

A Knowledge Plane for Wireless Mesh Networks^{*}

Roberto Riggio, Francesco De Pellegrini, Daniele Miorandi, and Imrich Chlamtac

CREATE-NET
Via dei Solteri 38, 38100, Trento, Italy,
`name.surname@create-net.org`

Abstract. The quest for the autonomic management of communication systems dates back to more than two decades ago. In practice, it became a compelling need when such systems started growing in size and complexity. The risk, in fact, was that the support/management of the infrastructure became so expensive to be the major design constraint. In telecommunication networks, in particular, autonomicity is perceived practically as an alias of Autonomic Network Management (ANM) [1]: proper handling of network complexity, in fact, is perceived as a cornerstone for achieving robustness and performance. But, despite the expectations, while ANM is emerging as one of the hottest research topics, little concrete results have been achieved so far. This can be ascribed, on one hand, to the complexity of the issue itself, whose theoretical foundations have not been completed so far, and on the other one to the lack of a research platform on which novel solutions can be tested in a controlled and replicable fashion. In such a scenario, due to their reconfigurability and ease of deployment, wireless mesh networks provide us the opportunity to design from scratch an autonomic control plane on top of a network of practical interest. In this paper we propose a novel *Knowledge Plane*, tailored specifically for the WMNs scenario and capable of enabling consistent sharing of services ontology among the entities participating the WMNs. As a case study, we present JANUS, a novel and freely available monitoring framework exploiting the proposed Knowledge Plane.

1 Introduction

The quest for the autonomic management of information and communication systems dates back to more than two decades ago. It became a compelling need when such systems started growing more and more complex. The risk, in fact, was that the support/management of the infrastructure would become too expensive, up to the point to undermine the development of new technologies. The

^{*} This work was partially supported by MIUR within the framework of the WING project (grant number RBIN04M292) and by the European Commission through the project BIONETS (grant number IST-27748). Part of this work has been presented at TridentCom 2007. On line resources available at <http://www.wing-project.com/>.

idea of automatic management, or self-management, was officially embraced in the IBM initiative on Autonomic Computing [2, 3]. In the networking field, in particular, autonomicity is often identified with the principle of Autonomic Network Management (ANM) [1]. The need for efficient ANM indeed reflects the fact that the complexity of the network has a serious impact on the design of robust and performing architectures. But, despite the expectations, while ANM is emerging as one of the hottest research topics, little concrete results have been achieved so far. This can be ascribed, on one hand, to the complexity of the issue itself, whose theoretical foundations have not been completed so far, and on the other one to the lack of a research platform on which novel solutions can be tested in a controlled and replicable fashion. In such a scenario, due to their reconfigurability and ease of deployment, wireless mesh networks provide us the opportunity to design from scratch an autonomic control plane on top of a network of practical interest.

Wireless Mesh Networks (WMNs) [4, 5] rely on a multi-hop wireless backhaul for delivering connectivity to the end-users. WMNs provide a technological bridge between the ad hoc networking paradigm and traditional wireless networks in terms of both common syntax and semantics such IEEE 802.11-based WLANs. However, unlike ad hoc networks, where node mobility was the major concern, research on WMNs moved the focus on network scalability. Thus, on one hand, several companies are already providing proprietary solutions based on the IEEE 802.11 family of standards [6], and on the other hand several significant efforts are coming from the academic world in order to develop testbed and prototypes based on off-the-shelf technologies and open source software [7–11]. A parallel standardization initiative is carried out by the IEEE 802.11s Task Group that plans to integrate mesh networking functionality within 802.11 MAC Layer.

The contribution of this paper is twofold. First we propose a novel *Knowledge Plane* [12] tailored for the WMNs scenario and capable of enabling consistent sharing of services ontology among the entities participating the WMNs. Second, we present JANUS, a novel and freely available¹ monitoring framework exploiting the proposed *Knowledge Plane*. The following use case scenarios may be envisioned for the current prototype:

- *Experiments reproducibility*. Experiments carried out over current WMN testbed are characterized by a scarce reproducibility due to the extreme time-varying nature of the network. In such a scenario, the prototype’s monitoring capabilities are suitable to be exploited in laboratory/field network testbeds in order to enhance experiments reproducibility. The information contained in the knowledge plane can in fact be exploited as check point of the network conditions (in terms of link states, load, environmental noise, etc) for each experiment providing providing precious information for both the analysis of the collected data and the design of further experiments.
- *Manned/Unmanned network profiling*. Network wide performance tuning may be a challenging task in WMNs where several factor may influence the

¹ The code is released under the BSD License [13].

global behaviour. Nowadays a wide range of techniques is available to support network and network devices management. Common examples include SNMP [14], ICMP [15], and netconf [16]. However, such solutions are developed using a strongly centralized approach, while the distributed and self-organizing nature of WMNs suggest a transition from network management (in terms of manual tweaking) to network sensing (in terms of distributed and automated fault detection and/or performance analysis). In such a context, the proposed Knowledge Plane is suitable to be exploited as rollback point for a working or well performing network configuration in manual network tuning as well as objective method to evaluate the effects of network configuration changes in an Autonomic Networking scenario.

The Knowledge Plane is here intended as a network construct additional to both the Data and the Control Plane. We remark that the Knowledge plane is not meant to support the domain-specific knowledge, as it is nowadays embedded into the network design itself. In the case of the TCP congestion control loop, for example, roundtrip times knowledge regulate the congestion window size. At such level of granularity, accurate parameters can be safely estimated at the appropriate layer (transport layer in this case). In what follows, conversely, we address a separate and distributed cognitive construct aimed at equipping the network with an high level view the tasks.

The knowledge plane is then required to encompass consistent syntax and semantic in order to allow its exploitation for monitoring purposes as well as to adapt the behavior of the node depending on the particular operating conditions (e.g., traffic type, channel perturbations, network status, node selfishness and/or maliciousness, among the others). In our approach, such a feature is obtained by exploiting both a meta-model, general enough to support the different aspects of existing mesh networking paradigms and a meta-data exchange facility. These elements have been identified respectively in the Meta Object Facility (MOF) [17] and the XML Metadata Interchange (XMI) [18]. As it will be clear in the following, the exploitation of these two standards allows uniformity among involved models to be easily gained. The philosophy of our approach is aligned with [3], where a consistent definition of the available resources and controls is considered mandatory to effective autonomic computing.

The remainder of this paper is organized as follows. Section 2 presents and discusses some related works. The autonomic networking paradigm is illustrated in Sec. 3. Section 4, summarize the state-of-the-art in WMNs. Section 5 presents a general overview of our approach. In Sec. 6 we describe our prototype. Finally, Sec. 7 concludes the paper pointing out directions for future work.

2 Related Work

A large number of protocols exists to support network and network devices management. Common protocols include SNMP [14], ICMP [15], and netconf [16]. Also, the MOME project [19] is maintaining a database with more than 400

measurement tools. However, most of such tools are designed on centralized architectures meaning that each node participating to the network runs a process which gathers information about the current network state. When a problem is recognized, the running process sends alerts to some management entities. Upon receiving these alerts, the management entities are programmed to react by taking some actions (e.g., operator notification, event logging, system reboot/shutdown, etc.). Management entities can also poll end-stations to check the values of certain variables.

The Simple Network Management Protocol (SNMP) is an application layer protocol developed in order to standardize the exchange of management information between network devices. Each device implementing SNMP owns a Management Information Base (MIB). A MIB is a collection of information that is organized hierarchically. MIBs are accessed using SNMP.

A Distributed Architecture for Monitoring Mobile Networks (DAMON) is introduced in [20]. DAMON relies on agents within the network to actively monitor network behavior and to send this information to data repositories. DAMON's generic architecture supports the monitoring of any protocol, device, or network parameter. VISUM [21] is a distributed framework for monitoring wireless networks. Data, collected by agents distributed over several hosts, are gathered at a centralized repository and can be exploited by a visualization tool.

In MobileMAN [22], the information collected at different layers of the networking stack are shared in a common local memory structure and exploited to adapt the behavior to the current system status. Such an approach satisfies the layer separation principle, i.e., protocols belonging to different layers can be added/removed from the protocol stack without modifying the protocols operating at the other layers. Moreover, it is full compatible with existing standards, since it does not change the core functionality of each layer maintaining all the advantages of a modular architecture.

In [3], a conceptual architecture for autonomic computing is sketched. The authors analyze the motivation behind the quest for autonomic computing and focus on the control loops introduced by the self-management routines. Several control *disciplines* are identified (e.g. self-configuring, self-healing, etc) together with the need for an high level orchestrations among them in order to control the mutual interaction of control loops, that could otherwise lead to unpredicted and possibly disruptive effects. In the architecture envisioned in [3], control loops leverage a common knowledge of the system. However, in order to enable effective data sharing across heterogeneous systems, a common syntax and semantic is required in order to communicate the system status or to act on the system.

Along this line, in [12] the authors summarize this need in a new concept: the *Knowledge Plane*. In the envisioned scenario, the *Knowledge Plane* is supposed to collect information about the network status as well as about services constraints and policies. A comparison with the Internet is due in this case. The current Internet is the offspring of a simple idea: building a transparent core network and move all the complexity to the edges. This approach led to a situation where the network core is not aware of its expected behavior and the end-user

applications cannot tell what's happening in the network that appears to them as a black box. According to the authors, the use of a *Knowledge Plane* has two main targets. On one hand it will make a network able to describe itself, and as result capable of supporting self-configuring and self-healing operations. On the other hand it should give the applications the capability to *reason* about the operating environment. In fact, current Internet design is based on a transparent core network routing packets between intelligent edges. As a result the core network only deals with packets without knowing their purpose, while the edges understand the data being carried with the core network appearing to them as a "black box". The deployment of a network wide Knowledge Plane would allow the core network to understand the applications running across it and the behavior that they expect from the network. The paper envision modeling of the entities that we want to understand as the approach to achieve such reasoning features.

In this paper we propose a novel *Knowledge Plane* which addresses the interoperability issues raised in [3]. In our study the focus is on the wireless mesh networking paradigm: the interest there is due to the fact that, being meant as unplanned systems, WMNs are required to cope with dynamically changing environment and operations conditions, thus naturally demanding for self-configuring and self-healing capabilities. Moreover, unlike other network technologies, where pragmatic considerations make the management plane hard to change, being an emerging paradigm, WMNs allows us to craft from scratch a knowledge plane. The proposed approach leverages a common meta-modeling language and meta-data exchange in order to provide a consistent definition of the knowledge across heterogeneous WMN architectures. Our approach is a step toward the definition of a network-wide knowledge base that can be exploited by nodes in order to adapt their behavior (e.g. for identifying malicious and/or not cooperative nodes). To the best of the authors' knowledge there are no other works that address such issue in the WMN scenario.

3 Autonomic Network Management

The driving factor for ANM [1] is the consideration that nowadays networks cannot be considered static objects. On the contrary, real world networks, grow, change, evolve and (sometimes) disappear. Legacy network management tackles the issue of network adaptation, intended as the ability of management policies to cope with minor changes in the system configurations. As networks become more and more dynamic there is a growing need to resort to human operators tuning in order to adapt them to the new, changing environment. Such labor-intensive tasks have a high cost and are intrinsically error-prone. The latter feature is extremely harmful as communication networks get used to control critical infrastructures (e.g., electrical power grid). In the networking scenario we can envision an autonomic manager dedicated to self-management functionalities. Such a manager must monitor the status of the resources it manages,

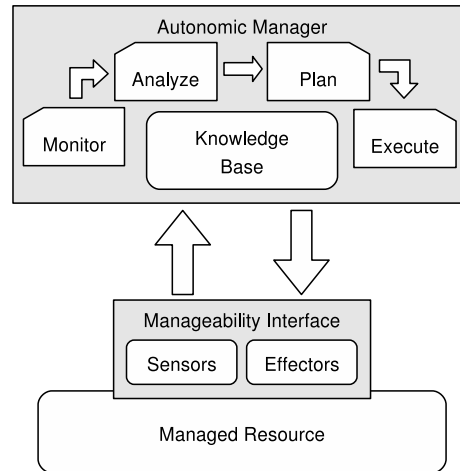


Fig. 1. Autonomic manager architecture.

analyze the current conditions against some policies and take some actions if such policies are not verified.

An autonomic computing system system is defined by its capacity to monitor the operating environment, model its behavior and take actions according to some policies. An autonomic computing architecture consists of one or more control loops capable of dynamically managing various aspects of the computing architecture. Figure 1 shows the conceptual structure of an autonomic manager. In this figure the control loop is decomposed into four distinct functions sharing a common knowledge of the operation environment. Such functions are generally referred with the acronym MAPE [3]:

- *Monitor*: it probes the managed resources for operating parameters (e.g. links status). It also performs aggregation, filtering and reporting operations over such data;
- *Analyze*: it provides the autonomic manager with a mechanism to model the operating environment. Such a model can then be exploited in match data with the system dynamics and to forecast future trends;
- *Plan*: it provides the reasoning framework that is supposed to define the actions in order to achieve a specific objectives (defined as policies in the autonomic manager knowledge base);
- *Execute*: it is in charge for the implementation of the actions output of the plan function.

The *Knowledge Base* represents the data used by the autonomic manager in MAPE process. The *Manageability Interface* presents to the *Autonomic Manager* a transparent interface to the various management technologies that exist in the market today, e.g. SNMP, Java Management Extension (JMX), Control And Provisioning of Wireless Access Points (CAPWAP), etc. In [3] Web Services are

envisioned as suitable technology for the implementation of the *Manageability Interface*. The *Knowledge Base* can be built using the *Manageability Interface* by querying the *Managed Resource* and/or generated by the *Autonomic Manager* during its operation. In such a scenario exploiting a common semantic is mandatory in order to enable effective data sharing across the MAPE process.

4 Wireless Mesh Networks

As said before, the wireless mesh network paradigm provide us with a unique opportunity to develop a novel monitoring and control plane able to adapt the original network setup to a wide range of circumstances, with huge savings in manual (i.e. human) tuning and verification. As a matter of fact, the need of a preliminary step in defining a knowledge plane for WMNs is rather clear to all those who currently deal with WMNs and experienced the huge problems of portability and reproducibility of experiments on current installations. In this section we will briefly introduce the wireless mesh networking paradigm. We refer to [4, 5] for a comprehensive introduction to wireless mesh networking.

WMNs consist of several nodes exploiting the wireless medium and using possibly multiple radio technologies/interfaces in order to gain connectivity. Communications take place by means of multi-hopping, in that packets are relayed by the nodes in the network until the final destination is reached and the routing algorithm is responsible for the selection of the path along which data packets are sent. A widely used technique to achieve this goal requires to build a graph where each node represents a network element (e.g. a router) and each edge represents a communication line (often called link) between a pair of networking elements. Given such premise, a route between two pair of routers is the shortest path between them on the graph. One way of measuring path length is the number of hops between a pair of nodes. However, exploiting such a metric in WMNs leads to poor performances since it is biased to use links between distant nodes without taking into account link quality. A detailed evaluation of the performance of different routing metrics is reported in [23]. The WMN paradigm as introduces the concept of *node specialization*, in which a distinction exists in terms of logical roles supported by the physical devices. Each node in a WMN “plays” at least one of the following roles:

- *Relay*: it builds the multi-hop wireless backhaul by establishing links with neighboring nodes;
- *Gateway*: it provides an interface between WMN and another network, typically the Internet;
- *Access point*: it delivers wireless connectivity to the clients;
- *Client*: the end-user devices.

Nodes providing relaying/access functionality are generally computationally powerful devices with no constraints on power consumption and possibly equipped with multiple radio interfaces/technologies. Those nodes are generally called “mesh routers” as opposed to the end-user devices generally referred to

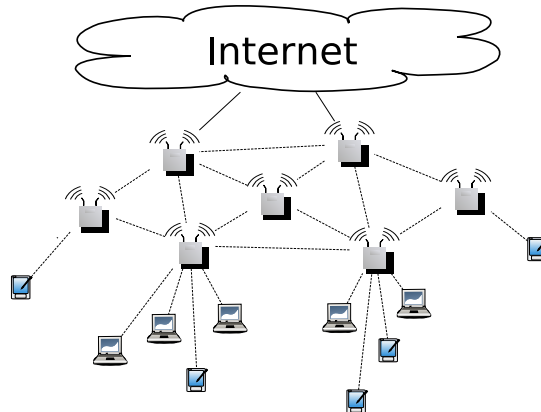


Fig. 2. Two-tier architecture with mesh routers providing backhaul connectivity to the Internet, while the clients act just as sources/destinations of Internet connections.

as “mesh clients”. Mesh routers can also act as gateways providing the WMN with Internet connectivity.

Multi-tier architectures can be envisaged [5]. Figure 2 shows a two-tier architecture where mesh routers realize a mesh backbone, providing the clients with the opportunity to connect to a remote Internet gateway. Typical applications of this architecture are in community/neighborhood networking and in wireless mesh ISPs, where mesh routers are placed on the roof and a local in-home distribution service (either wired or wireless) is added to provide end-user connectivity. Examples of such architecture include the MIT’s Roofnet [7] and the (commercial) LocustWorld [9] deployments.

5 Overview of the Proposed Approach

As stated in Sec. 3, the applicability of the ANM concepts lies in the definition of a common syntax and semantic for the *Knowledge Base*. However, at present, both commercial and academic solutions for wireless mesh networking are characterized by a significant diversities in their implementations. Moreover, the *node specialization* feature described in Sec. 4 introduces an additional heterogeneity, in that nodes with different capabilities are participating the same WMN; e.g. powerful gateways providing both Internet connectivity and WLAN access to clients and stripped down nodes providing just relaying functionalities.

Hence, the key issue in the definition of the *Knowledge Base* is to make it consistent across different WMNs implementations and yet expressive enough to handle the dynamic and distributed nature of such systems. The MOF allows models to be exported from one application into another, transmitted across a network and stored into suitable repositories. Such benefits are extended to non-UML models as long as they are expresses using a MOF-based language. In such a context, defining mappings between the meta-models which correspond to

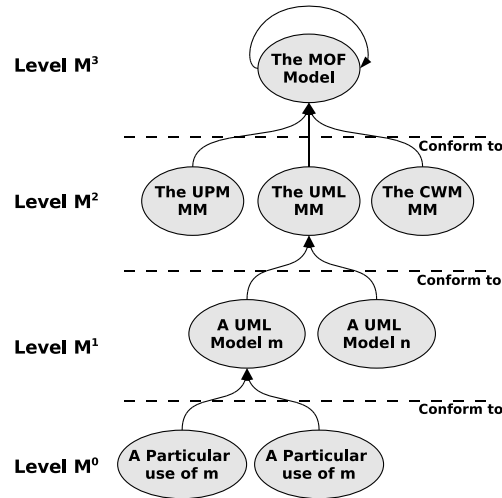


Fig. 3. The MOF four-layer architecture.

different wireless mesh networking paradigms and the MOF meta-model would automatically imply the possibility to exploit MOF as a common language for defining the *Knowledge Base* and XMI as a common language for meta-data exchange enhancing re-usability and interoperability. In this section we shall illustrate all details of our approach.

XMI defines the way a generic MOF-compliant model can be represented as an XML document. For a given meta-model, each XMI-conforming implementation will produce a DTD (or an XML Scheme), representing the meta-model, and an XML document, representing an instance of the given meta-model. The specific generation rules rely on a MOF definition of the model's meta-model; therefore, a meta-model can have its models interchanged through XMI only if it is represented as an instance of the MOF meta-meta-model. It is worth pointing out that XMI works at all abstraction levels of the meta-model architecture defined by MOF. This implies that it can be used for both the object serialization and the meta-data exchange.

Our architecture is a particular case of the MOF meta-data architecture. An example of this last architecture, tailored for the UML [24] environment, is shown in Fig. 3. Here:

- *The lowest layer*, sometimes called *original level* [25], is that originating the model and often contains run-time data. At this level XMI can be used for handling the object serialization.
- *The model layer* includes the meta-data relative to the lowest layer. Meta-data are aggregated as models. At this level XMI can be used for handling the model or the meta-data exchange among tools using the same meta-model.
- *The meta-model layer* includes the description of both the structure and the semantics of the meta-data (i.e., the meta-meta-data). The meta-meta-data

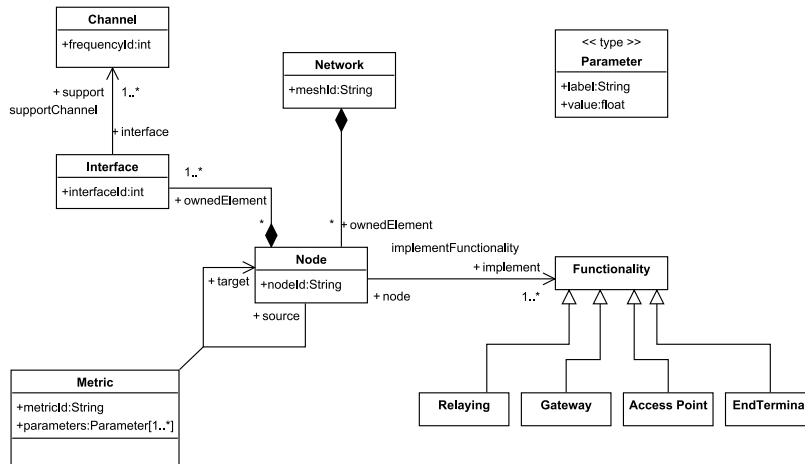


Fig. 4. Meta-model for WMNs.

are aggregated as meta-models. A meta-model is an “abstract language” for describing different kinds of data. At this level, XMI can be used for representing the model language (i.e., the meta-model).

- *The meta-metamodel layer* includes the description of both the structure and the semantics of the meta-meta-data. The use of XMI at this level allows an MOF model to be represented as an XML document.

In the standard MOF stack, the meta-metamodel (also called MOF Model) is self-defined and allows the definition of meta-models at the third layer. The UML meta-model is one of the well-known examples of meta-models; it is possible to define also other generic languages for meta-modeling.

Due to the peculiarities of the scenario, our modelling constructs are fundamentally different than the ones provided by UML. In such a context extending the UML meta-model in order to create a MOF meta-model defining the abstract syntax of such modeling constructs would imply carrying all the overhead of the UML meta-model. This issue is particularly relevant since we plan an extensive exploitation of the XMI format in order to circulate the network models among the nodes participating the WMN.

As stated in Sec. 4, a WMN is composed by several nodes that communicates with each other exploiting the wireless medium. Connection with Internet is provided by specialized nodes called *Gateways*. Such modeling constructs do not fit easily into any of the UML modeling paradigms, then a meta-model that is tightly coupled with the modeling domain and that does not carry all the overhead of the UML meta-model can provide a better support. According to the WMN definition introduced in Sec. 4 the following key concepts have been identified and integrated in the proposed meta-model (see Fig. 4):

1. *Node Specialization*. Each node must implement at least one of the following functionalities: Relay, Gateway, Access Point, and End-Terminal.
2. *Routing Metrics*. Consists of at least one parameter exploited by the routing algorithm in order to determine whether one route should perform better than another. Metrics can cover information such as bandwidth, delay, hop count, etc.
3. *Multiple Interfaces*. Each node participating the WMN communicates with the others exploiting one or multiple channels. Such a behavior can be implemented either using a single interface that dynamically adjust its communication frequency or using multiple interfaces each of them tuned to one frequency. We can also envision a scenario multiple interfaces are dynamically tuned to a certain frequency in order to optimize resources allocation.

Albeit the MOF four-layers architecture may seem too cumbersome for “bare” network monitoring in WMNs, we argue that the ANM scenario requires a “paradigm shift” analogous to that characterizing the software engineering community with the larger and larger exploitation of model engineering [26]. Such a shift was motivated by the consideration that, without interoperability between different environments, users are forced to use a small set of development tools or, alternatively, to totally renounce to modeling. In the same way, the ANM paradigm requires consistent modeling of the various networking aspects in order to enable effective orchestration of heterogeneous network devices. Moreover, it is worth pointing out that XMI provides a mechanism for specifying differences between models. This feature is particularly useful when updates must be disseminated quickly or very often and transmitting the entire model is resource consuming in terms, for example, of time and/or network load.

6 System Architecture

This Section describes JANUS, the prototype developed in order to prove the suitability of our meta-model to a practical case study, namely the wireless mesh networking paradigm. JANUS is a novel monitoring framework designed with the goal of addressing the peculiarities of WMNs. It exploits Pastry [27], a peer-to-peer overlay network, in order to make network information, collected at different layers of the stack, available to all nodes in the system. Such information can be used for monitoring purposes as well as to adapt the behavior of the node depending on the particular operating conditions (e.g., traffic type, channel perturbations, network status, node selfishness and/or maliciousness, among the others).

JANUS is implemented in the form of a Java application built on top of Scribe [28], which is a scalable group communication system allowing participants to subscribe to a topic and to publish messages. Scribe exploits Pastry in order to build an efficient multicast tree for the distribution of events to a topic. Any Scribe node is allowed to create a new topic making it available for subscription to the other nodes. A credential-based system is used for controlling

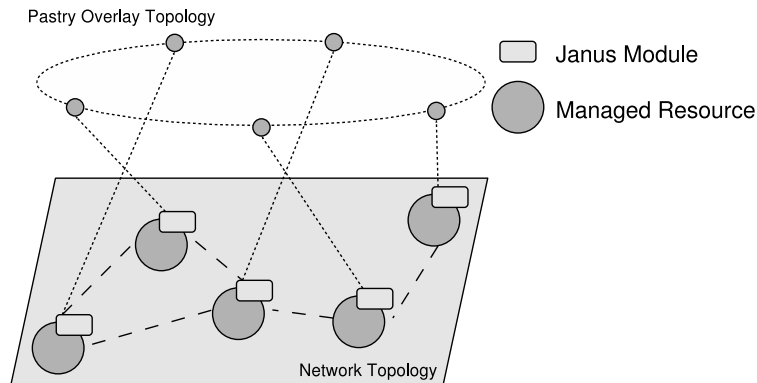


Fig. 5. Building blocks and relationships in the JANUS architecture.

the publication of message for a specific topic. Also, we used FreePastry [29] as the reference peer-to-peer platform. FreePastry is an open-source implementation of Pastry’s API from Rice University, which comprises, among the other, a Scribe implementation. The building blocks of JANUS and their relationships are sketched in Fig. 5. In this section we shall illustrate the implementation details of each block.

It is worth pointing out that, with JANUS, we aim at providing a framework where information about the entities participating the WMN can be extracted and the corresponding models can be represented and shared. Application built on top of JANUS can exploit the available knowledge in order to, for example, diagnose failures, implement self-healing policies, or monitoring the behavior of malicious nodes.

6.1 Managed Resource

The Managed Resource is a network node participating the WMN and running the JANUS software. The Microsoft Mesh Connectivity Layer [8] is exploited in order to realize mesh connectivity among the nodes. MCL routes using a modified version of DSR [30] called Link Quality Source Routing (LQSR) [31]. LQSR assigns a weight to each link. This weight is the expected amount of time it would take to successfully transmit a packet of some fixed size on that link. In addition, the channel, the bandwidth, and the loss rate are determined for every possible link. This information is sent to all the nodes. Based on this information, LQSR uses a routing metric called Weighted Cumulative Expected Transmission Time (WCETT) [23], a variant of the ETT [32] metric, to define the best path for the transmission of data from a given source to a given destination.

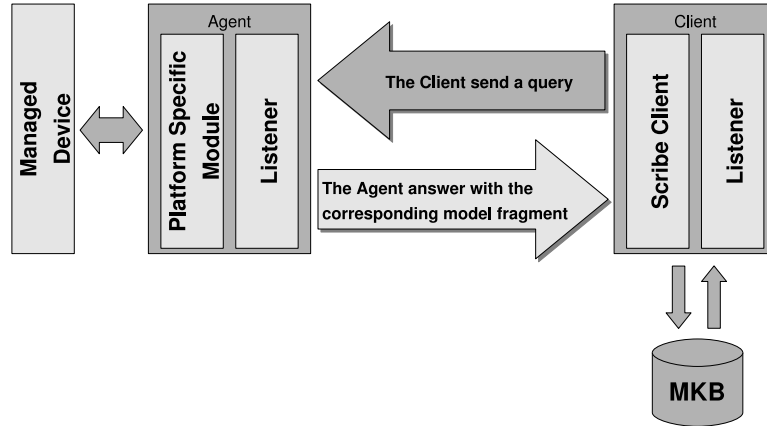


Fig. 6. Interactions between Agent and Client.

6.2 Agent

The Agent provides an implementation of the *Manageability Interface* depicted in Sec. 3. It is implemented in the form of a daemon running on each managed device. All the Agents share a list of *objects* that they can monitor. An object is a particular aspect of the managed resource, as for example an Agent can track TCP connections running across the managed device’s outgoing links. The Client can then query each link and gather the status of each TCP connection. Agents can also send traps to the Client in order to asynchronously notify some events². Figure 6 shows the interactions between agents and clients.

Information about the *Managed Resource* are collected by the **Platform Specific Module** and then converted in a format suitable for the integration into the *Mesh Knowledge Base*. Figure 7 shows an XMI document conforming with the meta-model introduced in Sec. 5 representing a node’s link cache. The link cache itself is basically a list of “known” nodes, including the link metrics between those nodes. Each link cache entry can be described by the following tuple

$$LS = \langle SA, DA, M \rangle,$$

where SA is the source node’s address, DA is the destination node’s address, and M is the link metric, being

$$M = \langle ETX, B \rangle$$

where ETX is the Expected Transmission Count (ETX) [32] and measures the expected number of transmissions, including retransmissions, needed to send a unicast packet across a link, and B is the link bandwidth.

² In the current implementation traps from the Agent to the Client are not supported.

```

<?xml version='1.0' encoding='UTF-8' ?>
<Janus>
  <Janus.Network xmi.id="palantir">
    <Janus.Network.ownedElement>
      <Janus.Node xmi.id="ec-a0-b9-dc-76-c5" />
      <Janus.Node xmi.id="1c-12-78-6c-83-c0" />
    </Janus.Network.ownedElement>
    <Janus.subSequent>
      <Janus.subSequent.from>
        <Janus.Node xmi.idref="ec-a0-b9-dc-76-c5" />
      </Janus.subSequent.from>
      <Janus.subSequent.to>
        <Janus.Node xmi.idref="1c-12-78-6c-83-c0" />
      </Janus.subSequent.to>
      <Janus.Metric>
        <Janus.Metric.parameters>
          <Janus.Parameter>
            <Janus.Parameter.label>ETX</Janus.Parameter.label>
            <Janus.Parameter.value>0.17</Janus.Parameter.value>
          </Janus.Parameter>
          <Janus.Parameter>
            <Janus.Parameter.label>Bitrate</Janus.Parameter.label>
            <Janus.Parameter.value>0.9</Janus.Parameter.value>
          </Janus.Parameter>
        </Janus.Metric.parameters>
      </Janus.Metric>
    </Janus.subSequent>
  </Janus.Network>
</Janus>

```

Fig. 7. Node's Link Cache as represented in the MKB.

6.3 Client

The Client provides a framework for the implementation of an *Autonomic Manager*. At initialization time each Client tries to connect to a bootstrap host in order to subscribe a Scribe group. If no bootstrap host is found, then a new Scribe ring is initialized by the Client. Please note that the bootstrap host is required only at initialization time. Each Client periodically queries the agent in order to gather the managed objects. Gathered objects are first used for updating the local MKB and then published on the Scribe ring. When objects are delivered each Client takes care of merging them with the local MKB (during the bootstrap phase the MKB is initialized with the locally collected objects).

Clients support plug-ins for extending their capabilities. Such plug-ins exploits the MKB in order implements an arbitrary MAPE process. Current prototype ships with a *Listener* module implementing monitor functionality and allowing an external application for querying the node's MKB. JANUS has been

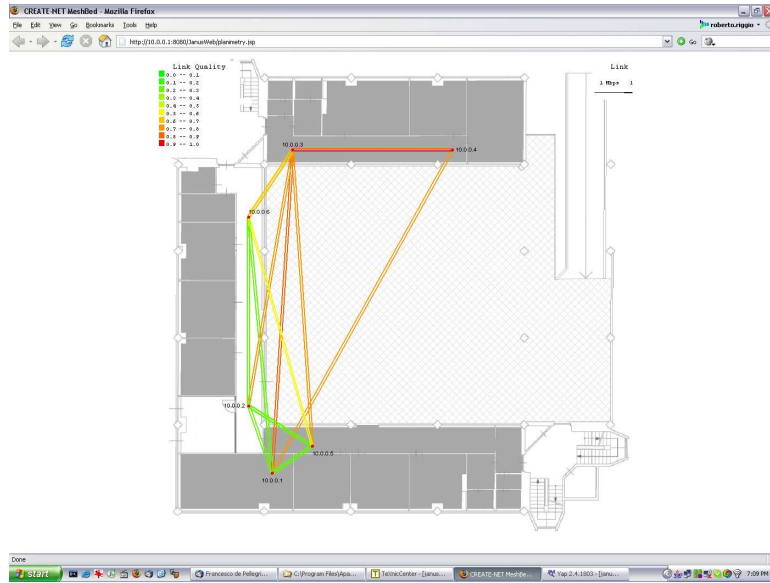


Fig. 8. JANUS: Snapshot of the network topology for the 6-nodes WMN testbed.

exploited for monitoring the topology of a 6-nodes wireless testbed deployed in a typical office environment implementing a single-tier structure (see Sec. 4). The connection made available by the **Listener** module running in each Client is exploited by a web server running on node number one in order to build a real-time map of the network topology. More specifically the web server queries the **Listener** for the Link Cache. The resulting XML document is then translated in a format compatible with GeoPlot [33]. GeoPlot is a java applet that creates a geographical image of a data set. Basically, GeoPlot plots a set of nodes and a set of lines that connect these nodes on an image specified by the user. Figure 8 shows one sample of such a map.

6.4 Mesh Knowledge Base

The Mesh Knowledge Base (MKB) is a collection of all the aspects (e.g. status of the interfaces, link cache, etc.) of the Managed Resource that can be tracked by the Agent. Such MKB is defined as a model conforming to the meta-model introduced in Sec. 5. XMI is exploited as meta-data exchange facility. Being XML-based, the XMI meta-data exchange facility allows to access the model information using its DOM (Document Object Model) [34] representation. DOM provides an object oriented application programming interface that allows parsing HTML or XML into a well defined tree structure and operating on its contents. The MKB is navigated by the Client using XPath [35] expressions. XPath is a language for addressing XML documents and can be considered as

a small query language. As for example the subtree containing the link cache is identified by the following XPath statement:

```
GET /Janus/Janus.Network/Janus.Network.ownedElement/Janus.subSequent
```

7 Conclusions

WMNs are emerging as leading paradigm for ubiquitous Internet access. Being meant as unplanned systems, WMNs are required to show advanced self-configuration and self-optimization capabilities. However, while early deployments, mostly based on the IEEE 802.11 standard, are already in place and commercial solutions are available, considerable research efforts are still required. In particular, due to their dynamic and distributed nature, WMNs pose significant management challenges claiming themselves as a natural playground for the application of Autonomic Network Management Schemes.

In this paper we proposed a novel *Knowledge Plane* tailored for the WMNs scenario and capable of enabling consistent sharing of services ontology among the entities participating the WMNs. Such features can be can be exploited for both monitoring purposes and to adapt the behavior of the node depending on the particular operating conditions. Moreover, we have developed JANUS, a novel and freely available monitoring framework exploiting the proposed Knowledge Plane.

As future work, we are planning to extend the meta-model underlying the Knowledge Plane JANUS framework in order support other networking aspects. Nevertheless, a number of issues remain open. Deploying applications on top of JANUS is likely to enrich the Knowledge Plane but, at the same time, the increasing volume of data can lead to scalability issues that need extensive investigation. In this context, we would also like to extend the prototype's capabilities by supporting incremental updates and by differentiating the messages according to their temporal and spatial characteristics. Moreover, while models can be exchanged, e.g., using XMI, the semantic meaning of the models and meta-models in each environment may be different. Ontology may serve as an appropriate tool in this context.

References

1. R. Mortier and E. Kiciman, "Autonomic network management: Some pragmatic considerations," in *Proc. of ACM SIGCOMM*, Pisa, Italy, 2006.
2. J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *IEEE Computer Magazine*, vol. 36, no. 1, pp. 41 – 50, Jan. 2003.
3. "Practical autonomic computing: Roadmap to self managing technology," IBM, Tech. Rep., 2006.
4. I. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Elsevier Computer Networks*, vol. 47, no. 4, pp. 445 – 487, Mar. 2005.
5. R. Bruno, M. Conti, and E. Gregori, "Mesh Networks: Commodity Multihop Ad Hoc Networks," *IEEE Communications Magazine*, vol. 43, no. 3, pp. 123 – 131, Mar. 2005.

6. "Ieee standards for information technology – telecommunications and information exchange between systems – local and metropolitan area network – specific requirements – part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," IEEE 802.11, 1999 Edition, 1999.
7. "Mit roofnet." [Online]. Available: <http://pdos.csail.mit.edu/roofnet/>
8. "Microsoft mesh connectivity layer." [Online]. Available: <http://research.microsoft.com/mesh/>
9. "LocustWorld." [Online]. Available: <http://www.locustworld.com/>
10. "Strix." [Online]. Available: <http://www.strix.com/>
11. "Firetide." [Online]. Available: <http://www.firtide.com/>
12. D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the internet," in *Proc. of ACM SIGCOMM*, Karlsruhe, Germany, 2003.
13. "JANUS." [Online]. Available: <http://www.wing-project.org/>
14. J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A simple network management protocol," IETF RFC 1157, May 1990. [Online]. Available: <http://www.ietf.org/rfc/rfc1157.txt>
15. J. Postel, "Internet control message protocol," IETF RFC 0792, Sep. 1981. [Online]. Available: <http://www.ietf.org/rfc/rfc0792.txt>
16. R. Enns, "Netconf configuration protocol," IETF RFC 4741, Dec. 2006. [Online]. Available: <http://tools.ietf.org/html/rfc4741>
17. Object Management Group, "Meta Object Facility (MOF) Specification, Version 1.4.1, July 2005." [Online]. Available: <http://www.omg.org/docs/formal/05-05-05.pdf>
18. —, "XML Metadata Interchange (XMI) Specification, Version 1.3, May 2003." [Online]. Available: <http://www.omg.org/docs/formal/03-05-01.pdf>
19. "MOME Project." [Online]. Available: <http://www.ist-mome.org/>
20. K. Ramachandran, E. M. Belding-Royer, and K. C. Almeroth, "DAMON: A distributed architecture for monitoring multi-hop mobile networks," in *Proc. of IEEE SECON*, Santa Clara, California, USA, 2004.
21. C. C. Ho, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer, "A scalable framework for wireless network monitoring," in *Proc. of WMASH*, Philadelphia, Pennsylvania, USA, 2004.
22. M. Conti, S. Giordano, G. Maselli, and G. Turi, "Mobileman: Mobile metropolitan ad hoc networks," in *Proc. of Eight International IFIP-TC6 Conference*, Venice, Italy, 2003.
23. R. Draves, J. Padhye, and B. Zill, "Comparison of Routing Metrics for Static Multi-Hop Wireless Networks," in *Proc. of ACM SIGCOMM*, Portland, Oregon, USA, 2004.
24. Object Management Group, "Unified Modeling Language (UML) Specification, Version 2.0, August 2005." [Online]. Available: <http://www.omg.org/docs/formal/05-07-04.pdf>
25. D. Karagiannis and H. Kühn, "Metamodelling platforms," in *Proc. of EC-Web*, Aix-en-Provence, France, 2002.
26. R. Riggio, D. Ursino, H. Kühn, and D. Karagiannis, "Interoperability in meta-environments: an xmi-based approach," in *Proc. of CAiSE*, Porto, Portugal, 2005.
27. A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proc. of IFIP/ACM International Conference on Distributed Systems Platforms*, Heidelberg, Germany, 2001.
28. M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "Scribe: A large-scale and decentralised application-level multicast infrastructure," *IEEE Journal on Selected Areas in Communication*, vol. 20, no. 8, pp. 1489 – 1499, Oct. 2002.

29. "Freepastry." [Online]. Available: <http://freepastry.org/FreePastry/>
30. D. Johnson, Y. Hu, and D. Maltz, "The dynamic source routing protocol (dsr) for mobile ad hoc networks for ipv4," IETF RFC 4728, Feb. 2007. [Online]. Available: <http://www.ietf.org/rfc/rfc4728.txt>
31. R. Draves, J. Padhye, and B. Zill, "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks," in *Proc. of ACM MOBICOM*, Philadelphia, Pennsylvania, USA, 2004.
32. D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proc. of ACM MobiCom*, San Diego, California, USA, 2003.
33. "Caida geoplot." [Online]. Available: <http://www.caida.org/tools/visualization/geoplot/>
34. W3C, "Document Object model (DOM) Level 3 Core Specification Version 1.0." [Online]. Available: <http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/>
35. —, "XML Path Language (xpath) Version 1.0." [Online]. Available: <http://www.w3.org/TR/xpath/>