Using MPLS in a Wireless Mesh Network to Improve the Resiliency of SCADA Systems

Stefano Avallone[1,2] and Salvatore D'Antonio[3,2]

[1] University of Naples "Federico II", Via Claudio 21, 80125 Naples, Italy
stefano.avallone@unina.it
[2] CINI Consorzio Nazionale Interuniversitario per l'Informatica, Italy
[3] University of Naples "Parthenope", Centro Direzionale di Napoli, Isola C4, 80143 Naples, Italy
salvatore.dantonio@uniparthenope.it

Abstract. Critical infrastructure are often and often being managed by SCADA systems that communicates using the *de facto* standard TCP/IP suite of protocols. Given the recent advances in wireless communications and the cost-effectiveness of deploying a multi-hop wireless network, we also foresee the use of wireless networks as the communication infrastructure of SCADA systems in the near future. In order for that to happen, the needed requirements of SCADA systems in terms of performance, reliability and resiliency must be ensured. To this end, a critical role is played by the routing protocol. In a previous work, we proposed a forwarding paradigm for wireless networks which was shown to be robust against node/link failures. However, such a forwarding paradigm cannot be used as-is in the case of SCADA systems due to its loose control over the network traffic. In this paper, we present an alternative implementation of such forwarding paradigm, which uses MPLS – a widely adopted technology in the wired world – as its forwarding engine and describe how such solution solves the issues presented by the original version of our forwarding paradigm.

Keywords: SCADA systems, network resiliency, MPLS, wireless mesh networks.

1 Introduction

Critical infrastructures are usually managed with the aid of SCADA (Supervisory Control And Data Acquisition) systems, which consist of several RTUs (Remote Terminal Units) sensing physical parameters and servers where human operators can have a picture of the real-time situation. RTUs are typically sensors that monitor physical parameters of interest and transmit the sensed data to the SCADA servers. It is therefore clear that the communication between RTUs and servers plays a fundamental role in the ability to keep the system under control and promptly react to failures. It is indeed needed a fast, secure and reliable communication between RTUs and SCADA servers. Historically, such communication has been ruled by proprietary solutions. More recently, however, the

P. Chatzimisios et al. (Eds.): MobiLight 2010, LNICST 45, pp. 440–450, 2010.

widespread adoption of the TCP/IP suite of protocols has pushed a big technological change in the communication infrastructure of SCADA networks. In the last years, indeed, a growing number of SCADA systems are being realized (or converted to) using the standard TCP/IP suite of protocols. Such a move brings many advantages, such as for instance a large base of well established protocols. They are well known protocols, which makes it easier to design, maintain and troubleshoot the communication infrastructure. Also, it is easier to interconnect different sites through the public Internet.

There is, however, the other side of the coin. The use of well known protocols and the interconnection with the public Internet expose the SCADA systems to malicious attacks. It is thus of utmost importance to guarantee the secure operation of SCADA systems while exploiting the advantages of the use of standard protocols. Also, the TCP/IP suite of protocols has been designed for a computer network offering data delivery in a best effort manner, while SCADA systems have precise requirements in terms of reliability and timeliness of the delivery of messages. As a consequence, it is necessary to employ all the instruments that have been designed in the TCP/IP world to meet such requirements.

In this paper, we focus on the routing layer of the communication infrastructure, which can play a key role in achieving the goal of having the TCP/IP suite of protocols meet the requirements of SCADA systems. Indeed, routing determines the path taken by packets and hence it may allow to:

- find a proper path that ensures timeliness in the delivery of messages
- re-route packets around a failed node/link
- re-route packets around an attacked node

Thus, routing can help make a SCADA system resilient to both failures and attacks, while at the same time achieving their communication requirements.

Moreover, the communication between RTUs and SCADA servers has typically made use of a cabled infrastructure. With the growing spread and continuos enhancements of wireless communications, we also foresee the use of a wireless infrastructure to transport messages between RTUs and SCADA servers. A wireless network has many advantages over the cabled counterpart, such as low cost and easiness of deployment and maintenance due to lack of cables and speed of repairing (think of cables inadvertently cut by humans or by animals). Given the emergence of multi-hop techniques, it is nowadays possible to cover large areas using wireless technologies. Wireless networks where nodes are not mobile and form a backbone that routes packets generated by possibly mobile clients are denoted as wireless mesh networks [1]. In fact, there are more and more cases of city-wide wireless mesh networks witnessing the potential of such technology. Thus, we envisage the possible use of wireless mesh networks as wireless communication infrastructures for SCADA systems.

The main challenge when dealing with wireless networks is that interference exists between simultaneous transmissions taking place on the same frequency channel and in the same neighborhood. Thus, the routing algorithm should also take interference into account while pursuing the above mentioned objectives. In [2], we proposed a Layer-2.5 forwarding protocol with the goal of achieving

high throughput and providing fast reaction to node/link failures, though the focus was not on SCADA systems. In this paper, we design an enhancement of the Layer-2.5 forwarding protocol to overcome some issues pointed out in [2] and increase its robustness, in order for it to address the hard requirements of SCADA systems. In particular, we show how MPLS (Multi-Protocol Label Switching) [3], a forwarding mechanism designed for and widely adopted in wired networks, can be effectively used to enhance the performance of our routing protocol for wireless mesh networks.

The rest of the paper is structured as follows. Section 2 briefly introduces how MPLS works, while Section 3 gives an overview of the characteristics of wireless mesh networks. Section 4 presents some details of the previously proposed Layer-2.5 forwarding paradigm and analyzes some of its issues. Section 5 describes the proposed implementation of the Layer-2.5 forwarding paradigm by using MPLS as its forwarding engine. Finally, Section 6 concludes this paper.

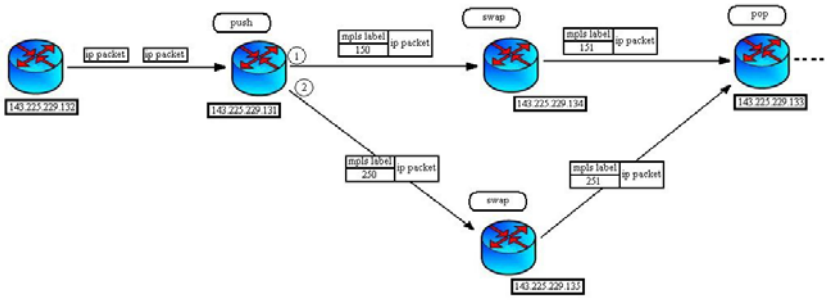


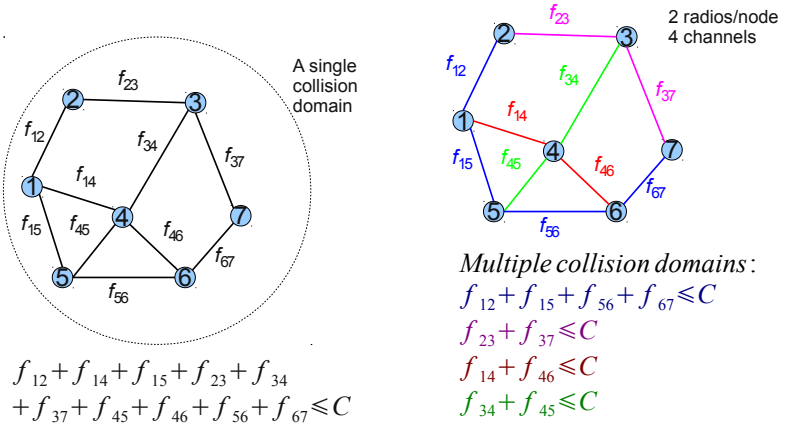
Fig. 1. Label switching in an MPLS network

2 Multi-Protocol Label Switching

MPLS is a forwarding paradigm that sits between the IP layer (or other layer-3 protocols) and a data link technology, hence it is often referred to as a layer-2.5 protocol. MPLS makes use of virtual circuits, that are denoted as Label Switched Paths (LSPs) in the MPLS jargon, and the forwarding is based on the switching of labels at every hop (fig. 1). For this purpose, a shim header is inserted between the layer-3 header and the layer-2 header which includes a 20-bit label that determines the next hop and the new label to be inserted into the packet. Clearly, it follows that, in an MPLS network, a packet is always processed by the MPLS entities and it is never passed to the IP entities, but when exiting the MPLS network. Thus, the path taken by packets is determined by how the MPLS forwarding tables have been configured. With respect to the IP routing, the MPLS forwarding enables two important features that give MPLS a great flexibility. First, packets are not constrained to follow the least cost path to the destination, but can follow any path, provided that labels have been properly configured. Second, while in the IP routing a Forwarding Equivalence Class (FEC), i.e., the set of all the packets that are routed in the same manner, is

represented by all the packets destined to the same node, in the MPLS forwarding a FEC can be identified by different criteria including source node, source and destination ports and a combination of them. Thus, in an MPLS network, it is possible that packets destined to the same node but having, e.g., different destination ports are routed along distinct paths.

Another important feature of MPLS is that it allows fast mechanisms to recover from a node/link failure (possibly due to an attack). When a failure occurs in an IP network, it is necessary to wait for a node to detect the failure and then for the convergence time, i.e., the time necessary for nodes to exchange routing messages and have all a consistent view of the network topology. With MPLS, instead, it is possible to pre-configure alternative paths, denoted as backup paths, that can be used in case of failure of the active paths. The down-time is thus restricted to the time necessary for a node to detect the failure. There are several possible mechanisms for MPLS restoration. For instance, the backup path may be path disjoint or single node/link disjoint with respect to the active path. In the former case, the active path and the backup path do not share any link or node. In the latter case, different backup paths are pre-established, each of which protects against a failure of a single link or a single node.



(a) Single-radio case (b) Multi-radio case

Fig. 2. Constraints on the available bandwidth

3 Wireless Mesh Networks

Wireless mesh networks have emerged as an infrastructure to create wireless backbones that extend the coverage of traditional WLANs by routing packets from one mesh router to another through multiple wireless hops. As wireless transmissions are involved, interference is to be taken into account. The main effect of interference is to prevent simultaneous transmissions to take place on the same frequency channel, which implies that a node cannot use the whole

capacity of the wireless link, but all the nodes in a neighborhood have to share the channel capacity. Thus, the main difference between the wired scenario and the wireless scenario is that in the latter one the bandwidth available on a link is less than the link capacity. This situation is illustrated in Fig. 2a, where we assume that all the links belong to the same collision domain and denote by f_{ij} the bandwidth available on link $i \rightarrow j$. If all the nodes have a single radio and therefore transmit on the same frequency channel, then the channel capacity must be shared among all the links, and hence the sum of the available bandwidth on all the links must not exceed the channel capacity C.

Given the recent availability of cost-effective wireless network interface cards, multi-radio devices have been introduced [4,5] in order to increase the throughput of a wireless mesh network. Indeed, wireless standards provide multiple orthogonal channels (e.g., the IEEE 802.11b/g and IEEE 802.11a standards define, respectively, 3 and 12 non-overlapping channels) and mesh nodes can exploit the availability of multiple radios to transmit/receive simultaneously on non-interfering channels. Figure 2b shows the improvement in performance brought by multiple radios. If different links in a neighborhood transmit on distinct channels, the result is a splitting in multiple collision domains, each associated with a channel. In each collision domain the constraint that the sum of the bandwidth available on the links must not exceed the channel capacity still holds. However, as fig. 2b shows, collision domains are now smaller with respect to the single-radio case, which allows for higher link bandwidth availability and hence higher network throughput.

Figure 2 clearly shows that the amount of bandwidth available on each link depends on how channels are assigned to links, which determines the formation of collision domains. Since routing should be tailored to the bandwidth available on links, it turns out that channel assignment and routing are strictly interdependent on each other. In fact, a conspicuous amount of papers [6,7,8,2,9,10] have been published addressing the joint channel assignment and routing problem. In particular, in [2] we proposed a joint channel assignment and routing algorithm which computes a channel assignment and the resulting available bandwidth on each link, Also, we proposed a forwarding paradigm, denoted as Layer-2.5 forwarding paradigm, to route packets in order to utilize network links according to the computed bandwidth availablility. Such forwarding paradigm is illustrated in some details in the next section.

4 A Layer-2.5 Forwarding Paradigm

In this section, we provide some details on the Layer-2.5 forwarding paradigm (L2.5) we proposed in [2]. We refer to [2] for more details. L2.5 requires that a solution to the joint channel assignment and routing problem has been found and thus each radio is assigned a channel and each wireless link is associated with a flow rate value. The goal is to enable every router to utilize each of its links in proportion to their flow rates. We underline that the operations of our forwarding paradigm do not involve the use of destination-based routing tables.

Each router is not required to have a complete knowledge of the network topology, but only the information about the minimum hop count to each possible destination in the WMN. We denote by $\boldsymbol{HC}_u$ the *hop count* vector of node u, whose component $HC_u(d)$ represents the hop count from u to the destination d. A simple procedure for having the routers build their own hop count vectors is described in [2]. Such hop count vectors, however, are not directly used to forward packets along the shortest paths. Instead, each router u sets a timer which expires at regular intervals and records the amount of bytes $b(v)$ sent to each neighbor v since the last timer expiration. Every time u has to take a forwarding decision, it computes the gap $\Delta_u(v)$ between the desired and the current utilization of link $u \rightarrow v$ for each neighbor v:

$$\Delta_u(v) = \frac{f(u \rightarrow v)}{\sum_{\forall u \rightarrow i} f(u \rightarrow i)} - \frac{b(v)}{\sum_{\forall u \rightarrow i} b(i)}$$

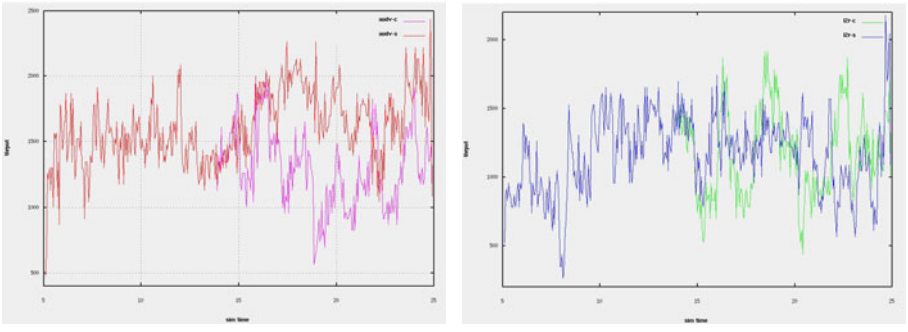
In the attempt to keep the transmission rate on all of its links proportional to the corresponding flow rate, router u will send the packet to the neighbor node having the largest $\Delta_u(v)$ value. Clearly, a router cannot decide the next hop of a packet based only on the $\Delta_u(v)$ values, as packets could take extremely long paths to get to destination. This is where the hop count vectors come into play. In the process of building its own hop count vector, each router also learns the hop count of its neighbors to every destination. This information is used to partition the neighbors based on the hop count to a certain destination. For a generic destination d, the set $\mathfrak{N}(u)$ of neighbors of node u is partitioned into three sets: $\mathfrak{N}_d^{=}(u)$ (containing the neighbors with the same hop count to d as u), $\mathfrak{N}_d^{+}(u)$ (containing the neighbors with u's hop count to d plus 1) and $\mathfrak{N}_d^{-}(u)$ (containing the neighbors with u's hop count to d minus 1).

L2.5 provides that every source node s in the WMN puts a *maximum hop count* value into the HC^{max} field of the *L2.5 header* of each packet it sends. Such a value equals the minimum hop count to the destination multiplied by a constant factor $\alpha > 1$. The value into the HC^{max} field is decremented at each intermediate hop and is used to determine the set of candidate next hop neighbors for a packet. Indeed, each intermediate router must take a forwarding decision that allows the packet to reach the destination within HC^{max} hops. Thus, for instance, if the value in the HC^{max} field equals the minimum hop count to the destination for the intermediate node u, then the packet must be necessarily sent to a neighbor in $\mathfrak{N}_d^{-}(u)$ and will thus follow a minimum hop path. Otherwise, the packet may also be sent to neighbors in $\mathfrak{N}_d^{=}(u)$ or even $\mathfrak{N}_d^{+}(u)$. The neighbor v with the maximum $\Delta_u(v)$ among those in the set of candidate next hop neighbors is then selected.

More precisely, a router u receiving a packet from node w with maximum hop count HC_u^{max} and destined to node d, determines the set $\mathfrak{S}$ of candidate next hop neighbors as follows:

$$\mathfrak{S} = \begin{cases} \mathfrak{N}(u) - \{w\} & \text{if } HC_u^{max} > HC_u(d) + 1, \\ \mathfrak{N}_d^{=}(u) \cup \mathfrak{N}_d^{-}(u) - \{w\} & \text{if } HC_u^{max} = HC_u(d) + 1, \\ \mathfrak{N}_d^{-}(u) - \{w\} & \text{if } HC_u^{max} = HC_u(d). \end{cases} \quad (1)$$

According to L2.5, thus, a router u determines the set $\mathfrak{S}$ of candidate neighbors, computes $\Delta_u(v)$ for all neighbors $v \in \mathfrak{S}$, decrements the HC^{max} field and sends the packet to the neighbor $v \in \mathfrak{S}$ with the maximum $\Delta_u(v)$.



(a) AODV (b) L2.5

Fig. 3. Instantaneous throughput with (-c) and without (-s) a node failure

4.1 Analysis of L2.5

The proposed Layer-2.5 forwarding paradigm aims at taking the link bandwidth availability resulting from a channel assignment into account in order to increase the network throughput. Also, L2.5 does not use routing tables in order to rapidly react to node/link failures. Indeed, if an intermediate node fails, L2.5 can rapidly avoid sending packets to the failed node by simply eliminating it from the set of candidate next-hop nodes. Recovering from a failure also does not require to exchange information among nodes, thus avoiding to generate overhead traffic. We compared L2.5 to AODV [11] (a destination-based routing protocol which uses routing tables) by evaluating their reaction to node failures. For this purpose, we conducted some simulations by using the *ns-2* network simulator. Figure 3 shows the results of one of such simulations. At a given time instant (time instant 14 in the simulation), a node failure is simulated. Figures 3a and 3b show the instantaneous throughput during the simulation time with and without the node failure achieved by AODV and L2.5, respectively. We can observe that, with respect to the case with no failure, the throughput decrease for AODV is of 0.3 Mb/s after 1 second from the failure and more than 1 Mb/s after 4 seconds. Thus, a destination-based routing protocol such as AODV takes a significant amount of time to recover from the node failure. L2.5 instead shows a decrease of 0.3 Mb/s in the throughput after 2 seconds but soon after recovers the average throughput achieved in the failure-free scenario.

However, the mechanism used by L2.5 to keep the utilization of network links close to their flow rate turns to be also one of its limits. Indeed, despite L2.5 ensures that packets reach the destination in at most *maximum hop count* hops (which is usually α times the length of the shortest path, as determined by the

source mesh router), it cannot avoid that packets take long path to the destination. Simulations have shown that most of the packets take exactly *maximum hop count* hops to the destination. Despite a countermeasure has been proposed in [2] (the introduction of a parameter *beta* which weighs $\Delta_u(v)$ depending on the ratio of the minimum hop count to the destination to the residual *maximum hop count*), such issue is not solved, but simply converted into a trade-off between accurate link utilization and limited path length. The basic problem is that L2.5 does not provide adequate control over the paths taken by packets and the underlying IP layer does not allow to differentiate among packets belonging to different flows. In the next section we show how the use of MPLS allows to better implement the idea behind L2.5 and thus avoid the drawbacks described so far.

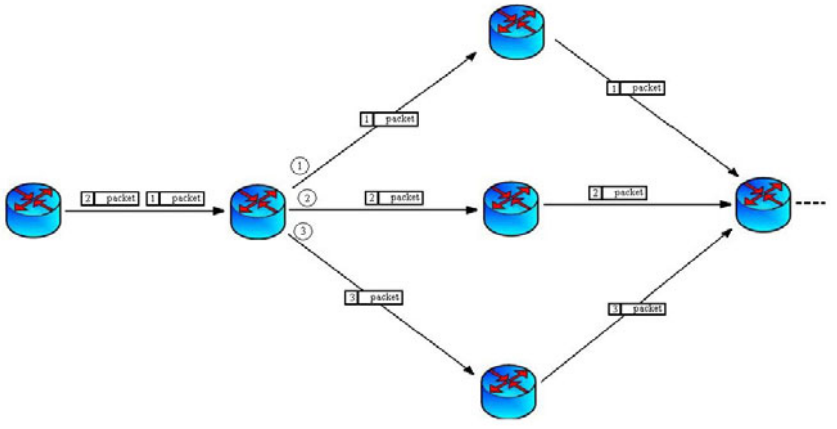


Fig. 4. Splitting in an MPLS network

5 Implementing L2.5 in an MPLS Network

Based on the above description of L2.5, it should be clear that MPLS cannot be used as is to support L2.5. Indeed, with L2.5 each node forwards packets to its neighbors based on the utilization of its links, regardless of the flow they belong to. Consequently, the packets of a flow follow multiple distinct paths. MPLS, instead, establishes tunnels and maps each flow onto a single tunnel. Hence, all the packets of a flow are routed along the corresponding tunnel. Thus, we would need the ability to split MPLS packets across multiple tunnels. Fortunately, we have already implemented a non-standard feature [12] (as a patch against the Linux kernel) that allows an MPLS router (there is a project [13] adding MPLS support to the Linux kernel) to split the packets of a flow across multiple tunnels (fig. 4). Different policies can be used at a node to select one of the multiple outgoing tunnels. The simplest one is clearly the round robin strategy, but advanced policies which take the current link load into account can be implemented. If all the network nodes are given such a splitting capability, then MPLS can be effectively used to support L2.5.

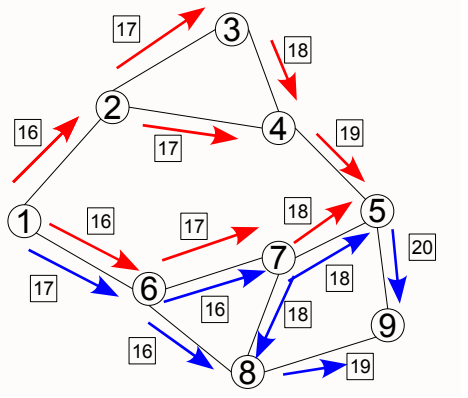


Fig. 5. Splitting in a wireless mesh network

Figure 5 shows how it is possible to implement L2.5 by using MPLS enhanced with the splitting feature. In particular, MPLS labels can be configured to force a given flow to follow a pre-determined set of paths. For instance, node 1 in fig. 5 can be configured so that it splits the packets destined to node 5 between node 2 and node 6 (red arrows). In both cases, the label pushed into those packets is 16. If labels are configured properly, the packets of the flow between nodes 1 and 5 follow all the possible paths marked with red arrows (there are actually three possible such paths). At node 1, instead, packets destined to node 9 are all sent to node 6 using label 17. Node 6 can know which flow each received packet belongs to by simply inspecting its MPLS label: packets of the flow 1→5 are labelled with 16, while those of the flow 1→9 are labelled with 17. Packets of the former flow are all sent to node 7, while packets of the latter flow are split between node 7 and 8. Thus, node 6 can differentiate the forwarding of packets based on the flow they belong to. This capability is simply not possible with plain IP and allows a strict control over the path taken by packets.

The use of MPLS to implement L2.5 brings several important advantages:

- MPLS is a standard, well known forwarding paradigm with a large base of installations
- MPLS provides strict control over the path followed by packets. It is thus possible to avoid that packets take long paths
- By using distinct labels for different flows, it is possible to differentiate the forwarding of packets belonging to different flows
- The MPLS header includes a TTL field, whose value can be used at an intermediate node to determine how many hops a packet has already travelled

The main drawback of L2.5 is the scarce control over the path taken by packets, which is solved by using MPLS. MPLS is thus a powerful instrument which gives us high flexibility in the management of the traffic crossing the network. Also, MPLS enables us to use standard mechanisms to provide some functionalities (e.g., the standard TTL field in the MPLS header can be used as a counter of

the hops taken by a packet). Clearly, in order to implement L2.5 by using MPLS in an efficient manner, it is needed:

- to properly engineer the possible paths taken by the packets of the flows between each source-destination pair
- to define a proper splitting policy, to be used at every node in order to keep the utilization of links close to the computed flow rates

In particular, a proper splitting policy should inspect the TTL field in the MPLS header of the received packet to determine the subset of outgoing tunnels that allow the packet to reach the destination in a *reasonable* number of hops. We assume that each node knows the maximum number of hops to reach the destination associated with each of the outgoing tunnels. Then, the outgoing tunnel can be selected by considering the link with the highest gap between the current utilization and the computed flow rate.

6 Conclusions and Future Work

Given the recent advances in wireless communications and the cost-effectiveness of deploying a multi-hop wireless network, we foresee the use of wireless mesh networks as the communication infrastructure of SCADA systems. A key role to provide the needed requirements in terms of performance, reliability and resiliency of SCADA systems is played by the routing protocol. In a previous work, we defined a forwarding paradigm for wireless mesh networks which was shown to be robust against node/link failures. However, such a forwarding paradigm cannot be used as-is in the case of SCADA systems due to its loose control over the network traffic. In this paper, we have proposed to use MPLS as the forwarding engine of our paradigm and described the resulting advantages. As future work, we plan to design efficient algorithm to compute the tunnels to be configured in the MPLS network and an efficient policy for the splitting mechanism.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement No. 225553 (INSPIRE Project).

References

1. Akyildiz, I.F., Wang, X., Wang, W.: Wireless mesh networks: a survey. Computer Networks 47(4), 445–487 (2005)
2. Avallone, S., Akyildiz, I.F., Ventre, G.: A Channel and Rate Assignment Algorithm and a Layer-2.5 Forwarding Paradigm for Multi-Radio Wireless Mesh Networks. IEEE/ACM Transactions on Networking 17(1), 267–280 (2009)
3. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. RFC 3031, IETF (January 2001)

4. Bel Air Networks, http://www.belairnetworks.com/products/
5. Proxim Wireless, http://www.proxim.com/
6. Raniwala, A., Gopalan, K., Chiueh, T.: Centralized Channel Assignment and Routing Algorithms for Multi-Channel Wireless Mesh Networks. ACM Mobile Computing and Communications Review 8(2), 50–65 (2004)
7. Alicherry, M., Bhatia, R., Li, E.: Joint Channel Assignment and Routing for Throughput Optimization in Multiradio Wireless Mesh Networks. IEEE Journal on Selected Areas in Communications 24(11), 1960–1971 (2006)
8. Kodialam, M., Nandagopal, T.: Characterizing the Capacity Region in Multi-Radio Multi-Channel Wireless Mesh Networks. In: Proc. of ACM MobiCom, pp. 73–87 (2005)
9. Mohsenian Rad, A.H., Wong, V.W.S.: Joint logical topology design, interface assignment, channel allocation, and routing for multi-channel wireless mesh networks. IEEE Transactions on Wireless Communications 6(12), 4432–4440 (2007)
10. Chen, Y.Y., Liu, S.C., Chen, C.: Channel assignment and routing for multi-channel wireless mesh networks using simulated annealing. In: Proc. of IEEE GLOBECOM, pp. 1–5 (November 2006)
11. Perkins, C., Belding-Royer, E., Das, S.: Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, IETF (July 2003)
12. Avallone, S., Manetti, V., Mariano, M., Romano, S.P.: A splitting infrastructure for load balancing and security in an MPLS network. In: Proceedings of TridentCom 2007, Orlando, FL, USA. IEEE, Los Alamitos (May 2007)
13. MPLS for Linux, http://mpls-linux.sourceforge.net