Chapter 14
An Intelligent Model for Distributed Systems in Next Generation Networks

Pakawat Pupatwibul, Ameen Banjar, Abdallah AL Sabbagh, and Robin Braun

Abstract. Over the past twenty years, network technology has been improved rapidly in term of speed, performance, component, and functionalities. Therefore, a number of different types of network devices have been developed; this led to an increase in the complexity of network systems. Traditional network structures are inadequate to meet today requirements. It is centralized network which imposes on human operators to have a high experience on how to detect changes, configure new services, recover from failures and maximize Quality of Service (QoS). Therefore, network management involves heavy reliance on expert's operators. The adopted centralized network management is not suitable for new technologies emerging, which are complex and difficult to interact among heterogeneous networks that contain different types of services, products and applications from multiple vendors. As a result, the current network management lacks of efficiency and scalability; however, it has an acceptable performance generally. The centralized information model cannot stand and achieve the requirements from such complex, distributed electronic environments. This paper studies the need of distributed systems in next generation networks. Then, the paper presents three network structure paradigms: centralized, hybrid and distributed. After that, Software-Defined Networking (SDN) is described. Finally, the paper proposes a distributed approach for OpenFlow technology using a Distributed Active Information Model (DAIM) which supports an autonomic management of the distributed electronic environment.

14.1 Introduction

The increasing adoption of advanced technologies in communication networking, computing applications, and information modelling have played a significant role in

Pakawat Pupatwibul · Ameen Banjar · Abdallah AL Sabbagh · Robin Braun
University of Technology, Sydney (UTS), Sydney, Australia
e-mail: {Pakawat.Pupatwibul,ameen.r.banjar}@student.uts.edu.au,
{abdallah.alsabbagh,robin.braun}@uts.edu.au

R. Klempous et al. (eds.), *Advanced Methods and Applications in Computational Intelligence*, Topics in Intelligent Engineering and Informatics 6,

DOI: 10.1007/978-3-319-01436-4_14,

providing management services for large and complex systems. As the complexity of centralized system grows over time, an effective management requires monitoring, interpreting, and handling the behaviour of the managed resources in order to ensure required Quality of Service (QoS) and improve networks performance.

Currently, many network management systems pursue a platform-centred paradigm, where all of the computation is controlled at a central location. As an example, in traditional Simple Network Management Protocol (SNMP), a fully centralized management paradigm is used. Agents are accessed by applications via management protocol to monitor the system and collect the network information. Moreover, several researchers in the network management discipline believe that in most, if not all, network management problems can be addressed by using appropriate centralized systems and intelligence control [16]. However, in today's real networks, there are many network management complexity and limitations that cannot be adequately solved by a fully centralized approach such as lack of flexibility and information bottlenecks.

Centralized implementations are also inefficient to handle the huge number of high level decision makings. This paper compares the centralized paradigm with distributed system paradigm (decentralized), in which some or all of the intelligence and management control are locally distributed within the network entities. In wired networks, distributed system minimizes the complexity that occurs in layer 3 devices (e.g. routers) by distributing some roles into layer 2 devices (e.g. switches). OpenFlow is an example of a network that may be able to apply new distributed model on. In wireless networks, distributed system also minimizes the complexity that occurs in the core network and the Radio Network Controllers (RNC) such as in Beyond 3G (B3G) network by distributing some management functions (e.g. decision making for allocation of radio resources) into the Evolved Nodes B (eNodes B) such as in Long Term Evolution (LTE) network or into Base Stations (BSs) such as in Mobile Worldwide Interoperability for Microwave Access (WiMAX) [1],[20].

A new information model named: Distributed Active Information Model (DAIM) is presented to allow the local decision making process, that will essentially contribute to complex distributed network environments [5]. An implementation of DAIM model is expected to meet the requirements of the autonomic components of the distribution networks, such as self-management. An autonomic system in this context means that, each distributed device can draws its own strategies for adaptation driven by the goals of the system [7]. The distributed autonomic system adapts the network for needs of dynamic changing in business, and reduces operations and management complexities. Benefits that can be achieved through implementing the DAIM model include control of any network device, which OpenFlow enabled from any vendor, and rapidly configure, and update the hardware in the entire network. The approach accelerates business innovation by allowing network operators of Information Technology (IT) to program the network in real time to meet the business needs and specific requirements of the users. Accomplish this approaches is a challenging task. This is because the network control plane mechanisms take several years to fully design, and even longer to spread widely, a new control protocol. Also, it should consider that characteristics of incremental properties, complexity of

new network operators, and missing some functions in elements of the network [22]. Another part of the approach is that, distributed system needs to have a well-defined Network Requirement Database (NRD), in order to maintain self-management, self-configuration, self-preservation and recovery [5]. In addition, current networks have many restrictions, including difficulties to meet the business and technical needs over the past few decades, while industry have developed protocols for the networks to provide a high performance, reliability and greater connectivity, and security more stringent. Moreover, there are difficulties to add or remove any device or configure these devices which must be touched by an IT person that need to configure many switches, routers and firewalls using the device-level management tools [19].

As a result of these limitations, networks today are relatively constant as it seeks to reduce the risk of interruption of service. To stay competitive, next generation network must provide a higher value than ever, and provide the best customer service. A promising solution for these requirements is the autonomic distribution system [22].

The approach has not been introduced before. This is because, in the classical routing or switching, fast forward packets (data plane), and high-level routing decisions (control plane) occur on same device. In addition, vendors' devices are closed and not accessible. Moreover, if the device is not described by Management Information Base (MIB), the device does not exist. Therefore, each device has a MIB in ASCII format that you need to access and edit to achieve the new requirements [10]. From vendors' side, they have lack of standard and open interfaces, and there are limitations in the ability of network operators to design the network to meet different individual requirements. This makes a gap between market requirements and network capabilities [19].

In response, the industry has created Software-Defined Networking (SDN) architecture and develop standards associated with it. The OpenFlow is an implementation of the SDN architecture that separates data and control plane functions. The data path is still residing on the switch, while the high-level routing decisions are separating in different device called controller, usually a standard server. The OpenFlow enabled switch and the controller to communicate over OpenFlow protocol, which defined messages, such as packet-received and send-packet-out [3]. As a result, companies get the programmability, automation, and the control of the network, which enable them to build highly scalable, flexible networks that can be easily adapted to different changing environment. Recent development techniques enable dynamic reprogramming of the devices through the data flow [3]. The DAIM model based-OpenFlow gives the distributed system environment a sustainable information model, which collects, maintains updates and synchronizes all the related information. Each device has decision-making ability on the basis of the information that is collected to autonomously adapt any changing environments [5]. The autonomic approach of distribution system leads to rapid innovation through the ability to provide network capabilities and new services without having to configure individual devices or wait for the launch from the seller. In recent times, there is a growing movement led by both industry and academia, aim to design mechanisms

to reach a control model in which the separation of control plane from data plane and built it as a distributed system [22].

Thus, this paper describes the distributed system as a next generation network, which could meet the self-x paradigm. In addition, the paper covers a number of functionalities such as the DAIM model, support autonomic management for distributed system, and OpenFlow capabilities. The remainder of the paper is organized as follows. In Sect. 14.2 we demonstrate the needs of distributed systems. In Sect. 14.3 we present the different network structure paradigms. We present more details of SDN and how to implement the DAIM model to OpenFlow in Sect. 14.4, Sect. 14.5, and Sect. 14.6. Finally, conclusion and future works are shown in Sect. 14.7.

14.2 The Needs of Distributed Systems

Traditional static networks are struggling to meet the rapidly growing requirements of today's enterprises, carriers, and end-users. In addition, as the network scale in size and complexity, a distributed networking system is needed to ensure good quality of network services and performance. A distributed network system can refer to an application that executes a set of protocols to correspond the actions of multiple processes on a network [13]. Moreover, all components cooperate together to operate a single or small set of related tasks. The devices that are in a distributed system can be physically link together and connected by a local network. They can also be geographically distant and connected by a wide area network [14]. A distributed system can consist of any number of possible configurations, such as personal computers, minicomputers, mainframes, workstations, and so on.

A distributed system aims to make a network environment work as a single computer in order to cope with the extremely significant demand of users in both data storage and processing power. Examples of distributed systems may include Distributed File Systems (Hadoop), P2P Network, Cloud Network, Grid Computing, Web Server and Indexing Server, and Pervasive Computing.

There are several advantages such as the ability to connect remote users with remote resources in an open and scalable way. Regarding open, we mean that each component is continually open to interaction with other components. Whereas scalable, we mean that the system can easily be altered to accommodate changes in the number of users, computing entities, and resources [13].

Therefore, a distributed system can bring many benefits given the combined capabilities of the distributed components, than combinations of stand-alone systems [15]. However, it is not easy for a distributed system to be useful; it should provide system reliability as well. This is a very difficult goal to achieve due to the complexity of the interactions between simultaneously running components.

Ref. [4] indicates the concerned of reliability in distributed computing systems. The following table14.1 summarises the characteristics of a reliable distributed system.

Table 14.1. Characteristics of a reliable distributed system

Characteristic	Description
Fault Tolerant	It can recover from network failures without performing incorrect actions.
Highly Available	It can restore operations, instructing it to resume network services even when some components have failed.
Recoverable	Failed components can reboot themselves and re-join the system, after the cause of failure has been recovered.
Consistent	The system can execute corresponding actions of multiple components often in the case of concurrency and failure. This underlies the ability of a distributed system to act as a non-distributed system.
Scalable	It can operate properly even some aspect of the system is scaled to a larger size. For example, if the number of users or servers increases, the overall load on the system should not have a significant effect.
Predictable Performance	The ability to provide desired responsiveness in a timely manner.
Secure	The system authenticates access to data and network services

Some of the key computing trends driving the need for a distributed system paradigm include [19]:

14.2.1 Change of Traffic Patterns

Regarding the enterprise data center, traffic patterns have changed dramatically. In contrast to client-server applications, the bulk of the traffic occurs between one client and one server, applications today access various databases and servers, and thus creating a flurry of "east-west" machine-to-machine traffic before returning data to the users' device in the classic "north-south" traffic pattern. Meanwhile, network traffic patterns are changed by users as they push for access to corporate content and applications connecting from any type of device anywhere and anytime. Moreover, many enterprise and carrier managers are contemplating a utility computing model, which might include a public cloud, private cloud, or a mix of both. This may result in additional traffic across the wide area network.

14.2.2 The Consumerization of IT

Users are increasingly employing mobile technology such as smart phones, tablets, and laptops to access the corporate network. This can cause pressure for IT to accommodate these personal devices in a fine-grained manner, while protecting intellectual property as well as corporate data and meeting compliance mandates.

14.2.3 The Rise of Cloud Services

The highly demand of enterprises for both public and private cloud services has result in unprecedented growth of these services. Today's enterprises want the agility to access applications, infrastructure, and resources on demand. Providing self-service provisioning in either a public or private cloud requires flexible scaling of storage, computing, and network resources, basically from a common viewpoint and with a common suite of tools.

14.2.4 Huge Data Demand More Bandwidth

Dealing with today's mega data-sets requires efficient parallel processing on thousands of servers, which all need direct connections to each other. This emerging trend of mega data-sets has led to a constant demand for additional network capacity in the data center. Administrators of large-scale data center networks face the daunting task of managing the network to previously unimaginable size due to ever increasing network complexity.

14.3 Network Structure Paradigms

This section briefly reviews different communication network paradigms and some concepts of each one, also how they have been built and how they work. There are varieties of network topologies that can be categorized into three groups: centralized (star), distributed (grid/mesh) and combination of these two called Hybrid which is group of stars paradigm connected together [2]. So, this section will compromise between these three network structures, to form reliable network, which is capable for implementing DAIM model to support autonomic network management.

14.3.1 Centralized Network Structure

The first paradigm is the centralized network (star) paradigm which has a single node as a core node, and multiple nodes connected to that core node where each node has an operating system and applications (see Figure14.1, respectively). Illustrate the centralized network. The core node is able to configure all connected nodes using an Operating System Communication Application (OSCA)[18]. The centralized paradigm has a major drawback such as increasing the number of connected nodes will affect the core node which has a limitation of process power [17]. In addition it is obviously weak, if destruction happened to that single central core; will

lead to destroy the communication between end nodes [2]. However, the main advantage of the centralized architecture is that one core node can be responsible for managing all connected nodes, and thus managing the entire network from a single point [17].

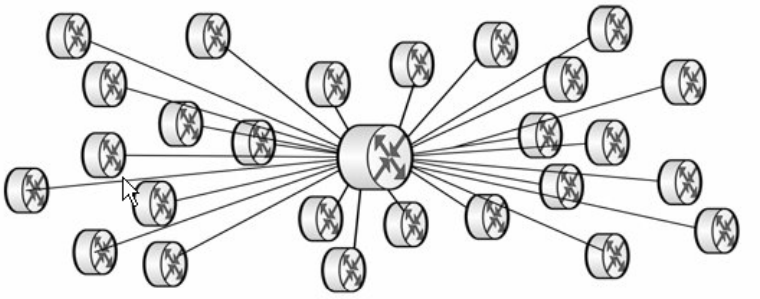

Fig. 14.1. Centralized network structure (star network)

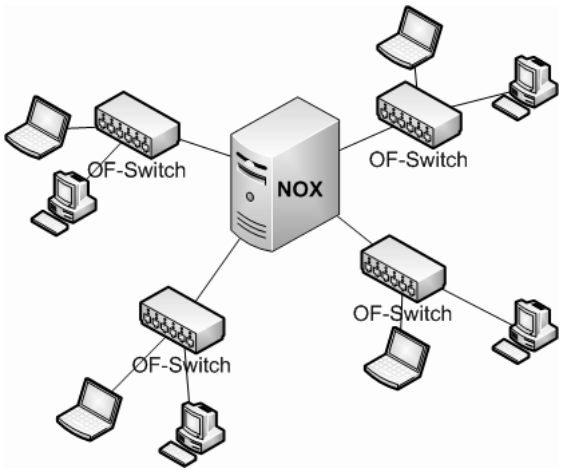

Fig. 14.2. OpenFlow network is logically centralized but can be physically distributed

The centralized structure of OpenFlow contains one NOX controller (control plane) and a number of OpenFlow switches (data forwarding plane) as shown in the Fig.14.2, NOX is network operating system, runs on a single server, which manages the forwarding decisions of multiple OpenFlow switches. It also has components developed as network requirements called network's applications to handle the network events, and able to access the network traffic, and generate traffic [12].

The combination of OpenFlow enabled switch and NOX controller, work as a router to deliver the packets from source to destination with high-level routing decisions made by the NOX controller. The data-forwarding plane is depending on the flow table, which contains flow entries to match with the first packet within a flow, happened on OpenFlow switch [23].

If the first packet of a new flow could not find a match in the flow entry, the switch will send directly to the NOX controller for high-level routing decisions. That packet passed to interested network applications to decide: whether to forward the flow on the network; collect statistics; modify the packets in the flow or view more packets within the same flow to gain more information [12].

A number of different flow-based applications have been developed in the NOX controller, they are able to control the network, reconstruct topology, track the computers whatever changing on topology, provide network access controls, and manage network history [12]. However, the centralized network structure is not feasible to implement the DAIM model because this network structure relies heavily on a single point and the approach is to distribute the self-management capabilities.

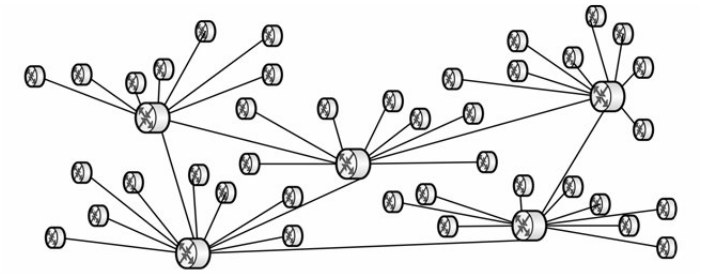

Fig. 14.3. Hybrid network structure (Combination of star and mesh)

14.3.2 Hybrid Network Structure

The second paradigm is the hybrid network paradigm which is a combination of star and grid paradigms. The hybrid network does not usually require a complete reliance on a single point [2]. Referring to figure 14.3, which shows a hierarchical structure of numbers of stars connected together as a massive star with an extra link to make a loop.

OpenFlow undertakes a logically centralized controller, which can be physically distributed as shown in Fig. 14.4. It contains three sites of OpenFlow network; each site has its own controller to serve the local requests. However, current implementation depends on a centralized single controller, which has drawbacks as mentioned

in centralized network, including lack of scalability. This hybrid network structure can distribute the event-based control plane for OpenFlow environment. In addition, it provides the scalability and at the same time keeping the power of centralized network [23].

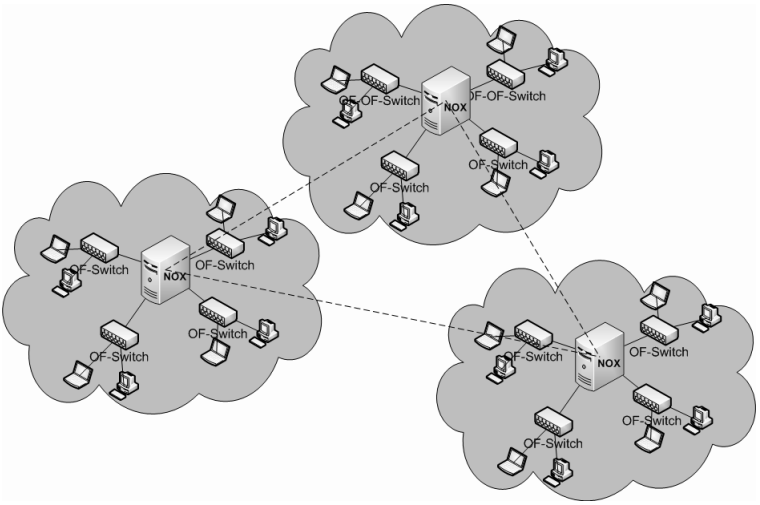

Fig. 14.4. Hybrid OpenFlow structure with number of controllers

People have introduced HyperFlow as a distributed control plane for OpenFlow, which is an application for NOX controller to communicate those distributed NOXs. In addition, each OpenFlow switch makes a local decision (relies on local flow table and local controller). HyperFlow use to passively synchronize state upon whole OpenFlow networks controllers which can enable local decision by individual controller to all packets. Thus, significantly reduces the response time of control plane for data plane requests [23].

Each controller in HyperFlow network has the ability to control the whole network because it has a coherent view of the entire network. If any controller fails, all affected switches have to reconfigure by themselves to join the other nearest controller [23]. Although this hybrid network structure has distributed control planes, however, it is not efficient to implement the DAIM model because it still relies on a central point which can control the network, even if that not usually relays on central nodes.

14.3.3 Distributed Structure Network

Lastly, it is expected that the current and next generation network should rely on distributed paradigm (mesh/grid) as presented in figure 14.5. The distributed paradigms

rely on number of distributed nodes connected together, and not rely on centralized point to communicate with all nodes. The distributed paradigm is continuous connection and it has ability to reconfigure itself within entire network [11]. For example, the flow packets can rout from node to node until they reach their destination.

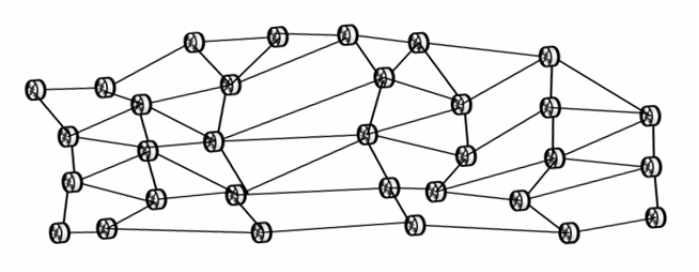

Fig. 14.5. Mesh or Grid network structure

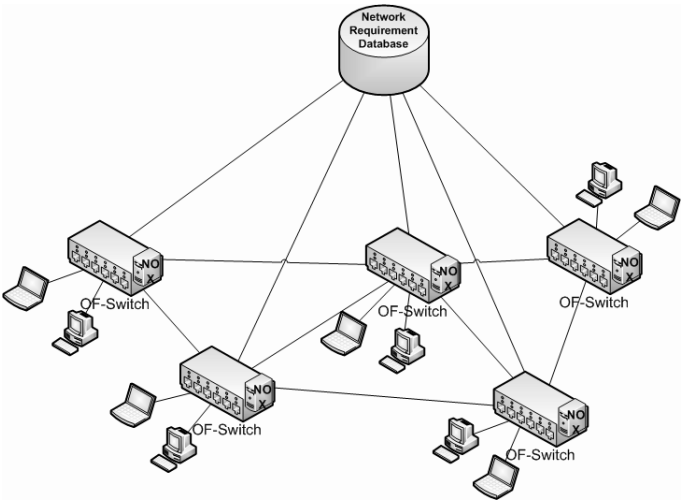

Fig. 14.6. Distributed switches which able to manage the whole network

This section is just a hypothesis to the paper approaches. Given that having a distributed structure, it is able to implement DAIM model to support autonomic network management (see Fig.14.6). This distributed structure contains some of OpenFlow switches embedded with mini-NOX inside each switch. The switches are connected together, while each switch is connected to a Network Requirement Database (NRD).

Sending packets in this structure typically uses the OpenFlow environment. The main idea is to provide distributed autonomic management using mini-NOX, which is connected to each other. Each mini-NOX is connected to the NRD to obtain all network information needed to achieve autonomic functionalities as well as obtaining information of the entire network. Moreover, each mini NOX actively synchronises with the NRD, and hosts are connected to their local switch, which is responsible to fully serve all packets within its site, unless failure happened. If the failure happened then hosts should reconfigure to the nearest switch instead of that failed switch. The new switches can actively synchronise with NRD to know all the information and the requirements to serve connected hosts, which have been adapted themselves from other site.

The mini controllers can publish events to NRD and actively synchronizes with NRD, so that other controllers can reconstruct the whole information about the network. Individual switch can serve any coming packets locally or from other switches. As a benefit of NRD, it gives the ability to any local change within individual switch as a distributed switch to deploy autonomic functions such as self-configuration. Thus, the distributed system structure is feasible to deploy the DAIM model, which has the ability to synchronize controllers and the whole network events together, to achieve the self-management as well as enabling all autonomic functions.

14.4 Software-Defined Networking (SDN)

The future of networking domain will rely more and more on software. SDN, standardised by a non-profit industry called the Open Networking Foundation (ONF), is an emerging network architecture that seeks to transform traditional static networks into flexible programmable platforms by decoupling the network control and data planes. In addition, network intelligence and state are logically centralised, and the underlying infrastructure is abstracted from the applications, which treats the network as a logical or virtual entity. With SDN, carriers and enterprises can gain unprecedented automation, programmability, and network control that will enable them to create highly scalable and flexible networks in order to meet the changing business needs[19].

Fig. 14.7 shows a logical view of the SDN structure, where network intelligence is (logically) centralised in software-based SDN controllers, which maintain a global view of the network. Therefore, the network appears to the policy engines and applications as a single logical switch. SDN provides vendor independent control over the entire network from a single logical point. This can greatly simplify the network design and operation for enterprises and carriers. Moreover, SDN also simplifies the network devices themselves as they do not need to understand and process all of the standard protocols, but accepting instructions from the SDN controllers instead.

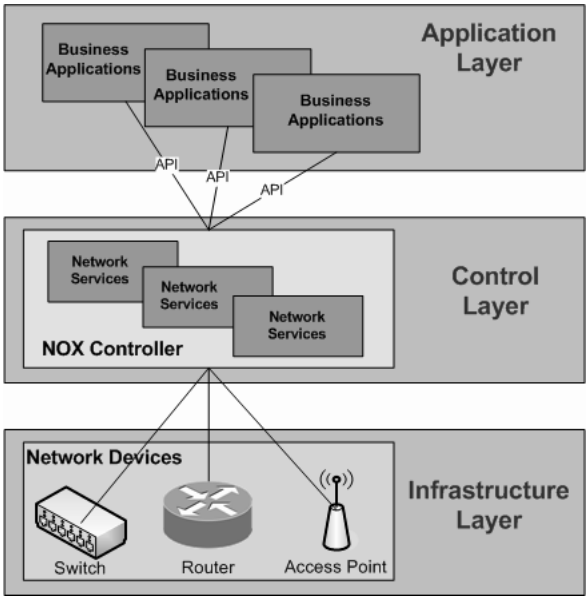

Fig. 14.7. Software-Defined Network Architecture

Most importantly, network administrators can programmatically configure this simplified network abstraction instead of having to hand-code thousands of lines of configuration among thousands of distributed nodes. This migration of control gives network operators the flexibility to manage, configure, secure, and optimise network resources via the SDN control software dynamically. As a result, they can efficiently alter the network behaviour in real-time as well as deploying new network services and applications.

OpenFlow-Based SDN architecture can present several substantial benefits including [19]:

- Centralised management and control of network devices from various vendors;
- Improve the management and automation by using common APIs to abstract the underlying network details from the provisioning systems and applications;
- Ability to provide new network capabilities and services without having to configure individual devices or wait for vendor's release;
- Programmability by administrators, enterprises, users, and software vendors using common programming environment;
- Increase network reliability and security as a result of centralised and automated management of networking devices, low configuration errors, and uniform policy statement;
- Advance network control as applications exploit central network state information to seamlessly adapt network behaviour to user needs

SDN requires some method in order for network control to communicate with the switch data path. One such mechanism is "OpenFlow" which is a standard interface for controlling computer networking switches.

14.5 Distributed Active Information Model (DAIM)

Autonomic communication relies heavily on a functional information model that provides source data to drive both decision-making process and information mining processes. The DAIM model consists of two major parts (1) O:MIB, (2) hybrid O:XML. In this section, a new information model is required to cope with the dynamics in distributed ACNs. In addition, an active object-oriented management information base (O:MIB) is proposed as a theoretical framework for the rest of the research with the hope to replace the current MIB. The corresponding programming language hybrid O:XML is explored as a practical technology to implement O:MIB, with platform-independent Java agents (e.g. Jade and JadeX). However, the details of O:XML will not be mentioned in this paper.

O:MIB Theory

DAIM model can be applied with distributed communication networks to enable autonomic functions. One of the most significant barriers when dealing with large-scale and complex distributed systems is insufficient centralised service management. Because the development of agent-based in the field of Distributed Artificial Intelligent (DAI) has grown rapidly, autonomous decentralised systems (ADSs) and multi-agent technology are by far the best solution for complex network environments. The DAIM model consists of adaptation algorithms for adapting the intelligent agents and information objects to be applied to large-scale distributed electronic systems. The main purpose of this model is to re-engineer the structure of network information model, so that the new structure can effectively cope with the next generation communication networks. It also aims to redesign the traditional MIB structure by adopting the object-oriented principles, which is required to fulfil management services such as configuration management, topology discovery, activating application process, and assigning resource process. Fig. 14.8 shows a characteristic comparison between the standard MIBs and proposed O:MIBs.

Furthermore, O:MIB can play as a part of the distributed information model to enable autonomic software agents that act as the network elements (other routers, switches, hosts, etc). These autonomic agents (AAs) inherit the surrounding agent's behaviour and also make local decisions based on the state of the network. The agents of distributed O:MIB technology will allow the richness of self-organized management. For example, dynamic software configurations, service activation, and service discovery. Moreover, DAIM model is developed specifically with embedded smart algorithms for distributed elements to improve the efficiency of local execution abilities.

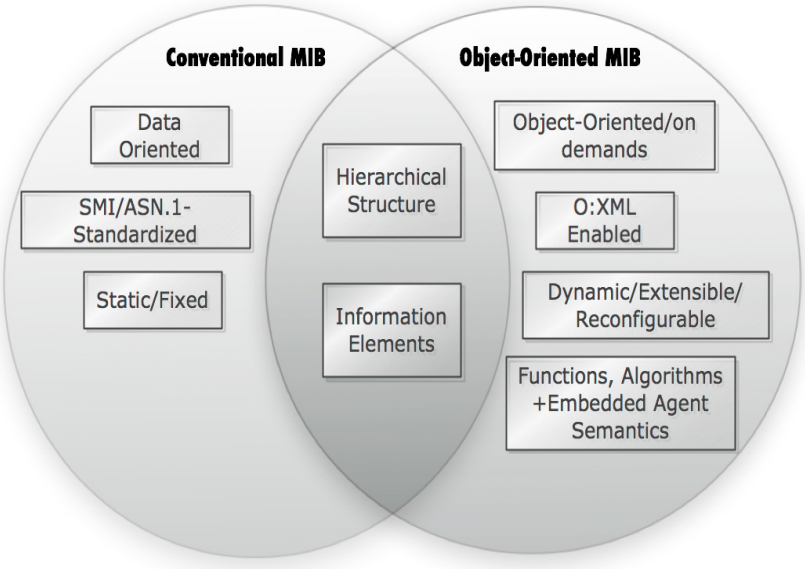

Fig. 14.8. Comparison between traditional SNMP MIB and O:MIB

These large amount of heterogeneous O:MIBs need to be well organised in a way to favour the distributed environment. This implies a distributed intelligent holonic system in order to efficiently manage and implement the O:MIBs in the hierarchical telecommunication system. The main characteristics of the holonic O:MIBs are that it's object-oriented MIBs with methods embedded, and exists on a holonic-level. For example, they are embedded into individual electronic devices such as mobile phones, printers, and even further in the sub-level of devices (chip-level). Braun and Chiang (2008) state that the holonic system is not only a component-based communication architecture, but also a universal way to construct distributed MEs in various levels. The proposed holonic agent-based O:MIB consists of three parts: (1) conventional MIB; (2) user-accessible provisioning; (3) methods/operation. Intelligent algorithms and functions are embedded in each holonic subsystem to fulfil any network tasks for agents to cooperate together and share synchronised information.

The necessary contents of an O:MIB class should cover four divisions of information: QoS parameters; device information; service information; application information and dependency information related to devices and services. Chiang et. al. (2007) designed this O:MIB class for each category of network components. When an object is required for activation during run time, any instance of the O:MIB_Class is directly created by java codes via the keyword "new":

OMIB_ClassOMIB_object = new OMIB_Class( );

The O:MIB model is expected to be used in peer-to-peer networks, mobile technology, and wireless ad-hoc sensor networks (WASNs) as well as to address other complex issues. O:MIB adopts the object-oriented principles to manage the MIB objects. It has multiple distributed agents that remain in every network component and node, which functions with its own O:MIB as a way to activate applications when required. These network components can also analyse the important data, learning the systems environment, calculate situations, and perform adapting capability. Therefore, a full understanding of autonomic communication will be obtained. Object or element is the basic information unit of the O:MIB. Each important element comprises [8]:

• Attributes: It specifies the information values that represent the characteristics of the managed object identifiers (OIDs).

• Method behaviours: An action helping to achieve autonomic communications. This can includes the self-awareness function in real time and intensive and spatial data.

• Algorithms: These are the algorithms that will support a specific network task to be embedded into O:MIB domains. It also represents a set of predefines uses of the total available method calls. For example, humidity of the network environment, temperature monitoring, and predicting the level of raising alarms and risks in autonomic communication networks.

• Messages: In a response of the on-demand requests, local messaging daemon action can invoke messages in order to obtain general information like network topology or mapping discovery.

Fig. 14.9 indicates the implementation process of O:MIB via O:XML. The software agents remain on each node having O:XML employed to populate the recorded data to the corresponding agents. In addition, JAVA agents are also involved because it is platform independent, and due to other agent development tools are mainly based on JAVA technology as well. Each agent is defined from instantiated agents according to their electronic environment. The O:MIB algorithms are invoked by the instantiated agents, whereas information values are re-configured by java-based agents. Ultimately, the agent's life-cycle are accomplished while the program is operating.

The overall stages of this approach can be described as the follows [5]:

1. Observing the current agents on distributed nodes and noticing the environments.

2. Generating new agents when a new environment is identified through the adaptation and learning strategies.

3. Functioning the local O:MIB by invoking the algorithms and methods instructed into the local O:MIB systems by default.

4. Adapting the node in regard to the awareness of the system-level objectives.

5. Wrap up the agent's lifecycle until the next round of process is ready.

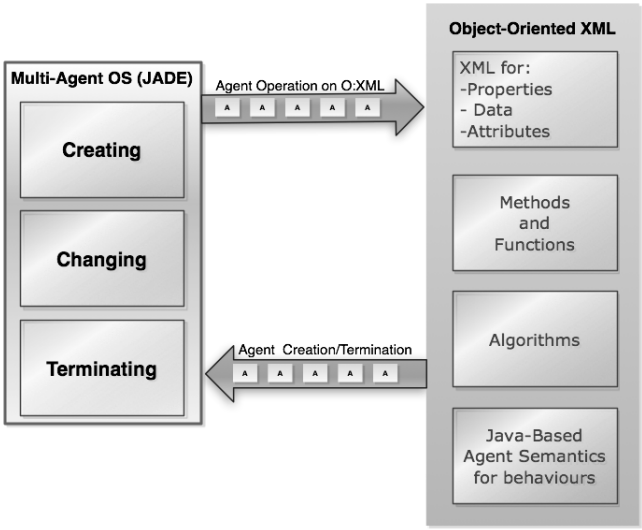

Fig. 14.9. Self-maintained process

This efficient O:MIB-based DAIM model approach is introduce to cope with managing autonomic communications in terms of self-configuring, self-adapting, self-optimizing, self-learning, self-awareness, and so on. This new information model scheme can also be applied into other Self-x properties in ACNs. The attributes of each information object in O:MIB-based DAIM model can be implemented in one O:XML file. This brings the possibility of embedding DAIM agents into portable communication devices as well as applying into real networks in the future such as wireless networks, including WASNs, MANET, Peer-to-Peer networks, and Mesh networks.

14.6 Implementing DAIM Model in OpenFlow: A Case Study

In traditional network configurations, if the circumstances should change or the requirements should change, then the network requires re-configuration again. For example, we are applying the classical compiler to executable paradigm as the following method. Firstly, we should get a software requirement. Then create the codes to meet the requirement. Finally, compile these codes into an executable program. In the case of OpenFlow-Based SDN, the process would be as the following: (1) get the system requirements from the business needs; then (2) create system configuration/code to meet business needs; finally (3) compile this into a SDN enabled configuration.

Thus, the challenge with network is that if the circumstances or requirements change, the network requires repeating all the above methodology. For example, new capacity change or the performance requirement change within the network, traffic load and traffic requirement change, therefore, this would be the problem. In order to overcome this issue, SDN decoupled the control plane and the data plan by developing components on top of the network operating system as network applications. However, these applications provide very low-level methods for interfacing with the network as the operators configure them. For example, application for discover the links in the networks by sending LLDP packets out of every switch interface, application for mapping network topology, and application for routing.

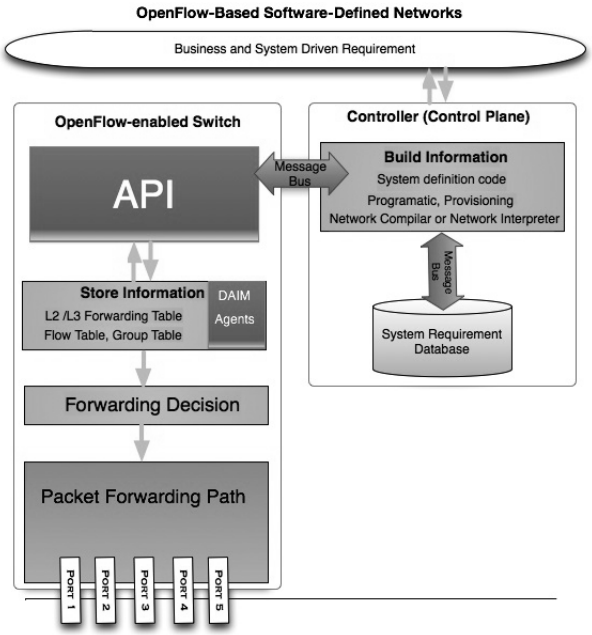

Fig. 14.10. OpenFlow-based SDN embedded with intelligent DAIM agents.

Hence, the alternatives could be implementing DAIM as a reactive interpreter network to enable autonomic behaviours – the paradigm is very similar to the SDN case. However, this new application model has some significant differences, which are based on the intelligent DAIM agents to collecting information, and the system requirement database driven from the business applications as shown in Fig. 14.10. The similarities of the proposed model and OpenFlow is that compiling the system configuration into some kind of intermediary binary code, which then hands over to a run-time environment (network operating system), which would be similar to how the NOX operates as an OpenFlow controller.

We are proposing that by creating a DAIM model on the networks could give effect to the reactive interpreter networks, so it would be a truly computational intelligent environment. Where the DAIM agents reside in the network elements, which would be OpenFlow switches in the case of OpenFlow. The actual variables in the OpenFlow table's entries, embedded within the OpenFlow switches, would be the properties of DAIM agents. These agents would then can modify or adapt their variables' values so as to implement the requirements of the network driven by the business needs. Therefore, the DAIM model would extend across all these network elements and could be thought of as reactive distributed interpreter, which is interpreting the system requirements to enable the infrastructure to provide for the business needs.

14.7 Conclusion and Future Works

Distributed systems can bring many significant benefits to the next generation networks. This paper has studied the needs of distributed systems. Three network structure paradigms: centralized, hybrid, and distributed, have been described. Then, the SDN architecture for next generation networks has been presented. Moreover, the paper has proposed an implementation of the DAIM model in next generation networks. It also identified a number of characteristics that can help to determine whether a given network structure paradigm should be realized in a centralized paradigm, a distributed paradigm, or hybridisation of the two paradigms to support the autonomic network management in the next generation networks. Future research works may include the following: continue to investigate on network structure approaches through a series of experiments and select the suitable choice of network structure paradigm, these experiments should extend the list of system requirements that has the efficiency to support autonomic network management; implement the DAIM model which can enable the autonomic management functionalities; and develop an intelligent hybrid Common Radio Resource Management approach to support the next generation wireless networks.

Acknowledgements. This work is sponsored by the Centre for Real-Time Information Networks (CRIN) in the Faculty of Engineering & Information Technology at the University of Technology, Sydney (UTS). This paper is an extended version of the ACASE'12 conference paper [21].

References

1. AL Sabbagh, A., Braun, R., Abolhasan, M.: A Comprehensive Survey on RAT Selection Algorithms for Heterogeneous Networks. Journal of World Academy of Science, Engineering and Technology (WASET) 73, 141–145 (2011)

2. Baran, P.: On Distributed Communications Networks. IEEE Transactions of Communications Systems 12(1), 1–9 (1964)
3. Bays, L.R., Marcon, D.S.: Flow Based Load Balancing: Optimizing Web Servers Resource Utilization. Journal of Applied Computing Research 1(2), 76–83 (2011)
4. Birman, K.P.: Reliable Distributed Systems: Technologies, Web Services, and Applications. Springer-Verlag, New York Inc. (2005)
5. Braun, R., Chiang, F.: A distributed Active Information Model Enabling distributed Autonomics in Complex Electronic Environments. In: Third International Conference Broadband Communications, Information Technology & Biomedical Application, pp. 473–479. IEEE (November 2008)
6. Chiang, F., Braun, R.: Self-adaptability and vulnerability assessment of secure autonomic communication networks. In: Proc. of Managing Next Generation Networks and Services Conf., pp. 112–122 (2007)
7. Chiang, F., Braun, R.: Towards a management paradigm with a constrained benchmark for autonomic communications. In: Proceedings of the Conference Computational Intelligence and Security, pp. 250–258 (2007)
8. Chiang, F., Mahadevan, V.: Towards the distributed autonomy in complex environments. In: International Conference on Information and Multimedia Technology, pp. 169–172 (2009)
9. Chiang, F., Fernandez, H., Braun, R., Agbinya, J.: Integrating Object-Oriented O: XML Semantics into Autonomic Decentralised Functionalities. In: 7th International Symposium on Communications and Information Technologies, pp. 768–773 (2007)
10. DenHartog, M.: How to Read and Understand the SNMP MIB. DPS Telecom (2008), http://www.dpstele.com/white-papers/snmp-mib/offer.php (viewed September 25, 2011)
11. Escribano, J.F.: Grid and Mesh Technologies, IT Entrepreneurship (2008), http://www.econectados.com/wp-content/uploads/grid_mesh.pdf (viewed April 30, 2012)
12. Github: NOX Introduction. noxrepo Tech. Rep. (2008), https://github.com/noxrepo/nox-classic/wiki/NOX-Introduction (viewed October 20, 2011)
13. Google Code University: Introduction to Distributed System Design, http://code.google.com/edu/parallel/dsd-tutorial.html
14. IBM: TXSeries for Multiplatforms: Concepts and Planning, 5th edn. International Business Machines Corporation (November 2005)
15. Manjula, K., Karthikeyan, P.: Distributed Computing Approaches for Scalability and high performance. In: Conference Proceedings of Distributed Computing, vol. 2, pp. 2328–2336 (2010)
16. Meyer, K., Erlinger, M., Betser, J., Sunshine, C., Goldszmidt, G., Yemini, Y.: Decentralizing Control and Intelligence in network Management. In: Proceedings of the 4th International Symposium on Integrated Network Management, Santa Barbara, CA (May 1995)
17. Morkved, T.: Peer-to-Peer Programming with Wireless Devices. Information and Communication Technology thesis, University of New South Wales, Sydney Australia & Agder University College, Grimstad Norway (2005)
18. NathRK, P.: Internet Technology with Client Server Architecture (2010), http://www.data-e-education.com/E079_Centralized_Network_Architecture.html (viewed April 17, 2012)
19. Open Networking Foundation: Software-Defined Networking: The New Norm for Networks. ONF White Paper (April 2012)

20. Perez-Romero, J., Sallent, O., Agusti, R., Diaz-Guerra, M.A.: Radio Resource Management Strategies in UMTS. John Wiley & Sons Ltd., Chichester (2005)
21. Pupatwibul, P., AL Sabbagh, A., Banjar, A., Braun, R.: Distributed Systems in Next Generation Networks. In: Conference Proceedings of 1st Australian Conference on the Applications of Systems Engineering, ACASE 2012, p. 32 (February 2012)
22. Rodriguez, H.F.: Active MIB, An Object Oriented Solution For Network Management. Master thesis, Chalmers University of Technology, Sweden (May 2007)
23. Tootoonchian, A., Ganjali, Y.: HyperFlow: A Distributed Control Plane for OpenFlow. In: Proc. of INM/WREN, San Jose, CA (April 2010)