JANUS: A Framework for Distributed Management of Wireless Mesh Networks

Roberto Riggio, Nicola Scalabrino, Daniele Miorandi, and Imrich Chlamtac
CREATE-NET
Via dei Solteri 38, 38100,Trento,Italy
Email: name.surname@create-net.org

***Abstract*— Wireless Mesh Networks (WMNs) are emerging as a potentially attractive access architecture for metropolitan-scale networks. While research on WMNs has been up to a large extent confined to the study of efficient routing protocols, there is a clear need to envision new network management tools, able to sufficiently exploit the peculiarities of WMNs. In particular, a new generation of middleware tools for network monitoring and profiling must be introduced in order to speed up development and testing of novel protocol architectures. Currently, management functionalities are developed using conventional centralized approaches. The distributed and self-organizing nature of WMNs suggest a transition from network monitoring to network sensing. In this work, we propose JANUS, a novel framework for distributed monitoring of WMNs. We describe the JANUS architecture, present a possible implementation based on open-source software and report some experimental measurements carried out on a small-scale testbed.**

***Index Terms*— wireless mesh networks, network management, distributed hash table, overlay networks, publish-subscribe systems**

I. Introduction

Wireless Mesh Networks (WMNs) are emerging as a potentially attractive access architecture for metropolitan-scale networks. WMNs [1], [2] rely on a multihop wireless backbone for delivering high-speed services to end-users without the need for deploying a fixed infrastructure. With respect to conventional star-shaped access network architectures, WMNs offer advantages in terms of enhanced robustness (in that no single point of failure is present and redundant links are encompassed) and flexibility (without the need for deploying cables, connectivity may be provided only where and when needed/economically attractive). With respect to conventional ad hoc networks [3], WMNs differ for (i) the *goal*, in that they are being intended as access architecture, not stand-alone systems (ii) the *heterogeneity of the devices*, in that there might be dedicated devices (with more powerful radio systems, multi-band capabilities, etc.) acting as pure wireless routers. As an example, we may consider a wireless interconnection of hot spots, providing enhanced coverage without the need of having all of them wired to the Internet.

While research of WMNs has been up to a large extent confined to the study of efficient routing protocols, there is a clear need to envision new network management tools, able to sufficiently exploit the peculiarities of WMNs. In this context, moving from a strictly layered protocol design in favor of a cross-layer architecture is a promising research direction. However, it is worth noting that, once the layering is broken, any architectural choice must be carefully analyzed in order to avoid undesirable system behaviors [4]. A new generation of middleware tools for network monitoring and profiling must then be developed in order to speed up development and testing of novel protocols and architectures. Nowadays, in network management, a set of tools, applications and devices are used in order to monitor and maintain networks. This includes performing functions such as initial network planning, frequency allocation, predetermined traffic routing to support load balancing, fault management, security management, performance management, bandwidth management, and accounting. Current management functionalities are developed using a strongly centralized approach. However, 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). The advantages provided by such an approach are twofold: (i) network operators can gain an increased insight into network behavior and resource utilization, to optimize system dimensioning and performance (ii) the research community is provided with a powerful tool for the rapid prototyping of innovative solutions.

In this work, we propose JANUS, a novel monitoring framework designed with the goal of addressing the peculiarities of WMNs. The proposed approach exploits the functionalities provided by Pastry [5], 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). In this paper, we describe the JANUS architecture, present its implementation details and report some experimental measurements carried out on a small-scale testbed deployed at the CREATE-NET premises. The experimental measurements are aimed at characterizing the impact of the traffic generated by JANUS on both best-effort elastic traffic as well as VoIP applications. All the developed software has

This work was partially supported by MIUR within the framework of the WING project (grant number RBIN04M292). On line resources available at http://www.wing-project.com/.

been released under the BSD License[1].

The remainder of this paper is organized as follows. Section II presents and discusses some related works. In Sec. III, we overview the state-of-the-art in WMNs and describe some of the main Pastry features. Section IV describes the architecture of the JANUS module. The detailed implementation is illustrated in Sec. V. Section VI describes the experimental setup of our small-scale testbed. Section VII concludes the paper pointing out directions for future work.

II. Related Work

A large number of protocols exists to support network and network devices management. Common protocols include SNMP [6], ICMP [7], and netconf [8]. The MOME project [9] is maintaining a database of more that 400 measurement tools. However most network management architectures are based on common concepts and architectures. In this context, all nodes in the network run a process gathering information about the node's state. When a problem is recognized, such process is designed to send alerts to some management entities. Upon receiving these alerts, management entities are programmed to react by taking some actions (e.g., operator notification, event logging, system shutdown, etc.). Management entities also can 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 enabling effective network management. In SNMP each managed node owns a Management Information Base (MIB). A MIB is a collection of information that is organized hierarchically. MIBs are accessed using SNMP. Our approach is a step toward the definition of a mesh-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).

In [10] DAMON, a Distributed Architecture for Monitoring Mobile Networks, is introduced. DAMON relies on agents within the network to actively monitor network behavior and send this information to data repositories. DAMON's generic architecture supports the monitoring of any protocol, device, or network parameter. The prototype presented in the paper is exploited for collecting statistics about data traffic and the AODV routing protocol. VISUM [11] is a distributed framework for monitoring wireless networks. Data, collected by agents distributed over several host, is collected at a centralized repository and can be exploited by a visualization tool for real-time monitoring. The novelty of our approach lies in the fully distributed paradigm enabling decentralized control, self-organization, and scalability.

In MobileMAN [12], information collected at different layers of the networking stack are shared in a common local memory structure and exploited to adapt the behavior of the node depending on the particular circumstance the node operates in. 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 standards, since it does not change the core functionalities of each layer maintaining all the advantages of a modular architecture.

[1]http://www.wing-project.org/

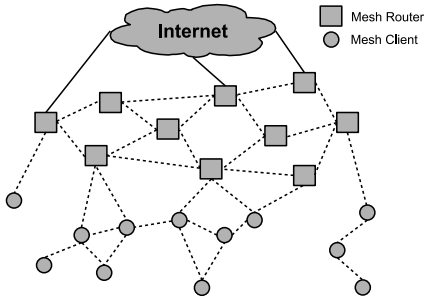


Fig. 1. An example of a two-tier wireless mesh network architecture.

III. Technical Background

A. Wireless Mesh Networks

A Wireless Mesh Network consists of several nodes (possibly employing multiple radio interfaces [13]) interconnected via wireless links to the Internet through one or multiple gateway(s). Communications take place by means of multihopping, in that the nodes in the network cooperate to forward packets (by means of store-and-forward operations) to/from the Internet and from/to the end node.

Nodes in a WMN can play two different logical roles, i.e., mesh clients and mesh routers [1]. Mesh clients can be the source/destination of connections, while mesh routers are in charge of forwarding packets to and from the Internet. Multitier architectures can be envisaged [2], with mesh routers providing multihop backhaul connectivity to the Internet, while the clients act just as sources/destinations of Internet connections. Such a kind or architecture is sketched in Fig. 1. It is worth stressing that, from our standpoint, WMNs are to be thought as *access network architectures*, and not as standalone "ad hoc" systems.

B. Pastry & Scribe

Peer-to-Peer overlay networks can be classified into two classes: unstructured peer-to-peer networks and structured ones. The core of such distinction lies in the way the data is assigned to each node. Unstructured peer-to-peer systems, such as Gnutella [14], do not assign responsibility for data to specific nodes. Data look-up is performed trough flooding or random walks. In the former approach the query is propagated to all the neighbors, in the latter approach the forwarders are chosen randomly. Both approach are characterized by a high network load. Moreover they cannot ensure both query termination and data location.

On the opposite, structured peer-to-peer networks organize themselves using a controlled topology. Both data distribution and lookup algorithms are based on a Distributed Hash Table (DHT) data structure. DHTs are a class of decentralized

systems that distribute a set of keys among participating nodes, and efficiently route messages to the owner of any given key. The first achievement of such an architecture is that the node that is responsible for a key can always be found. DHT-based peer-to-peer networks include CAN [15], Chord [16], Pastry [5], Tapestry [17] and Kadmelia [18]. On the other hand, structured peer-to-peer networks are not able to support keyword based searches since they are designed for exact match queries. Moreover the strictly controlled topology incur into an high overhead for repair operation for high churn rates.

Pastry is a self-organizing structured peer-to-peer overlay network. Each node participating the Pastry network in identified by a unique 128-bit `nodeId`. The `nodeId` set is uniformly distributed. Such a feature is typically achieved by hashing the node's IP address. The approach used by Pastry for routing messages is based on address prefixes and is a variation of hypercube routing [19]. The routing algorithm works by resolving a single digit at time from left to right. Basically at each step a node forwards the message to a node whose `nodeId` shares with the key a prefix that is at least one digit longer than the the prefix that the key shares with the current node. If such a node cannot be found the message is delivered to the node with the closest `nodeId`. The Pastry design assure that such a routing can be accomplished in less than $log_{2^b} N$ steps, where N is the number of nodes and b is a configuration parameter with typical value of 4.

Scribe [20] is a scalable group communication system allowing participants to subscribe to a topic and to publish messages. Scribe exploit 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 the publication of message for a specific topic.

IV. The JANUS Module

We start by describing the components of the SNMP management framework. An SNMP-managed network consists of four key components:

- *Managed devices.* A managed device is a network node that contains an SNMP agent and that resides on a managed network. Managed devices collect and store management information and make this information available to NMSs using SNMP. Managed devices can be routers, switches, hosts, etc.
- *Agents.* An agent is a network-management software module that resides in a managed device. An agent has local knowledge of management information and translates that information into a form compatible with SNMP.
- *Network Management Systems (NMSs).* A NMS is in charge for monitoring and controlling managed devices. In a SNMP deployment, the NMS is responsible for polling and receiving traps from the Agents as well as for changing the values of variables stored within managed devices.

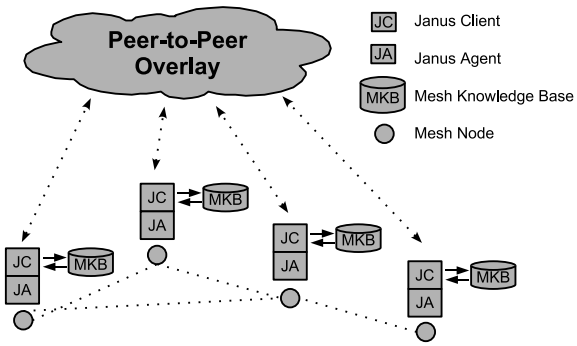


Fig. 2. JANUS Architecture.

- *Management Information Base (MIB).* A Management Information Base (MIB) is a collection of information that is organized hierarchically. MIBs are accessed using a network-management protocol such as SNMP.

The building blocks of JANUS and their relationships are sketched in Fig. 2. The JANUS architecture can be considered as a distributed version of the SNMP management framework, and consists of four components:

- *Mesh Node.* The Mesh Node (from now on *Managed device*) is a network node participating the WMN that runs a JANUS Agent. The node can be either a mesh node or a mesh router.
- *JANUS Agent.* The JANUS Agent (from now on *Agent*) is a software process that runs on a managed device. An Agent has a local knowledge of the network and provides management information to the *JANUS Client* by keeping track of the various aspect of the managed device. 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 events. It is worth noting the relevant information are gathered by an implementation specific module inside the Agent that can interact with the host operating system.
- *Mesh Knowledge Base (MKB).* All the Agents share a list of *objects* that they can monitor. An object is a particular aspect of the managed device (e.g. the status of each interfaces, the link cache, ...). The MKB is a database of all the objects that an Agent can track. Managed objects are organized hierarchically in a tree-like structure. Figure 3 shows the objects tree that is currently supported by the JANUS framework.
- *JANUS Client.* The JANUS Client (from now on *Client*) is a software process running on each managed device, it can be considered as the distributed version of the NMS.

At initialization time each Client tries to connect to a bootstrap host in order to subscribe a Scribe topic. If no bootstrap host is found the Client starts a new Scribe topic. Please note that the bootstrap host is required only at initialization time. Each *Client* periodically queries the *Agent* in order to gather

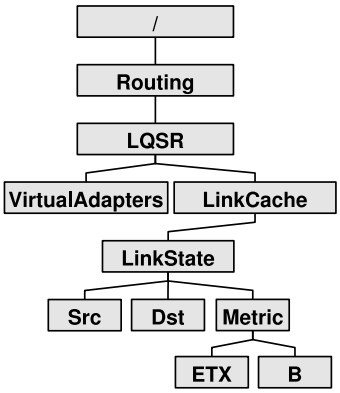


Fig. 3. MKB Objects tree.

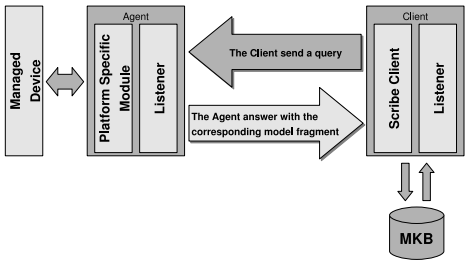


Fig. 4. A detailed view of the interaction between Agent and Client.

the managed objects. Gathered objects are used for updating the MKB and then published on the Scribe ring. When objects are delivered the Client takes care of merging them with the local MKB (during the bootstrap phase the MKB is initialized with the locally collected objects). A detailed view of the interactions between each JANUS components is shown in Fig. 4.

V. Implementation Details

JANUS is implemented in the form of a Java application built on top of Scribe. FreePastry is used as reference platform. FreePastry [21] is an open-source implementation of Pastry's API from Rice University, which comprises, among the other, a Scribe implementation.

A. Mesh Node

Mesh connectivity is realized using the Microsoft Mesh Connectivity Layer (MCL) [22]. The MCL is a loadable Microsoft Windows driver. It implements an interposition layer between layer 2 (the link layer) and layer 3 (the network layer) of the standard ISO/OSI model. It is sometimes referred to as layer 2.5 (see Fig. 5). To the higher layers, MCL appears to be just another Ethernet link, albeit a virtual one. To the lower layers, MCL appears to be just another protocol running over the physical link. MCL routes using a modified version of DSR [23] called Link Quality Source Routing (LQSR) [13]. 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 over that link. In addition, the channel, the bandwidth, and the loss

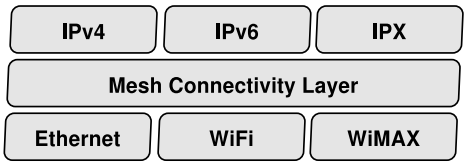


Fig. 5. Microsoft's MCL architecture.

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) [24], a variant of the Expected Transmission Time (ETT) metric [25], to define the best path for the transmission of data from a given source to a given destination.

B. JANUS Agent

Each node participating the WMN runs an instance of the JANUS Agent. The Agent is implemented in the form of a software process waiting for incoming connection on the port `1167`. In the current implementation traps from the Agent to the Client are not supported. Links status is obtained by querying the link cache using the *mcl.exe* utility that comes with MCL. This utility runs from the command-line and can be used to install, configure, and analyze MCL virtual adapters. A sample output of this command is shown in Fig. 6.

```
VA 1 Link Cache, 2 nodes of 4, Timeout 0s
ec-a0-b9-dc-76-c5: Hops 0 DMetric 0 CMetric 0 Prev -1
    From 2 To 1/1c-12-78-6c-83-c0
        TimeStamp -3s Usage 11 Metric 0.17-909Kbps-1 (11.15ms)
        Queue Drops 0 Failures 0 fraction 0
        ETX:
        TotSentProbes 11 LastProb 0.000
        PktPair:
        PairsSent 1 RepliesRcvd 1
        LastPktPair 9.57ms CurrMin 9.57ms
        NumValid 1 NumInvalid 0
1c-12-78-6c-83-c0: Hops 1 DMetric 11.15ms CMetric 11.15ms Prev 0
    From 1 To 2/ec-a0-b9-dc-76-c5
        TimeStamp 0s Usage 11 Metric 0.16-11Mbps-1 (1.11ms)
        RepliesSent 1
        TotRcvdProbes 15 FwdDeliv 13 RevDeliv 15
```

Fig. 6. Node's Link Cache as reported by the *mcl.exe* utility.

```
<?XML version="1.0"?>
<JANUS>
  <Routing>
    <LQSR>
      <LinkCache>
        <LinkState>
          <Src>ec-a0-b9-dc-76-c5</Src>
          <Dst>1c-12-78-6c-83-c0</Dst>
          <Metric>
            <ETX>0.17</ETX>
            <Bandwidth>909Kbps</Bandwidth>
          </Metric>
        </LinkState>
      </LinkCache>
    <LQSR>
  </Routing>
</JANUS>
```

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

The output of this command is then parsed by the `Platform Specific Module` in order build a structured version (see Fig. 7) of the link cache content that can be then exploited by the `Listener` in order to respond to the Client's queries. 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) [25] and measures the expected number of transmissions, including retransmissions, needed to send a unicast packet across a link.
- B is the link bandwidth.

Given a packet of size S sent over a link with raw data rate B, the ETT metric can be computed as follows:

$$ETT = ETX * \frac{S}{B}.$$

C. Mesh Knowledge Base

The MKB organizes all managed objects in a tree-like structure (see Fig. 3) represented by means of an XML document [26]. Such XML document is accessed using its DOM (Document Object Model) [27] 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 [28] 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:

```
/JANUS/Routing/LQSR/LinkCache
```

which selects LinkCache elements that are children of LQSR elements that are children of the Routing element that forms the outermost element of the XML document. XPath syntax is designed to mimic file path syntax.

D. JANUS Client

The `ScribeClient` module of the Client takes care of polling the Agent on port 1167 using XPath statements. The Agent replies with the MKB objects tree that correspond to the incoming query. The gathered objects are then published on the subscribed Scribe topic. The `Listener` is a software process waiting for incoming connection on port `1169`. It can be exploited by an external application for querying the node's MKB.

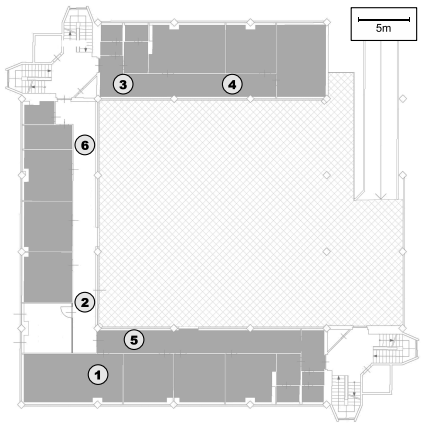


Fig. 8. Testbed planimetry.

VI. Experimental Setup

We implemented a version of JANUS on a small-scale WMN testbed developed at the CREATE-NET premises. We first tested JANUS for controlling the topology of the network (including indication of link strength). Then we carried out some experiments to infer the impact of the traffic generated by JANUS on both elastic data traffic as well as VoIP connections. The aim was to understand whether the JANUS architecture was lightweight enough to allow its use in running WMN deployments.

A. Network Configuration

JANUS has been exploited for monitoring the topology of a 6-nodes wireless testbed deployed in a typical office environment implementing a single-tier structure. Testbed's nodes are all Dell notebook model D600/D610/D810 equipped with a 1.86 GHz Intel Pentium M processor with 512 MB of memory. All nodes run Microsoft Windows XP Professional. Each node has a single Intel 2915ABG or a Dell 1470 Wireless adapter with RTC/CTS disabled. The testbed planimetry is illustrated in Fig. 8. Node number one acts as gateway providing Internet connectivity to the WMN. All measurement are run using IPv4 with statically assigned addresses and IEEE 802.11 operating in "g" mode. In order to increase the reliability of our results, we have exploited NetStumbler [29]. NetStumbler is a tool that allows to detect the presence of interference caused by other 802.11 devices. The operating channel for the WMN has been chosen according to this analysis.

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 [30]. 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 9 shows an example of such a map. While the information provided by the current implementation of JANUS

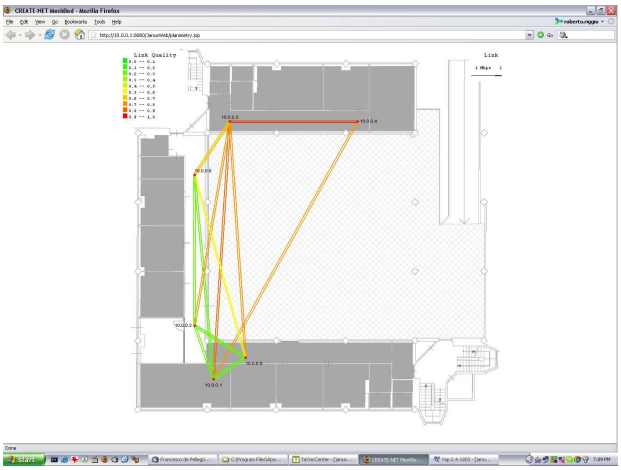


Fig. 9. JANUS: Snapshot of the network topology for the 6-nodes WMN testbed.

	VoIP (G.729.3)
Rate (Packets/sec)	33
Payload length (Bytes)	30

TABLE I

PARAMETERS CHARACTERIZING THE FLOWS USED FOR ANALYZING THE OVERHEAD INTRODUCED BY JANUS

are limited to links status, its generic architecture potentially supports the monitoring of a wide set of network protocols and devices.

B. Multimedia Traffic Patterns

In order to evaluate the overhead introduced by JANUS we analyzed the performance of the mesh using both real time and best effort traffic. The real time traffic has been modeled as a single UDP packet flow representing a voice stream encoded using the G.729.3 codec [31], a worldwide used speech codec for VoIP applications, with each packet containing three voice samples and without Voice Activity Detection. On the other hand, best effort traffic (in our case persistent TCP connections) is modeled considering a TCP socket working in saturation regime. The parameters for VoIP flows are reported in Table I. In order to collect reliable measure of delays, before each experiment we synchronized each node with a common reference using NTP [32].

The experimentation had been performed using synthetic traffic generated by means of the Distributed Internet Traffic Generator (D-ITG) [33], a freely available software tool. D-ITG can generate and inject different traffic patterns over TCP and/or UDP sockets. The traffic is then collected at the receiver side where suitable tools can provide a great variety of statistical analysis. By means of D-ITG it is possible to simulate many traffic scenarios originated by a large number of users and network devices, whereas other traffic generators have limited capabilities in terms of performance and range of source models.

C. Performance Measurements

In this section we report the outcomes of some experimental tests run with the equipment and settings described in Sec. VI-A. As said before, the aim of such experiments is to gain insight into the performance impairments experienced by both elastic and multimedia traffic in the presence of JANUS. The tests reported refer to downlink traffic only, i.e., traffic coming from the Internet, entering the mesh through the gateway and destined to nodes of the WMN. The nodes are activated according to the numbering in Fig. 8, so that when N flows are active hosts $2, 3, \ldots, N+1$ are downloading from host 1.

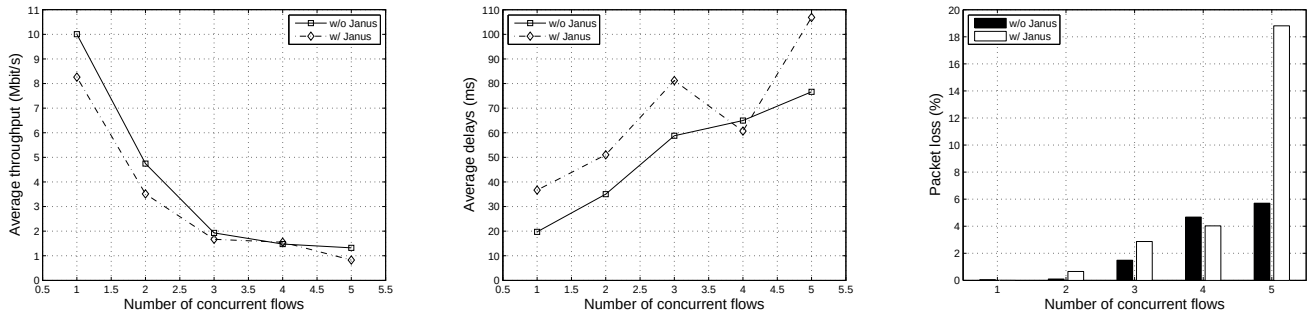
In the case of best effort traffic we will exploit the mean per-connection throughput as performance metric. The results are reported in Fig. 10(a). In the case of a homogeneous network, we would expect a $\frac{1}{n}$ pattern, n being the number of active connections; the results reported show a similar behavior. In general, the traffic generated by JANUS turns out to have a minor impact of the performance of elastic (TCP-controlled) traffic. In one situation (with 4 active connections), we even measured a performance improvement with respect to the case where JANUS is not active. We conjecture that this could be done to a sudden change in the environmental conditions, which led to results (for the case with JANUS) not fully reliable for such case. We can therefore conclude that JANUS does not affect much elastic traffic.

We also considered the performance under VoIP-like traffic, in order to assess the impact of JANUS on the packet delay and packet loss ratio, the most important performance metric for characterizing the performance of real-time applications. The VoIP traffic was modeled according to the parameters detailed in Sec. VI-B. The results for the average delay are reported in Fig. 10(b). In this case, we can see that the load generated by JANUS increases the average packet delay by approximately $15 - 20$ ms. Also in this situation, the case with 4 connections returned unreliable values (the average delay with JANUS was lower than that measured on the network without it). This suggests that a careful tuning of connection admission control may be necessary if JANUS is to be employed on WMNs running delay-sensitive applications.

Similar conclusions may be drawn from the packet loss ratios, reported in Fig. 10(c). Also in this case (again, apart for the scenario with 4 active connections) the use of JANUS entails an increase in the packet loss ratio. Such increase is limited for the cases of $1, 2$ and 3 connections, while it is considerable (of the order of 15%) for the scenario with 5 connections. We conjecture that this may be due to the fact that the network has been operated close to the stability limit.

VII. CONCLUSIONS

WMNs are emerging as a promising architecture for building metropolitan-scale access networks. In this paper we have proposed JANUS, a distributed monitoring architecture for wireless mesh networks, based on a structured peer-to-peer overlay network. We proposed an implementation, based on open-source software, and presented experimental measurements, showing the impact of the traffic load generated by



(a) TCP average throughput versus number of concurrent flows w/o and w/ the JANUS system.

(b) Average delays versus number of concurrent flows w/o and w/ the JANUS system.

(c) Packet loss for six concurrent Multimedia Flows w/o and w/ the JANUS system.

Fig. 10. Outcome of the measurements campaign exploiting both best effort and real time traffic patterns.

JANUS on both elastic as well as time-sensitive applications. It turns out that the traffic generated by JANUS has little impact on TCP-controlled traffic; on the other hand the increase in average packet delay experienced by VoIP connections suggests the necessity of a careful tuning of admission control strategies for multimedia flows.

As future work, we are planning to extend the JANUS framework in order to monitor other networking aspects. At the same time, we would also like to extend the prototype's capabilities by supporting incremental updates and by differentiating the messages according to their temporal characteristics.

REFERENCES

[1] I. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Elsevier Computer Networks*, vol. 47, no. 4, pp. 445 – 487, Mar. 2005.

[2] 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.

[3] I. Chlamtac, M. Conti, and J. Liu, "Mobile ad hoc networking: imperatives and challenges," *Elsevier Ad Hoc Networks*, vol. 1, no. 1, pp. 13 – 64, Jan. 2003.

[4] V. Kawadia and P. R. Kumar, "A cautionary perspective on cross-layer design," *IEEE Wireless Communication Magazine*, vol. 12, no. 1, pp. 3 – 11, Feb. 2005.

[5] 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.

[6] IETF, "A simple network management protocol." [Online]. Available: http://www.ietf.org/rfc/rfc1157.txt

[7] ——, "Internet control message protocol." [Online]. Available: http://www.ietf.org/rfc/rfc0792.txt

[8] ——, "Netconf configuration protocol." [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-netconf-prot-12.txt

[9] "MOME Project." [Online]. Available: http://www.ist-mome.org/

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] E. Adar and B. Huberman, "Free riding on gnutella," Xerox PARC, Tech. Rep., 2000.

[15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proc. of ACM SIGCOMM*, San Diego, CA, USA, 2001.

[16] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17 – 32, Feb. 2003.

[17] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 41 – 53, Jan. 2004.

[18] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," in *Proc. of IPTPS02*, Cambridge, MA, USA, 2002.

[19] C. G. Plaxton, R. Rajaraman, and A. W. Richa, "Accessing nearby copies of replicated objects in a distributed environment," in *Proc. of ACM SPAA*, Newport, USA, 1997.

[20] 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.

[21] Rice University, "Freepastry." [Online]. Available: http://freepastry.org/FreePastry/

[22] Microsoft, "MCL." [Online]. Available: http://research.microsoft.com/mesh/

[23] IETF, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)." [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt

[24] 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.

[25] 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.

[26] W3C, "Extensible Markup Language (XML) 1.0 (Fourth edition)." [Online]. Available: http://www.w3.org/TR/xml/

[27] ——, "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/

[28] ——, "XML Path Language (xpath) Version 1.0." [Online]. Available: http://www.w3.org/TR/xpath/

[29] "NetStumbler." [Online]. Available: http://netstumbler.com/

[30] CAIDA, "Geoplot." [Online]. Available: http://www.caida.org/tools/visualization/geoplot/

[31] "ITU-T Recommendation G.729 Annex B, A silence compression scheme for G.729 optimized for terminals conforming to Recommendation V.70," Nov. 1996.

[32] "Simple Network Time Protocol (SNTP) Version 4." [Online]. Available: http://www.apps.ietf.org/rfc/rfc2030.html

[33] "D-ITG, Distributed Internet Traffic Generator." [Online]. Available: http://www.grid.unina.it/software/ITG/