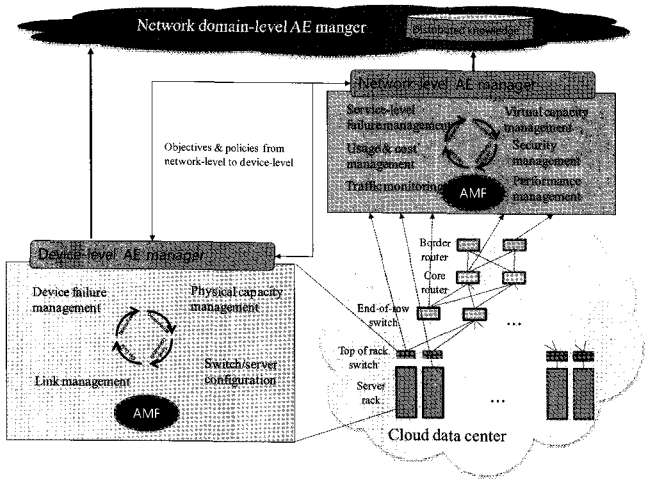



Fig. 12. Autonomic CVN management.

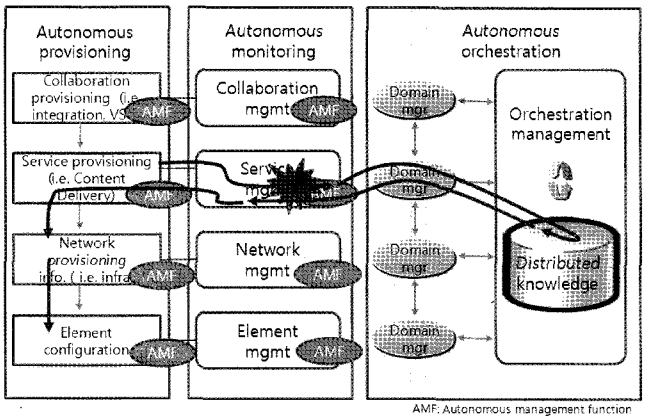



Fig. 13. Autonomic smart network and service management.

level manager decides on autonomic management policies and pushes them to the device-level manager for enforcement. The device-level manager also provides its own management information to the network-level manager for further analysis and decision making. Both managers send or retrieve rules, which requires cross-level coordination via the network-domain-level manager. The network-domain-level manager also contains a common knowledge repository in the form of an ontology and the associated cognition and inference engine. In addition, note that the autonomic management components are structured in a hierarchy, which simplifies the interaction between components and allows them to manage resources and govern child components in a more scalable manner. The advantages of this approach have been validated through analytical evaluation results, which are described in detail in [14].

### *B. Autonomic Management of Smart Network and Service*

Current networks face the following serious imminent challenges from the perspective of telecommunication operators:

- Supporting variety of new applications with different requirements for QoS, security, and mobility with increased diversity in user behavior and end-user device capabilities;
- Taking into account the dependency of services/applications in a local context with/without individual preferences and considering the impact to delivery of such services/applications; and
- Balancing between capital expenditure and operating expenditure.

Merely extending the capacity of current networks or simply replacing them with more powerful equipment, as today's network operators often do, will not be sufficient to address these challenges. One of the major telecommunications companies in Korea believes that this is the time to abandon its past ways of simply upgrading its network and to enhance its network capabilities or develop enhanced networks. In this way, it hopes to provide optimized and efficient use of available network/service resources and dynamically adapt existing resources in a smart way. For this purpose, they have come up with a new network called the "smart network."

Smart networks are packet-based networks (e.g., IP) that can transport and deliver a wide range of existing and emerging services to people and things. The services provided by the networks can cover aspects such as control, processing, and storage. The networks are smart in the sense that they are knowledgeable, context-aware, adaptable, autonomous, and programmable. In a smart network, a new type of node called a smart node is introduced. It adds intelligent media processing and storage capabilities on top of the existing data forwarding and routing capabilities. It also realizes intelligent knowledge and context-awareness by adding a new element called a smart gateway that provides functionalities like the application-layer traffic optimization (ALTO) functionality defined by Internet engineering task force (IETF). The major target service is high-quality multimedia content delivery, such as high definition (HD) video live streaming or truly seamless triple-play service.

Although building and deploying such networks are quite challenging, management, and ranging from service provisioning up to its assurance, is even more so. In this use case, HiMang attempts to provide an autonomic management solution for smart networks and their services. Fig. 13 illustrates autonomic provisioning and performance monitoring for smart networks. The event flows shown represent the process that occurs from the detection of a service-related fault up to the self-healing. When smart service quality degradation occurs, the management system identifies the root cause by consulting the orchestration manager, which manages distributed knowledge. This process may require multilayer/multidomain correlation of information. Once the root cause is identified, the management system generates a remedying policy and converts it into new re-provisioning rules that result in autonomic network and element-level configuration.

### *C. Autonomic Fault Management*

As a third use case, we describe autonomic fault management for legacy enterprise IP networks in order to ensure backward compatibility of our HiMang architecture. First, we introduce a target network environment. Next, we present how to apply our HiMang architecture to manage it. Finally, we present different control loops for managing different types of faults.

Fig. 14 shows a target network and how to manage it using

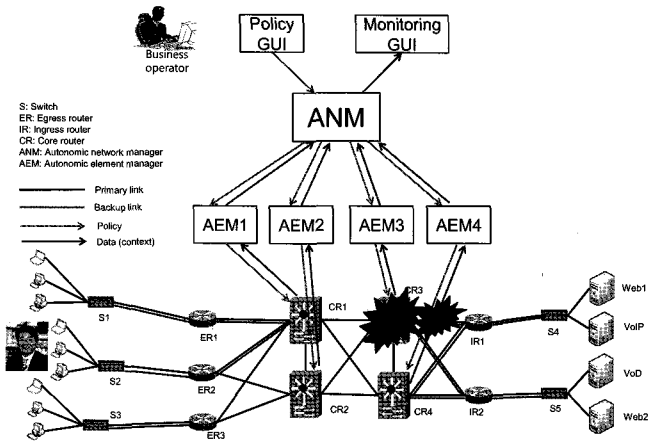



Fig. 14. Autonomic fault management using HiMang architecture.

our HiMang architecture. The network uses DiffServ to manage QoS. Marking and policing of traffic is performed at the edge routers (ERs). Each service flow for each end user is identified at the ER by an access control list made up of source and destination IP addresses and port numbers. Four core routers (CRs), two ingress routers (IRs), and three egress routers (ERs) are connected by primary links and backup links. All servers and clients are connected to each switch (S). In terms of applications, the end users are using web, video on demand (VoD), and voice over IP (VoIP) services based on their own service-level agreements (SLAs). In such an environment, we describe how to manage four CRs using the HiMang architecture. As shown in Fig. 14, each AEM manages a corresponding CR. If CRs made by different vendors, such as Cisco, Juniper, or some other CRs, are to be used together, the AEM uses a MBTL to translate the management information and commands to technology-neutral data and commands. In this example, a parent ANM manages four AEMs. Red dotted arrows represent policy flows, and blue arrows represent context flows.

Initially, a business operator configures a management goal using an SLA, which is a contract between a customer and an operator. For example, the current policy configured by a business operator may be as follows: "James gets gold service." This means that James is subscribed to gold service and that this network has to provide services to James based on configurations for gold service, including the appropriate bandwidth allocation. The business operator monitors the current network status using a monitoring GUI.

Our HiMang architecture translates a high-level business policy to a low-level system configuration based on a policy continuum. In the network and system view, "James" is translated to a unique ID, "gets" is translated to "subscribe," and "gold service" is one of the service packages: Gold, silver, and bronze. In the device view, the unique ID is translated to an IP address, "subscribe" is translated to a device configuration, and each service package is translated to device-level configurations, such as a bandwidth configuration. In the initial management of the target network, the business operator would monitor the high-level status of the network, whereas the HiMang management system monitors and solves problems using the new FOCALE cognition control loops. For example, if a primary link between CR3 and CR1 is down, AEM3 gets the "linkdown" trap from CR3 and determines whether it can solve this problem or not. Because AEM3 knows a backup link between CR3 and CR1, it changes the configuration to from the primary link to the backup link by itself without notifying the parent ANM. Otherwise, if CR3 is down and AEM3 cannot solve the problem, AEM3 notifies the parent ANM of this problem.

The ANM uses an inner control loop in the "CR3 is down" context. If this problem is known and urgent, the ANM uses a reactive control loop to solve the problem and provide an appropriate policy to AEM3. However, if this problem is unknown and the ANM needs management information from other AEMs, the ANM uses a deliberative control loop to select an alternative routing path for the current service. In this case, if the ANM needs more information, it uses a reflective control loop to learn and make inferences. Determining the optimal routing path is outside of our scope here; we can use known algorithms to solve this problem. Although this use case illustrates an example of an application of HiMang in a small enterprise environment, the same concept and mechanisms described above can be applied in real-world large-scale cases, as well. The feasibility of applying HiMang to these environments in terms of scalability, performance, and robustness remains to be evaluated, and we intend to pursue this in our near-term future work.

## VI. CONCLUSION

In this paper, we have identified core design principles and key characteristics for managing the future Internet. We then defined the HiMang architecture based on those design principles. Finally, we have proposed a proof-of-concept system with three real-life deployment use cases. Our design principles include several novel ideas, such as cognitive management, a policy continuum, hierarchical management, application placement, high-availability, and optimal resource allocation.

In our resource allocation algorithm study, even though application of the minimum cut algorithm to the virtual network topology reduces the amount of inter-cluster traffic, the minimum cut algorithm does not provide any information about the intra-cluster communication bandwidth. The cut-clustering algorithm [22], which can be used instead of the minimum cut algorithm, provides an upper bound on the inter-cluster communication bandwidth and a lower bound on the intra-cluster communication bandwidth. We are currently analyzing the improvement in performance when the cut-clustering algorithm is used instead of a simple minimum cut algorithm.

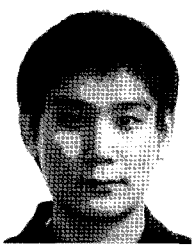
Our algorithm has not considered an important aspect of the QoS requirement: Network delay. We are currently trying to improve our provisioning algorithm to incorporate network delay. In addition, we are also evaluating the feasibility of the cognitive processes defined in our new cognition model in terms of performance.

## REFERENCES

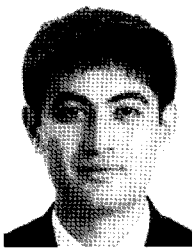
[1] P. Dimitri and Z. Theodore, "Future Internet design principles," FP7 Future Internet Architecture Working Group, Mar. 2011.
[2] J. Strassner, "Autonomic networking-theory and practice," in *Proc. NOMS 2008 Tutorial*, Brazil, Apr. 7, 2008.
[3] AUTOI. [Online]. Avaliable: http://ist-autoi.eu

[4] J. Strassner, N. Agoulmine, and E. Lehtihet, "FOCALE–a novel autonomic networking architecture," in *Proc. ITSSA J.*, vol. 3, no. 1, May 2007, pp. 64–79.
[5] J. Strassner, "Policy based network management," *Morgan Kaufman*, 2004.
[6] 4WARD. [Online]. Avaliable: http://www.4ward-project.eu
[7] FAME. [Online]. Avaliable: http://www.fame.ie
[8] J. Strassner, "Introduction to DEN-ng," Tutorial for FP7 PanLab II Project, 2009.
[9] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, and J. Rexford, "A clean slate 4D approach to network control and management," in *Proc. SIGCOMM CCR.*, vol. 35, no. 5, 2005.
[10] P. Francis and J. Lepreau. Towards complexity-oblivious network management. NSF NeTS-FIND Initiative. [Online]. Avaliable: http://www.nets-find.net/Funded/Towards Complexity.php
[11] FP6 Situated Autonomic Communications and other relevant FET Projects. [Online]. Avaliable: http://cordis.europa.eu/fp7/ict/fire/future-internet-projects_en.html
[12] J. Strassner, J. W. Hong, and S. van der Meer, "The design of an autonomic element for managing emerging networks and services," in *Proc. ICUMT*, St. Petersburg, Russia, Oct. 2009.
[13] M. Minsky, *The Society of Mind.* Simon and Schuster, New York:NY, 1988.
[14] J. Famaey, S. Latré, J. Strassner, and F. De Turck, "A hierarchical approach to autonomic network management," in *Proc. IEEE/IFIP NOMS*, Apr. 2010, pp. 225–232.
[15] B. Lee, Y. Jeongm, H. Song, and Y. Lee, "A scalable and highly available network management architecture," in *Proc. IEEE GLOBECOM* 2010.
[16] M. Choi, J. W. Hong, and H. Ju, "XML-based network management for IP networks," *ETRI J.*, vol. 25, no. 6, pp. 445–463, Dec. 2003.
[17] A. Feldmann, "Internet clean-slate design: What and why?," *IEEE/ACM SIGCOM Computer Commun. Rev.*, vol. 37, no. 3, July 2007.
[18] M. Blumenthal and D. Clark, "Rethinking the design of the Internet: The end to end arguments vs. the brave new world," *IEEE/ACM Trans. Internet Technol.*, vol 1, no. 1, pp. 70–109, Aug. 2001.
[19] J. Strassner, M. Ó FoghlÃº, W. Donnelly, and N. Agoulmine, "Beyond the knowledge plane: An inference plane to support the next generation Internet," in *Proc. IEEE GIIS*, July 2–6, 2007, pp. 112–119.
[20] S. Kim, Y. Won, M. Choi, J. Hong, and J. Strassner, "Towards management of the future Internet,"in *Proc. IEEE/IFIP ManFI, Long Island, NY, USA*, June 5, 2009, pp. 1–6.
[21] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A scalable application placement controller for enterprise data centers," in *Proc. Int. Conf. World Wide Web*, 2007, pp. 331–340.
[22] G. Flake, R. Tarjan, and K. Tsioutisiouliklis, "Graph clustering and minimum cut trees," *Internet Mathematics*, vol. 1, no. 4, pp. 385–408, 2004.

**Taesang Choi** is a Principal Engineering Staff in ETRI, having joined the institute in 1996 after research and development careers on network and service management of telecommunications during his Ph.D. studies at the University of Missouri at Kansas City. He has successfully managed a number of projects in the area of telecommunications and networking technologies, especially in Internet traffic engineering, traffic measurement and analysis, QoS management, and future Internet management. He has also been actively contributed to various standardization organizations such as ITU-T and IETF in the area of Internet traffic engineering, Internet traffic measurement and analysis, and future Internet since 1993. He is currently serving as a Rapporteur of ITU-T SG13 Q.4 and a project Initiator of FI management architecture project, HIMang.



**Tae-Ho Lee** received his B.S. degree from University of Texas at Austin and M.S. degree from Stanford University all in Electrical Engineering. Since 2009, he has joined Electronics and Telecommunications Research Institute (ETRI) as a Researcher. His major research interests include mathematical modeling and optimization, and its application in networks.



**Nodir Kodirov** received B.Sc. in Computer Science from 2004 to 2008 at Tashkent University of Information Technologies, in Tashkent, Uzbekistan. He received M.Sc. in the same major from 2008 to 2010 at Konkuk University, Seoul, Korea. Currently, he is working as a Researcher on Network Management domain at Electronics and Telecommunication Research Institute (ETRI), Korea. His research interests include cloud resource management and future Internet.

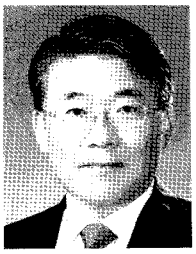
**Jaegi Lee** received his B.S. degree from Seoul National University of Science Technology and M.S. degree from Cheong-Ju University (1988) and Ph.D.(2004) in National Kong-Ju University. Since 1983, he has joined Electronics and Telecommunications Research Institute (ETRI) as a Researcher. His major research interests include software reliability and testing, and future Internet. He is a Member of KICS, IEEK, KIPS, and KCA.

**Doyeon Kim** received the M.S. degree in Electronic Engineering from the Chongju University of Cheongju, Korea, in 1990. He is currently Principal Engineer at the ETRI. He was an invited Visiting Research Fellow at the Department of Computer Science, Georgia Southern University, USA, from July 2006 to June 2007. His major research interests are in network and service management for future Internet and ethernet based on the packet transport network.

**Joon-Myung Kang** received his B.Sc. and Ph.D. in Computer Science and Engineering from POSTECH, Pohang, Korea in 2005 and 2011, respectively. From 2000 to 2004, he worked at Alticast Corporation, as a software engineer. He was a Postdoctoral Felllow in the Department of Computer Science and Engineering at POSTECH, Pohang, Korea from March 2011 to September 2011. Currently, he is a Postdoctoral Fellow in the Department of Electrical and Computer Engineering at Uniersity of Toronto, ON, Canada. His research interests include autonomic network management, mobile device management, mobility management, personalized services, and software product lines.

**Sungsu Kim** is Ph.D. student in the Department of Computer Science and Engineering at POSTECH, Pohang, Republic of Korea. He received a B.S. in the Department of Electrical and Computer Engineering from Sungkyunkwan University in 2007. His research interests include autonomic network management and semantic peer-to-peer network.

**John Strassner** is a Professor in the Division of IT Convergence Engineering in POSTECH, and leads its autonomic computing group. Previously, he was a Visiting Professor at Waterford Institute of Technology in Ireland, where he worked on various FP7 and Irish research programs. Before that, he was a Motorola Fellow and Vice President of Autonomic Research at Motorola Labs, where he was responsible for directing Motorola's efforts in autonomic computing and networking, policy management, and knowledge engineering. Previously, he was the Chief Strategy Officer for Intelliden and a former Cisco Fellow. He is the Chairman of the Autonomic Communications Forum, and the past Chair of the TMF's NGOSS SID, metamodel and policy working groups, along with the past chair of several IETF and WWRF groups. He has authored two books (Directory Enabled Networks and Policy Based Network Management), written chapters for 5 other books, and has been coeditor of 5 journals dedicated to network and service management and autonomics. He is the recipient of the IEEE Daniel A. Stokesbury memorial award for excellence in network management, the Albert Einstein award for innovation in high technology, is a Member of the Industry Advisory Board for both University of California Davis and DePaul University, a TMF Fellow, and has authored over 235 refereed journal papers and publications. He has 47 patents. He holds B.S.E.E., B.S.C.S., M.S.C.S., and Ph.D. degrees.



**James Won-Ki Hong** is a Professor and the Head of Division of IT Convergence Engineering, POSTECH, Pohang, Korea. He received a Ph.D. degree from the Univ. of Waterloo, Canada in 1991. His research interests include network management, network monitoring and analysis, convergence engineering, ubiquitous computing, and smartphonomics. He has served as Chair (2005-2009) for IEEE ComSoc Committee on Network Operations and Management (CNOM). He is serving as Director of Online Content for the IEEE ComSoc. He is a NOMS/IM Steering Committee Member and a Steering Committee Member of APNOMS. He was General Chair of APNOMS 2006 and General Co-Chair of APNOMS 2008 and APNOMS 2011. He was a General Co-Chair of 2010 IEEE/IFIP Network Operations and Management Symposium (NOMS 2010). He is an Associate EiC of IJNM and Editorial Board Member of IEEE TNSM, JNSM, JCN, and JTM.