Self-Reconfigurable Wireless Mesh Networks

Kyu-Han Kim, *Member, IEEE* and Kang G. Shin, *Fellow, IEEE*

***Abstract*—During their lifetime, multi-hop wireless mesh networks (WMNs) experience frequent link failures caused by channel interference, dynamic obstacles and/or applications' bandwidth demands. These failures cause severe performance degradation in WMNs or require expensive, manual network management for their real-time recovery. This paper presents an *Autonomous network Reconfiguration System* (ARS) that enables a multi-radio WMN to autonomously recover from local link failures to preserve network performance. By using channel and radio diversities in WMNs, ARS generates necessary changes in local radio and channel assignments in order to recover from failures. Next, based on the thus-generated configuration changes, the system cooperatively reconfigures network settings among *local* mesh routers. ARS has been implemented and evaluated extensively on our IEEE 802.11-based WMN test-bed as well as through *ns-2*-based simulation. Our evaluation results show that ARS outperforms existing failure-recovery schemes in improving channel-efficiency by more than 90% and in the ability of meeting the applications' bandwidth demands by an average of 200%.**

***Index Terms*—Multi-radio WMNs, wireless link failures, self-reconfigurable networks, IEEE 802.11.**

I. Introduction

Wireless mesh networks (WMNs) are being developed actively and deployed widely for a variety of applications, such as public safety, environment monitoring, and city-wide wireless Internet services [1]–[3]. They have also been evolving in various forms (e.g., using multi-radio/channel systems [4]–[7]) to meet the increasing capacity demands by the above-mentioned and other emerging applications. However, due to heterogeneous and fluctuating wireless link conditions [8]–[10], preserving the required performance of such WMNs is still a challenging problem. For example, some links of a WMN may experience significant channel interference from other co-existing wireless networks. Some parts of networks might not be able to meet increasing bandwidth demands from new mobile users and applications. Links in a certain area (e.g., a hospital or police station) might not be able to use some frequency channels because of spectrum etiquette or regulation [11].

Even though many solutions for WMNs to recover from wireless link failures have been proposed, they still have several limitations as follows. First, resource-allocation algorithms [12]–[14] can provide (theoretical) guidelines for initial network resource planning. However, even though their approach provides a comprehensive and optimal network configuration plan, they often require "global" configuration changes, which are undesirable in case of frequent local link failures. Next, a *greedy* channel-assignment algorithm (e.g., [15]) can reduce the requirement of network changes by changing settings of only the faulty link(s). However, this greedy change might not be able to realize full improvements, which can only be achieved by considering configurations of neighboring mesh routers in addition to the faulty link(s). Third, fault-tolerant routing protocols, such as local re-routing [16] or multi-path routing [17], can be adopted to use network-level path diversity for avoiding the faulty links. However, they rely on detour paths or redundant transmissions, which may require more network resources than link-level network reconfiguration.

To overcome the above limitations, we propose an *Autonomous network Reconfiguration System* (ARS) that allows a multi-radio WMN (mr-WMN) to autonomously reconfigure its local network settings—channel, radio, and route assignment—for real-time recovery from link failures. In its core, ARS is equipped with a reconfiguration planning algorithm that identifies local configuration changes for the recovery, while minimizing changes of healthy network settings. Briefly, ARS first searches for feasible local configuration changes available around a faulty area, based on current channel and radio associations. Then, by imposing current network settings as constraints, ARS identifies reconfiguration plans that require the minimum number of changes for the healthy network settings.

Next, ARS also includes a monitoring protocol that enables a WMN to perform real-time failure recovery in conjunction with the planning algorithm. The accurate link-quality information from the monitoring protocol is used to identify network changes that satisfy applications' new QoS demands or that avoid propagation of QoS failures to neighboring links (or 'ripple effects'). Running in every mesh node, the monitoring protocol periodically measures wireless link-conditions via a hybrid link-quality measurement technique, as we will explain in Section IV. Based on the measurement information, ARS detects link failures and/or generates QoS-aware network reconfiguration plans upon detection of a link failure.

ARS has been implemented and evaluated extensively via experimentation on our multi-radio WMN test-bed as well as via ns2-based simulation. Our evaluation results show that ARS outperforms existing failure-recovery methods, such as static or greedy channel assignments, and local re-routing. First, ARS's planning algorithm effectively identifies reconfiguration plans that maximally satisfy the applications' QoS demands, accommodating twice more flows than static assignment. Next, ARS avoids the ripple effect via QoS-aware reconfiguration planning, unlike the greedy approach. Third, ARS's local reconfiguration improves network throughput and channel-efficiency by more than 26% and 92%, respectively,

A 4-page summary of an earlier version of this work was presented in ACM MobiCom'07.

Kyu-Han Kim is currently with Deutsche Telekom Inc., R&D Lab in Los Altos, CA 94022, U.S.A. Kang G. Shin is with the Real-Time Computing Laboratory in the Department of Electrical Engineering and Computer Science at The University of Michigan, Ann Arbor, MI 48109, U.S.A. E-mail: kyu-han.kim@telekom.com and kgshin@eecs.umich.edu

over the local re-routing scheme.

The rest of this paper is organized as follows. Section II describes the motivation behind this work. Section III provides the design rationale and algorithms of ARS. Section IV describes the implementation and experimentation results on ARS. Section V shows in-depth simulation results of ARS. Section VI concludes the paper.

II. Motivation

We first describe the need for self-reconfigurable multi-radio WMNs (mr-WMNs). Next, we introduce the network model and assumptions to be used in this paper. Finally, we discuss the limitations of existing approaches to achieving self-reconfigurability of mr-WMNs.

A. Why is Self-Reconfigurability Necessary?

Maintaining the performance of WMNs in the face of dynamic link failures remains a challenging problem [18]. However, such failures can be withstood (hence maintaining the required performance) by enabling mr-WMNs to autonomously reconfigure channels and radio[1] assignments, as in the following examples.

- *Recovering from link-quality degradation*: The quality of wireless links in WMNs can degrade (i.e., *link-quality failure*), due to severe interference from other co-located wireless networks [8], [19]. For example, Bluetooth, cordless phones, and other co-existing wireless networks operating on the same or adjacent channels cause significant and varying degrees of losses or collisions in packet transmissions, as shown in Fig. 1. By switching the tuned channel of a link to other interference-free channels, local links can recover from such a link failure.
- *Satisfying dynamic QoS demands*: Links in some areas may not be able to accommodate increasing QoS demands from end users (*QoS failures*),[2] depending on spatial or temporal locality [20]. For example, links around a conference room may have to relay too much data/video traffic during the session. Likewise, relay links outside the room may fail to support all attendees' Voice-over-IP calls during a session break. By re-associating their radios/channels with under-utilized radios/channels available nearby, links can avoid communication failures.
- *Coping with heterogeneous channel availability*: Links in some areas may not be able to access wireless channels during a certain time period (*spectrum failures*), due to spectrum etiquette or regulation [11], [21]. For example, some links in a WMN need to vacate current channels if channels are being used for emergency response near the wireless links (e.g., hospital, public safety). Such links can seek and identify alternative channels available in the same area.

Motivated by these three and other possible benefits of using reconfigurable mr-WMNs, in the remainder of this paper, we would like to develop a system that allows mr-WMNs to autonomously change channel and radio assignments (i.e., *self-reconfigurable*) to recover from the channel-related link failures mentioned above.

[1]The terms "radio" and "interface" are used interchangeably in this paper.

[2]We consider link bandwidth as a QoS parameter of interest.

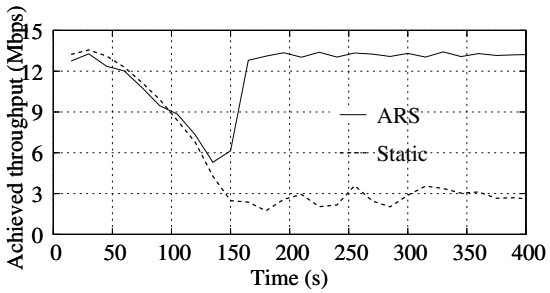


Fig. 1. *The effect of wireless link failures and the need for network reconfiguration:* A wireless link in WMNs often experiences link-quality degradation due to interference from co-existing other networks. To recover from such a link failure, the network needs (at 130 s) to switch to non-interfering channel in real time. Link-quality measurements are made using IEEE 802.11a NIC in our testbed.

B. Network Model and Assumptions

Multi-radio WMN: A network is assumed to consist of mesh nodes, IEEE 802.11-based wireless links, and one control gateway. Each mesh node is equipped with n radios, and each radio's channel and link assignments are initially made (e.g., see Fig. 2) by using global channel/link assignment algorithms [5], [12], [13]. Multiple orthogonal channels are assumed available. For example, an IEEE 802.11a/b/g combo PCMCIA card can tune 16 orthogonal channels. The interference among multiple radios in one node is assumed to be negligible via physical separation among antennas or by using shields. The gateway is connected to the Internet via wire-line links as well as to other mesh routers via wireless links.

QoS support: During its operation, each mesh node periodically sends its local channel usage and the quality information for all outgoing links via management messages to the control gateway. Then, based on this information, the gateway controls the admission of requests for voice or video flows. For admitted flows, the information about QoS requirements is delivered to the corresponding nodes for resource reservation through the RSVP protocol [22]. Next, the network runs routing protocols such as WCETT [6] or ETX [23] to determine the path of the admitted flows. This routing protocol is also assumed to include route discovery and recovery algorithms [16], [24], [25] that can be used for maintaining alternative paths even in the presence of link failures.

Link failures: Channel-related link failures that we focus on, are due mainly to narrow-band channel failures.[3] These failures are assumed to occur and last in the order of a few minutes to hours, and reconfiguration is triggered in the same order of failure occurrences. For short-term (lasting for milliseconds) failures, fine-grained (e.g., packet-level or in the order of milliseconds) dynamic resource allocation might be

[3]Thus, we assume that simple transmission rate adaptation algorithms (e.g., [26]) are not applicable to failure recovery.

sufficient [4], [7], and for a long-term (lasting for weeks or months) failures, network-wide planning algorithms [12]–[14] can be used. Note that hardware failures (e.g., node crashes) or broadband-channel failures (e.g., jamming) are beyond the scope of this paper.

C. Limitations of Existing Approaches

Given the above system models, we now discuss the pros and cons of using existing approaches for self-reconfigurable WMNs.

Localized reconfiguration: Network reconfiguration needs a planning algorithm that keeps necessary network changes (to recover from link failures) as local as possible, as opposed to changing the entire network settings. Existing channel assignment and scheduling algorithms [12]–[14] provide holistic guidelines such as throughput bounds and schedulability for channel assignment during a network deployment stage. However, the algorithms do not consider the degree of configuration changes from previous network settings, and hence they often require *global* network changes to meet all the constraints, akin to edge coloring problems [27]. Even though these algorithms are suitable for static or periodic network planning, they may cause network service disruption and thus are unsuitable for dynamic network reconfiguration that has to deal with frequent local link failures.

Next, the *greedy* channel-assignment algorithm, which considers only local areas in channel assignments (e.g., [15]), might do better in reducing the scope of network changes than the above-mentioned assignment algorithms. However, this approach still suffers from the ripple effect, in which one local change triggers the change of additional network settings at neighboring nodes (e.g., nodes using channel 3 in Fig. 2), due to association dependency among neighboring radios. This undesired effect might be avoided by transforming a mesh topology into a tree topology, but this transformation reduces network connectivity as well as path diversity among mesh nodes.

Finally, interference-aware channel-assignment algorithms [5], [28] can minimize interference by assigning orthogonal channels as closely as possible geographically. While this approach can improve overall network capacity by using additional channels, the algorithm could further improve its flexibility by considering both radio diversity (i.e., link association) and local traffic information. For example, in Fig. 2, if channel 5 is lightly-loaded in a faulty area, the second radio of node *C* can re-associate itself with the first radio of node *I*, avoiding configuration changes of other links.

QoS-awareness: Reconfiguration has to satisfy QoS constraints on each link as much as possible. First, given each link's bandwidth constraints, existing channel-assignment and scheduling algorithms [5], [12], [13] can provide approximately optimal network configurations. However, as pointed out earlier, these algorithms may require global network configuration changes from changing local QoS demands, thus causing network disruptions. We need instead a reconfiguration algorithm that incurs only local changes while maximizing the chance of meeting the QoS demands. For example, if link EH in Fig. 2 experiences a QoS failure on channel 1, then one simple reconfiguration plan would be to re-associate R1 of node H to R2 of node E in channel 5, which has enough bandwidth.

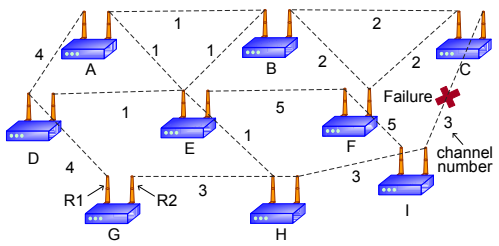


Fig. 2. *Multi-radio WMN*: A WMN has an initial assignment of frequency channels as shown above. The network often experiences wireless link failure and needs to reconfigure its settings.

Next, the greedy algorithm might be able to satisfy particular links' QoS demands by replacing a faulty channel with a new channel. However, neighboring links, whose channel has been changed due to ripple effects (e.g., links GH and HI in Fig. 2), may fail to meet QoS demands if the links in the new channel experience interference from other co-existing networks that operate in the same channel.

Cross-layer interaction: Network reconfiguration has to jointly consider network settings across multiple layers. In the network layer, fault-tolerant routing protocols, such as local re-routing [16] or multi-path routing [17], allow for flow reconfiguration to meet the QoS constraints by exploiting path diversity. However, they consume more network resources than link reconfiguration, because of their reliance on detour paths or redundant transmissions. On the other hand, channel and link assignments across the network and link layers can avoid the overhead of detouring, but they have to take interference into account to avoid additional QoS failures of neighboring nodes.

III. The ARS Architecture

We first present the design rationale and overall algorithm of ARS. Then, we detail ARS's reconfiguration algorithms. Finally, we discuss the complexity of ARS.

A. Overview

ARS is a distributed system that is easily deployable in IEEE802.11-based mr-WMNs. Running in every mesh node, ARS supports self-reconfigurability via the following distinct features:

- *Localized reconfiguration*: Based on multiple channels and radio associations available, ARS generates reconfiguration plans that allow for changes of network configurations only in the vicinity where link failures occurred, while retaining configurations in areas remote from failure locations.
- *QoS-aware planning*: ARS effectively identifies QoS-satisfiable reconfiguration plans by (i) estimating the QoS-satisfiability of generated reconfiguration plans and (ii) deriving their expected benefits in channel utilization.

Algorithm 1 ARS Operation at mesh node i

(1) Monitoring period (t_m)
1: **for every** link j **do**
2: measure link-quality (lq) using passive monitoring;
3: **end for**
4: send monitoring results to a gateway g;
(2) Failure detection and group formation period (t_f)
5: **if** link l violates link requirements r **then**
6: request a group formation on channel c of link l;
7: **end if**
8: participate in a leader election if a request is received;
(3) Planning period (M, t_p)
9: **if** node i is elected as a leader **then**
10: send a planning request message (c, M) to a gateway;
11: **else if** node i is a gateway **then**
12: synchronize requests from reconfiguration groups M_n
13: generate a reconfiguration plan (p) for M_i;
14: send a reconfiguration plan p to a leader of M_i;
15: **end if**
(4) Reconfiguration period (p, t_r)
16: **if** p includes changes of node i **then**
17: apply the changes to links at t;
18: **end if**
19: relay p to neighboring members, if any

- *Autonomous reconfiguration via link-quality monitoring*: ARS accurately monitors the quality[4] of links of each node in a distributed manner. Furthermore, based on the measurements and given links' QoS constraints, ARS detects local link failures and autonomously initiates network reconfiguration.
- *Cross-layer interaction*: ARS actively interacts across the network and link layers for planning. This interaction enables ARS to include a re-routing for reconfiguration planning in addition to link-layer reconfiguration. ARS can also maintain connectivity during recovery period with the help of a routing protocol.

Algorithm 1 describes the operation of ARS. First, ARS in every mesh node monitors the quality of its outgoing wireless links at every t_m (e.g., 10 sec) and reports the results to a gateway via a management message. Second, once it detects a link failure(s), ARS in the detector node(s) triggers the formation of a group among local mesh routers that use a faulty channel, and one of the group members is elected as a leader using the well-known bully algorithm [29], for coordinating the reconfiguration. Third, the leader node sends a planning-request message to a gateway. Then, the gateway synchronizes the planning requests—if there are multiples requests—and generates a reconfiguration plan for the request. Fourth, the gateway sends a reconfiguration plan to the leader node and the group members. Finally, all nodes in the group execute the corresponding configuration changes, if any, and resolve the group. We assume that during the formation and reconfiguration, all messages are reliably delivered via a routing protocol and per-hop retransmission timer.

[4]For example, the quality parameters include the packet-delivery ratio or data-transmission rate as discussed in Section III-B.

In what follows, we will detail each of these operations, including how to generate reconfiguration plans, how to monitor link conditions such as bandwidth (Section III-B), and how much overhead ARS generates for the monitoring and for maintaining a reconfiguration group (Section III-C).

B. Planning for Localized Network Reconfiguration

The core function of ARS is to *systematically* generate localized reconfiguration plans. A *reconfiguration plan* is defined as a set of links' configuration changes (e.g., channel switch, link association) necessary for a network to recover from a link(s) failure on a channel and there are usually multiple reconfiguration plans for each link failure. Existing channel-assignment and scheduling algorithms [5], [12], [13] seek 'optimal' solutions by considering tight QoS constraints on all links, thus requiring a large configuration space to be searched, and hence making the planning often an NP-complete problem [5]. In addition, change in a link's requirement may lead to completely different network configurations. By contrast, ARS systematically generates reconfiguration plans that localize network changes by dividing the reconfiguration planning into three processes—feasibility, QoS-satisfiability, and optimality—and applying different levels of constraints. As depicted in Figure 3, ARS first applies connectivity constraints to generate a *set* of feasible reconfiguration plans that enumerate feasible channel, link, and route changes around the faulty areas, given connectivity and link-failure constraints. Then, within the set, ARS applies strict constraints (i.e., QoS and network utilization) to identify a reconfiguration plan that satisfies the QoS demands and that improves network utilization most.

Feasible plan generation: Generating feasible plans is essentially to search all legitimate changes in links' configurations and their combinations, around the faulty area. Given multiple radios, channels, and routes, ARS identifies feasible changes that help avoid a local link failure but maintain existing network connectivity as much as possible. However, in generating such plans, ARS has to address the following challenges.

- *Avoiding a faulty channel*: ARS first has to ensure that the faulty link needs to be fixed via reconfiguration. To this end, ARS considers three primitive link changes, as explained in Table I. Specifically, to fix a faulty link(s), ARS can use (i) a channel-switch (S) where both end-radios of link AB can simultaneously change their tuned channel, (ii) a radio-switch (R) where one radio in node A can switch its channel and associate with another radio in node B, and (iii) a route-switch (D) where all traffic over the faulty link can use a detour path, instead of the faulty link.
- *Maintaining network connectivity and utilization*: While avoiding the use of the faulty channel, ARS needs to maintain connectivity with the full utilization of radio resources. Because each radio can associate itself with multiple neighboring nodes, a change in one link triggers other neighboring links to change their settings. To coordinate such propagation, ARS takes a *two-step* approach. ARS first generates feasible changes of each link using the primitives,

Fig. 3. *Localized reconfiguration planning in ARS*: ARS generates a reconfiguration plan by breaking down the planning process into three processes with different constraints.

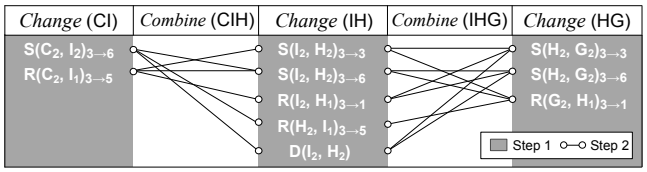


Fig. 4. *Example of network planning*: ARS generates per-link changes (blue columns) and then connects them for feasible reconfiguration plans (white columns) for recovery of the failure in Fig. 2.

and then combines a set of feasible changes that enable a network to maintain its own connectivity. Furthermore, for the combination, ARS maximizes the usage of network resources by making each radio of a mesh node associate itself with at least one link and by avoiding the use of same (*redundant*) channel among radios in one node.

- *Controlling the scope of reconfiguration changes*: ARS has to limit network changes as *local* as possible, but at the same time it needs to find a locally optimal solution by considering more network changes or scope. To make this tradeoff, ARS uses a k-hop reconfiguration parameter. Starting from a faulty link(s), ARS considers link changes within the first k hops and generates feasible plans. If ARS cannot find a local solution, it increases the number of hops (k) so that ARS may explore a broad range of link changes. Thus, the total number of reconfiguration changes is determined on the basis of existing configurations around the faulty area as well as the value of k.

Let us consider an illustrative example in Figure 4. Given the failure in link CI, ARS first generates feasible and desirable changes per link (gray columns) using the primitives. Here, the changes must not include the use of a faulty or redundant channel. Next, ARS combines the generated per-link primitives of neighboring links to generate a set of feasible plans. During the combination, ARS has to preserve link and/or radio connectivities. For example, plans $\text{S(C,I)}_{3\rightarrow 6}$ and $\text{S(H,I)}_{3\rightarrow 3}$ in Fig. 4 cannot be connected because each change requires the same radio of node *I* to set up different channels. After the two steps, ARS has 11 feasible reconfiguration plans ($\mathbb{F}$) by traversing connected changes of all links considered in the planning. Note that we set k to 2 in this example, but we will show the impact of k on the planning in Section V-B3.

QoS-Satisfiability Evaluation: Among a set of feasible plans $\mathbb{F}$, ARS now needs to identify QoS-satisfying reconfiguration plans by checking if the QoS constraints are met under each plan. Although each feasible plan ensures that a faulty link(s) will use non-faulty channels and maintain its connectivity, some plans might not satisfy the QoS constraints or even cause cascaded QoS failures on neighboring links. To filter out such plans, ARS has to solve the following challenges.

TABLE I
Definition of link-change in ARS. EACH CHANGE REPRESENTS A PRIMITIVE LINK CHANGE IN CHANNEL, ASSOCIATION, OR ROUTE. MULTIPLE CHANGES CAN BE JOINTLY USED TO REPRESENT CHANGES OF MULTIPLE LINKS.

Primitive changes	Description
Channel switch $(S(A_i, B_j)_{\alpha\rightarrow\beta})$	Radios A_i and B_j of link AB switch their channel (α) to other channel (β).
Radio switch $(R(A_i, B_j)_{\alpha\rightarrow\beta})$	Radio A_i in node *A* re-associates with radio B_j in node *B*, tuned in channel (β).
Detouring $(D(A_i, B_j))$	Both radios A_i and B_j of link *AB* remove their associations and use a detour path, if exists.

- *Per-link Bandwidth Estimation*: For each feasible plan, ARS has to check whether each link's configuration change satisfies its bandwidth requirement, so it must estimate link bandwidth. To estimate link bandwidth, ARS accurately measures each link's capacity and its available channel air-time. In multi-hop wireless networks equipped with a CSMA-like MAC, each link's achievable bandwidth (or throughput) can be affected by both link capacity and activities of other links that share the channel air-time. Even though numerous bandwidth-estimation techniques have been proposed, they focus on the average bandwidth of each node in a network [23], [30] or the end-to-end throughput of flows [17], which cannot be used to calculate the impact of per-link configuration changes. By contrast, ARS estimates an individual link's capacity (C), based on measured (or cached) link-quality information—packet-delivery ratio and data-transmission rate measured by passively monitoring the transmissions of data or probing packets [31]—and the formula derived in Appendix A. Here, we assume that ARS is assumed to cache link-quality information for other channels and use the cached information to generate reconfiguration plans. If the information becomes obsolete, ARS detects link failures and triggers another reconfiguration to find QoS-satisfiable plans—lazy monitoring.
- *Examining per-link bandwidth satisfiability*: Given measured bandwidth and bandwidth requirements, ARS has to check if the new link change(s) satisfies QoS requirements. ARS defines and uses the expected busy air-time ratio of each link to check the link's QoS satisfiability. Assuming that a link's bandwidth requirement (q) is given, the link's busy air-time ratio (BAR) can be defined as BAR = $\frac{q}{C}$ and must not exceed 1.0 (i.e., BAR $<$ 1.0) for a link to satisfy its bandwidth requirement. If multiple links share the air-time of one channel, ARS calculates aggregate BAR (*aBAR*) of end-radios of a link, which is defined as $aBAR(k) = \sum_{l\in L(k)} \frac{q_l}{C_l}$, where k is a radio ID, l a link associated with radio k, $L(k)$ the set of directed links within and across radio k's transmission range.
- *Avoiding cascaded link failures*: Besides the link change, ARS needs to check whether neighboring links are affected by local changes (i.e., cascaded link failures). To identify

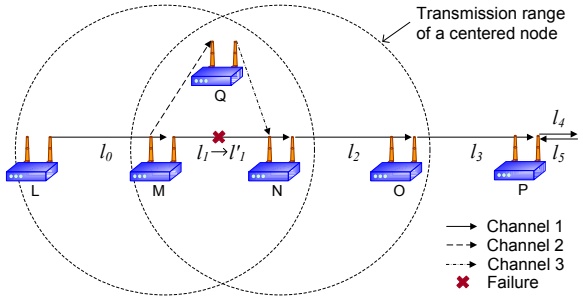


Fig. 5. *Busy Air-time Ratio (BAR) of a directed link*: BAR (e.g., l_1) is affected by activities of neighboring links (l_0, l_2,and l_3) in channel 1 and is used to evaluate QoS satisfiability of a link.

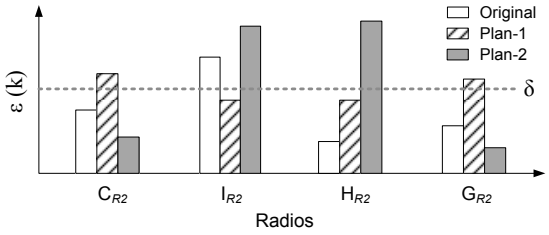


Fig. 6. *Benefit function*: B prefers a reconfiguration plan that improves overall channel utilization close to the desired parameter δ.

such adverse effect from a plan, ARS also estimates the QoS-satisfiability of links one hop away from member nodes whose links' capacity can be affected by the plan. If these one-hop-away links still meet the QoS requirement, the effects of the changes do not propagate thanks to spatial reuse of channels. Otherwise, the effects of local changes will propagate, causing cascaded QoS failures.

Let us consider an example in Fig. 5. Assuming BAR of each directed link (l_i) is 0.2 (e.g., $\frac{2Mbps}{10Mbps}$) in a tuned channel, *aBAR* of each radio tuned to channel 1 does not exceed 1.0, satisfying each link's QoS requirement. In addition, assuming that BAR(l_1) increases from 0.2 to 0.4 (l'_1) in Fig. 5. To accommodate this increase, reconfiguration plans that have a detour path through node Q do not affect the QoS-satisfiability of the neighboring nodes. On the other hand, plans with radio switches (e.g., R(L_2,M_1)$_{1\rightarrow 2}$) satisfy the QoS of link MN but cause *aBAR(O_{R2})* to exceed 1.0, resulting in cascaded QoS failures of links beyond node O.

Choosing the best plan: ARS now has a set of reconfiguration plans that are QoS-satisfiable, and needs to choose a plan within the set for a local network to have evenly distributed link capacity. However, to incorporate the notion of fair share into the planning, ARS needs to address the following challenges.

- *Quantifying the fairness of a plan*: ARS has to quantify the potential changes in link-capacity distribution from a plan. To this end, ARS defines and uses a benefit function $B(p)$ that quantifies the improvement of channel utilization that the reconfiguration plan p makes. Specifically, the benefit function is defined as $B(p) = \frac{1}{n}\sum_{k=1}^{n}\beta(k)$, where $\beta(k)$ is the relative improvement in the air-time usage of radio k, and n the number of radios whose $\beta(k)$ has changed from the plan. This definition allows the benefit function to quantify the overall change in air-time usage, resulting from the reconfiguration plan. Here, $\beta(k)$ is considered as a fairness index on the usage of channel air-time, and it is defined as follows:

$$\beta(k) = \begin{cases} \epsilon_1(k) - \epsilon_2(k) & \text{if } \epsilon_1(k), \epsilon_2(k) > \delta \\ \epsilon_2(k) - \epsilon_1(k) & \text{if } \epsilon_1(k), \epsilon_2(k) < \delta \\ \epsilon_1(k) + \epsilon_2(k) - 2\delta & \text{if } \epsilon_1(k) > \delta > \epsilon_2(k) \\ 2\delta - \epsilon_1(k) - \epsilon_2(k) & \text{if } \epsilon_2(k) > \delta > \epsilon_1(k) \end{cases} \quad (1)$$

where $\epsilon_1(k)$ and $\epsilon_2(k)$ are estimated *aBAR*s of a radio k in existing configurations and in new configurations, respectively, and δ the desired channel utilization.[5]

- *Breaking a tie among multiple plans*: Multiple reconfiguration plans can have the same benefit, and ARS needs to break a tie among them. ARS uses the number of link changes that each plan requires to break a tie. Although link configuration changes incur a small amount of flow disruption (e.g., in the order of 10 ms), the less changes in link configuration, the less network disruption.

Suppose δ is 0.5 as shown in Fig. 6; ARS favors a plan that reconfigures links to have 50% available channel air-time (e.g., plan 1 in the figure). If a plan reconfigures a WMN to make the links heavily utilized while idling others (e.g., plan 2), then the benefit function considers the plan ineffective, placing the plan in a lowly-ranked position. The effectiveness of β and δ will be evaluated and discussed further in Section V-B2. Eq. (1) implies that if a reconfiguration plan makes overall links' channel utilization closer to the desired utilization δ, then $\beta(k)$ gives a positive value, while giving a negative value otherwise.

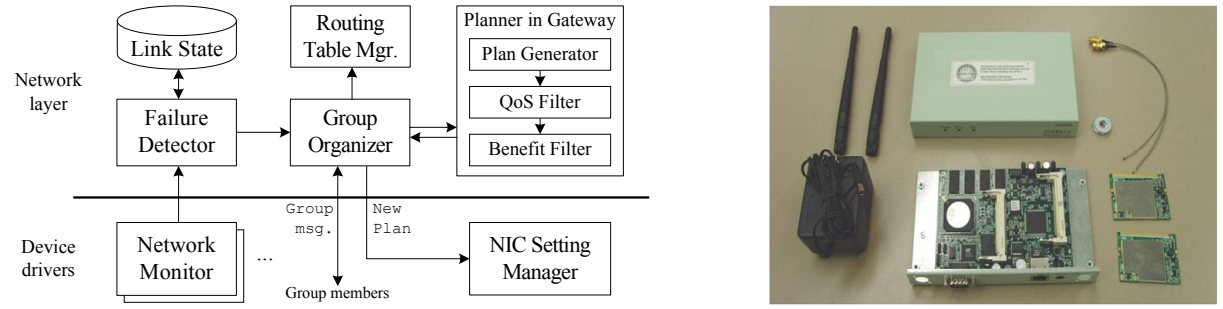
C. Complexity of ARS

Thanks to its distributed and localized design, ARS incurs reasonable bandwidth and computation overheads. First, the network monitoring part in the reconfiguration protocols is made highly efficient by exploiting existing data traffic and consumes less than 12 Kbps probing bandwidth (i.e., 1 packet per second) for each radio. In addition, the group formation requires only $O(n)$ message overhead (in forming a spanning tree), where n is the number of nodes in the group. Next, the computational overhead in ARS mainly stems from the planning algorithms. Specifically, generating its possible link plans incurs $O(n+m)$ complexity, where n is the number of available channels and m the number of radios. Next, a gateway node needs to generate and evaluate feasible plans, which incurs search overhead in a constraint graph that consists of $O(l(n+m))$ nodes, where l is the number of links that use a faulty channel in the group.

IV. System Implementation and Experimentation

We have implemented ARS in a Linux OS and evaluated it in our testbed. We first describe the implementation details,

[5]Note that δ is a system parameter set by network administrators, depending on the application scenarios of a WMN.



(a) ARS software architecture in each node

(b) ARS hardware prototype

Fig. 7. *ARS's implementation and prototype*: (a) ARS is implemented across network and link layers as a loadable module of Linux 2.6 kernel. (b) ARS software is then installed in Soekris wireless routers and evaluated extensively in our multi-radio WMN testbed.

and then present important experimental results on ARS.

A. Implementation Details

Fig. 7(a) shows the software architecture of ARS. First, ARS in the network layer is implemented using netfilter [32], which provides ARS with a hook to capture and send ARS-related packets such as group-formation messages. In addition, this module includes several important algorithms and protocols of ARS: (i) *network planner*, which generates reconfiguration plans only in a gateway node; (ii) *group organizer*, which forms a local group among mesh routers; (iii) *failure detector*, which periodically interacts with a network monitor in the device driver and maintains an up-to-date link-state table; and (iv) *routing table manager*, through which ARS obtains or updates states of a system routing table.

Next, ARS components in the device driver are implemented in an open source MADWiFi device driver [33]. This driver is designed for Atheros chipset-based 802.11 NICs [34] and allows for accessing various control and management registers (e.g., `longretry`, `txrate`) in the MAC layer, making network monitoring accurate. The module in this driver includes (i) *network monitor*, which efficiently monitors link-quality and is extensible to support as many multiple radios as possible [31]; and (ii) *NIC manager*, which effectively reconfigures NIC's settings based on a reconfiguration plan from the group organizer.

B. Experimental Setup

To evaluate our implementation, we constructed a multi-hop wireless mesh network testbed on the fourth floor of the Computer Science and Engineering (CSE) building at the University of Michigan. The testbed consists of 17 mesh nodes and has multiple (up to 5) links. Each node is deliberately placed on either ceiling panels or high-level shelves to send/receive strong signals to/from neighboring nodes. On the other hand, each node will experience enough multi-path fading effects from obstacles and interference from co-existing public wireless networks.

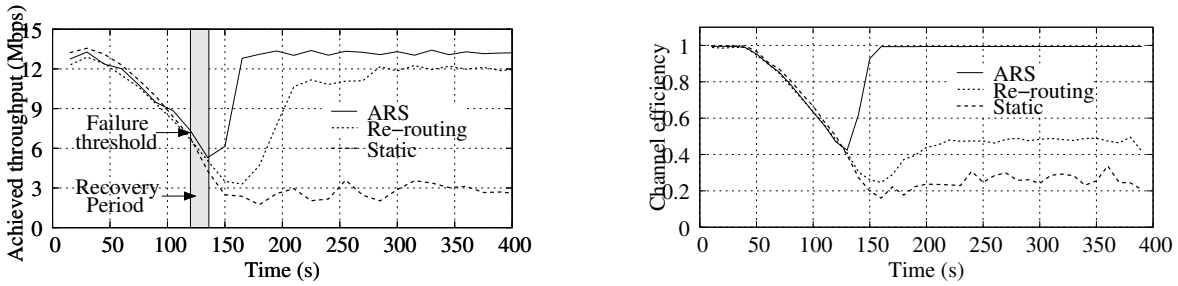
As shown in Fig. 7(b), each mesh node is a small-size wireless router—Soekris board 4826-50 [35] (Pentium-III 266 Mhz CPU, 128 MB memory). This router is equipped with two EMP IEEE 802.11 a/b/g miniPCI cards and 5-dBi gain indoor omni-directional antennae. Each card operates at IEEE 802.11a frequency with a pseudo ad-hoc mode, and is set to use fixed data rate and transmission power. Next, all nodes run the Linux OS (kernel-2.6), a MADWiFi device driver (version 0.9.2) for wireless interfaces, and the ARS implementation. In addition, ETX [23] and WCETT [6] routing metrics are implemented for routing protocols. Finally, the Iperf measurement tool [36] is used for measuring end-to-end throughput, and the numbers are derived by averaging the experimental results of 10 runs, unless otherwise specified.

C. Experimental Results

We evaluated the improvements achieved by ARS, including throughput and channel efficiency, QoS satisfiability, and reduction of ripple effects.

1) Throughput and channel-efficiency gains: We first study throughput and channel-efficiency gains via ARS's real-time reconfiguration. We run one UDP flow at a maximum rate over a randomly-chosen link in our testbed, while increasing the level of interference every 10 seconds. We also set the QoS requirement of every link to 6 Mbps, and measure the flow's throughput progression every 10 seconds during a 400-second run. For the purpose of comparison, we also ran the same scenario under the local re-routing with a WCETT (Weighted Cumulative Expected Transmission Time) metric [37], and static channel-assignment algorithms. Note that we do not intentionally run a greedy algorithm in this single-hop scenario, because its effect is subsumed by ARS. We, however, compare it with ARS in multi-hop scenarios in Section IV-C3.

Fig. 8(a) compares the progression of link throughput achieved by the above three methods. ARS effectively reconfigures the network on detection of a failure, achieving 450% and 25.6% more bandwidth than static-assignment and local re-routing, respectively. ARS accurately detects a link's QoS-failure using link-quality monitoring information, and completes network reconfiguration (i.e., channel switching) within 15 seconds on average, while the static-assignment experiences severe throughput degradation. Note that the 15-second delay is due mainly to link-quality information update and communication delay with a gateway, and the delay can be adjusted. Further, within the delay, actual channel switch delay is less than 3 ms, which causes negligible flow disruption. On the other hand, the local re-routing improves the throughput



(a) Throughput gains

(b) Channel-efficiency gains

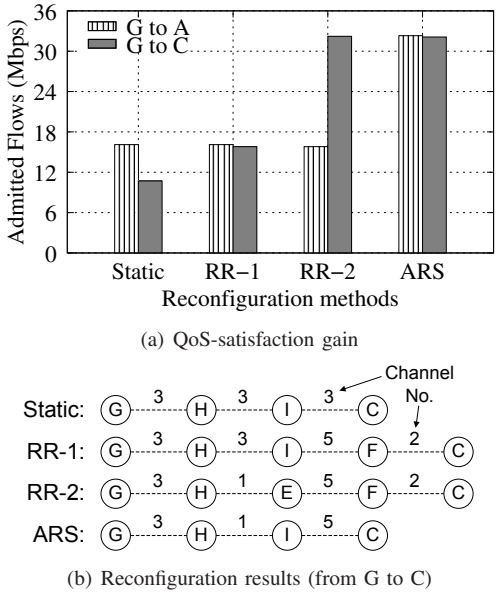
Fig. 8. *Gains in throughput and channel efficiency:* ARS effectively reconfigures the network around a faulty link, improving both network throughput and channel-efficiency by up to 26% and 92%, respectively. By contrast, local re-routing causes degradation in channel-efficiency due to the use of a detour path, and static channel-assignment does not react to faults in a timely manner.

by using a detour path, but still suffers from throughput degradation because of an increased loss rate along its detour path.

ARS also improves channel efficiency (i.e., the ratio of the number of successfully-delivered data packets to the number of total MAC frame transmissions) by more than 90% over the other recovery methods. Using the data collected during the previous experiment, we derive channel efficiency of the UDP flow by counting the number of total MAC frame transmissions and the number of successful transmissions. As shown in Fig. 8(b), ARS improves channel efficiency by up to 91.5% over the local re-routing scheme, thanks to its on-line channel reconfiguration. On the other hand, using static-channel assignment suffers poor channel utilization due to frame retransmissions on the faulty channel. Similarly, the local re-routing often makes traffic routed over longer or low link-quality paths, thus consuming more channel resources than ARS.

2) QoS-satisfaction gain: ARS enhances chance to meet the varying QoS demands. To show this gain, we first assign links and channels in our testbed as shown in Fig. 2. Here, nodes G, A, and C are a gateway, a mesh router in a conference room, and a mesh router in an office, respectively. We assume that mobile clients in the conference room request video streams through the router A during a meeting, and after the meeting, they return to the office room and connect to the router C. While increasing the number of video streams, we measure the total number of admitted streams after network reconfiguration for each place. We use static, WCETT routing metric that finds a path with diverse channels and ARS for reconfiguration.

QoS-aware reconfiguration planning in ARS improves the chance for a WMN to meet the varying QoS demands, on average, by 200%. As shown in Fig. 9(a), a static channel-assignment algorithm cannot support more bandwidth than the initial assignment (e.g., 9.2 Mbps from G to C). Moreover, using the WCETT metric helps find a path that has channel diversity (e.g., RR_1 in Fig. 9(b) favors the path G→H→I→F→C, and RR_2 favors the path G→H→E→F→C), but consumes more channel resource, due to the use of longer paths. On the other hand, ARS effectively discovers and uses idle channels through reconfiguration, thus satisfying QoS demands by up to three times more than static-assignment



(a) QoS-satisfaction gain

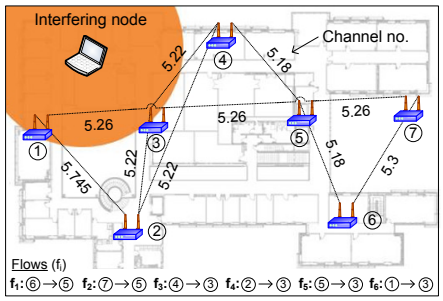
(b) Reconfiguration results (from G to C)

Fig. 9. *QoS-satisfaction gain*: ARS reconfigures networks to accommodate the varying QoS demands in time and space.

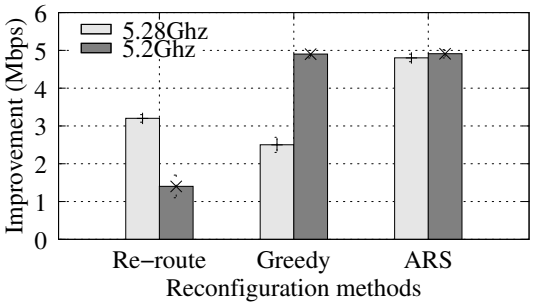
algorithms (e.g., the bandwidth from G to C in Fig. 9(a)).

3) Avoidance of ripple effects: We also studied ARS's effectiveness in avoiding the ripple effects of network reconfiguration. Fig. 10(a) shows initial channel and flow assignments in a part of our testbed. In this topology, we run 6 UDP flows ($f_1, \cdots, f_6$) each at 4Mbps, and measure each flow's throughput while injecting interference into a target channel. We run same scenarios with 2 different interference frequencies (5.28 and 5.2 Ghz) to induce failures on different links. Also, we use three failure-recovery methods (i.e., local re-routing, greedy, and ARS) for comparison.

Since ARS considers the effects of local changes on neighboring nodes via *aBAR*, it effectively identifies reconfiguration plans that avoid the ripple effects. Fig. 10(b) shows the average throughput improvement of the flows after network reconfigurations, with each of the three recovery schemes. First, with interference on 5.28 Ghz, nodes 1, 3, and 5 experience link-quality degradation, degrading 5 Mbps throughput among the 6 flows. Under ARS, the network performs reconfiguration and recovers an average 98% of the degraded throughput



(a) Channel and flow assignment



(b) Improvement from the degraded throughput (5 Mbps) by each reconfiguration method

Fig. 10. *ARS's avoidance of ripple effects*: ARS finds a local reconfiguration plan that avoids the ripple effects by considering neighboring nodes' channel utilization, whereas the greedy channel-switching and local re-routing cannot fully recover from the failure or cause additional QoS failures.

(4.8 Mbps). On the other hand, the flows via local re-routing achieves 82% of the throughput (3.2 Mbps), because of the use of detour paths (f_5:5→4→3, f_6:1→2→3). On the other hand, while partially recovering from the original link failures, the greedy approach causes throughput degradation of neighboring links. This is because one local greedy channel-switching (from 5.26 to 5.32 Ghz) requires the neighboring links' channel (e.g., between nodes 5 and 7) to change, creating interference to other neighboring nodes' link (e.g., between nodes 6 and 7) that use adjacent channels (e.g., 5.3 Ghz).

Next, in the second interference case (5.2 Ghz), ARS also identifies QoS-satisfying reconfiguration plans, achieving 97% throughput improvement over the degraded throughput, as shown in Fig. 10(b). On the detection of the interference, ARS switches the channel of all the fault-related links, based on its planning algorithm to other channel. Naturally, this result (configuration and throughput) is the same as that achieved by the greedy method. On the other hand, the local re-routing causes heavy channel contention for detour paths, degrading neighboring flows' performance (i.e., f_5) as well as others' (f_3, f_4).

V. Performance Evaluation

We have also evaluated ARS in large-scale network settings via simulation. We first describe our simulation methodology, and then present the evaluation results on ARS.

A. The Simulation Model & Methods

ns-2 [38] is used in our simulation study. Throughout the simulation, we use a grid topology with 25 nodes in an area of 1Km×1Km, as shown in Fig. 11(a). In the topology, adjacent nodes are separated by 180m and each node is equipped with a different number of radios, depending on its proximity to a gateway. The gateway is equipped with four radios, one-hop-away nodes from a gateway have three radios, and other nodes have two radios.

For each node in the above topology, we use the following network protocol stacks. First, the shadowing propagation model [39] is used to simulate varying channel quality and multi-path effects. Next, CMU 802.11 wireless extension is used for the MAC protocol with a fixed data rate (i.e., 11 Mbps) and is further modified to support multiple radios and multiple channels. Finally, a link-state routing protocol, a modification of DSDV [40], and multi-radio-aware routing metric (WCETT [6]) are implemented and used for routing.

In these settings, ARS is implemented as an agent in both the MAC layer and a routing protocol as explained in Sections III and IV. It periodically collects channel information from MAC, and requests channel switching or link-association changes based on its decision. At the same time, it informs the routing protocol of network failures or a routing table update.

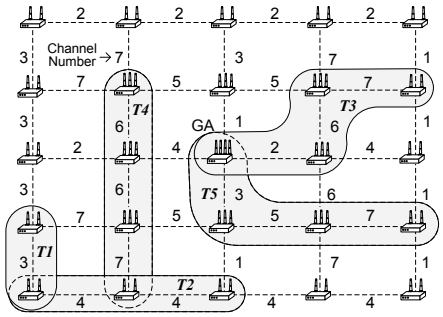
There are several settings to emulate real-network activities. First, to generate users' traffic, multiple UDP flows between a gateway and randomly-chosen mesh nodes are introduced. Each flow runs at 500 Kbps with a packet size of 1000 bytes. Second, to create network failures, uniformly-distributed channel faults are injected at a random time point. Random bit-error is used to emulate channel-related link failures and lasts for a given failure period. Finally, all experiments are run for 3000 seconds, and the results of 10 runs are averaged unless specified otherwise.

B. Evaluation Results

1) Effectiveness of QoS-aware planning: We measured the effectiveness of ARS in meeting the varying QoS requirements in a mr-WMN. We initially assign symmetric link capacity as shown in the channel assignment of the grid topology (Fig. 11(a)). Then, while changing the QoS constraints in gray areas at different times (i.e., $T1, \ldots, T5$), we evaluate the improvement of available capacity that ARS can generate via reconfiguration.

As shown in the tables of Fig. 11(b), ARS reconfigures a wireless mesh network to meet different QoS requirements. Before each reconfiguration, the gray areas can only accept 1 to 9 UDP flows. On the other hand, after reconfiguration, the network in the areas can admit 4 to 15 additional flows, improving the average network capacity of the gray areas by 3.5 times.

2) Impact of the benefit function: We also studied the impact of the benefit function on the ARS's planning algorithm. We conducted the same experiment as the previous one with different values of δ in the benefit function. As shown in Fig. 11(b), a high value (0.8) of δ allows ARS to keep local channel-efficiency high. By contrast, a low value (0.4) can deliver more available bandwidth (on average, 1.2 Mbps) than when the high value is used, since ARS tries to reserve more capacity.



(a) Scenarios for diverse QoS requests

Table-1: $\delta = 0.8$

Time	T1		T2		T3		T4		T5	
	(i)	(ii)	(i)	(ii)	(i)	(ii)	(i)	(ii)	(i)	(ii)
Admission	*no*	*yes*	*no*	*yes*	*no*	*yes*	*no*	*yes*	*no*	*yes*
Available capacity(Mb/s)	0.5	3.5	0.5	2.0	1.0	3.0	3.0	7.5	0.5	1.5

Table-2: $\delta = 0.4$

Time	T1		T2		T3		T4		T5	
	(i)	(ii)	(i)	(ii)	(i)	(ii)	(i)	(ii)	(i)	(ii)
Admission	*no*	*yes*	*no*	*yes*	*no*	*yes*	*no*	*yes*	*no*	*yes*
Available capacity(Mb/s)	0.5	5.5	0.5	5.5	1.0	3.0	0.5	1.5	0.5	1.5

(b) QoS benefits from reconfigurations

Fig. 11. *Satisfying varying QoS constraints*: Fig. 11(a) shows requests with different QoS requirements. Next, Fig. 11(b) shows improved (or changed) network capability (i) before and (ii) after reconfiguration.

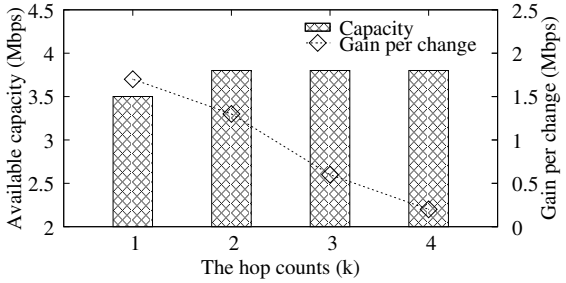


Fig. 12. *The impact of reconfiguration range*: The hop length can help ARS search for reconfiguration plans. However, the benefit from the increased length is small, whereas the number of total changes for the reconfiguration adversely increases.

3) Impact of the reconfiguration range: We evaluated the impact of the reconfiguration range. We used the same experiment settings as the previous one and focused on reconfiguration requests at $T1$. As we increase the hop count (k) from a faulty link(s), we measure the capacity improvement achieved by the reconfiguration plans. In addition, we calculate the capacity gain per change as the cost-effectiveness of reconfiguration planning with different k values.

Fig. 12 plots the available capacity of the faulty area after reconfigurations. As shown in the figure, ARS can improve the available links' capacity by increasing the reconfiguration range. However, its improvement becomes marginal as the range increases. This saturation results mainly from the fixed number of radios of each node. In other words, the improvement is essentially bounded by the total capacity of physical radios. Furthermore, because reconfiguration plans with a larger range are required to incur more changes in network settings, the bandwidth gain per change significantly degrades (e.g., capacity gain per change at the hop-count of 4 in Fig. 12). We also observed the similar results in other reconfiguration requests ($T2$,$T3$,$T4$), but omitted them for brevity.

VI. Conclusion

We first make concluding remarks and then discuss some of future work.

A. Concluding Remarks

This paper presented an Autonomous network Reconfiguration System (ARS) that enables a multi-radio WMN to autonomously recover from wireless link failures. ARS generates an effective reconfiguration plan that requires only local network configuration changes by exploiting channel, radio, and path diversity. Furthermore, ARS effectively identifies reconfiguration plans that satisfy applications' QoS constraints, admitting up to two times more flows than static assignment, through QoS-aware planning. Next, ARS's on-line reconfigurability allows for real-time failure detection and network reconfiguration, thus improving channel-efficiency by 92%. Our experimental evaluation on a Linux-based implementation and *ns-2*-based simulation have demonstrated the effectiveness of ARS in recovering from local link-failures and in satisfying applications' diverse QoS demands.

B. Future Work

Joint optimization with flow assignment and routing: ARS decouples network reconfiguration from flow assignment and routing. Reconfiguration might be able to achieve better performance if two problems are jointly considered. Even though there have been a couple of proposals to solve this problem [5], [12], they only provide theoretical bounds without considering practical system issues. Even though its design goal is to recover from network failures as a best-effort service, ARS is the first step to solve this optimization problem, which we will address in a forthcoming paper.

Use of ARS in IEEE 802.11b/g WMNs: ARS is mainly evaluated in IEEE 802.11a networks, where 13 orthogonal channels are available. However, ARS can also be effective in a network with a small number of orthogonal channels (e.g., 3 in IEEE 802.11b/g). Because ARS includes a link-association primitive, it can learn available channel capacity by associating with idle interfaces of neighboring nodes, and it further limits the range of a reconfiguration group (e.g., nodes within 4 hops).

Acknoweldgements

The work reported in this paper was supported in part by NSF under Grants CNS-0519498 and CNS-0721529, and Intel Corporation. The authors would like to thank networking

research group members in RTCL and other graduate students in SSL for helping us build a multi-radio WMN in the CSE building at the University of Michigan.

REFERENCES

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

[2] MIT Roofnet, http://www.pdos.lcs.mit.edu/roofnet.

[3] Motorola Inc. Mesh Broadband, http://www.motorola.com/mesh.

[4] P. Kyasanur and N. Vaidya, "Capacity of multi-channel wireless networks: Impact of number of channels and interfaces," in *Proceedings of ACM MobiCom*, Cologne, Germany, Aug. 2005.

[5] K. Ramanchandran, E. Belding-Royer, and M. Buddhikot, "Interference-aware channel assignment in multi-radio wireless mesh networks," in *Proceedings of IEEE InfoCom*, Barcelona, Spain, Apr. 2006.

[6] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *Proceedings of ACM MobiCom*, Philadelphia, PA, Sept. 2004.

[7] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: Slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks," in *Proceedings of ACM MobiCom*, Philadelphia, PA, Sept. 2004.

[8] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level measurements from an 802.11b mesh network," in *Proceedings of ACM SIGCOMM*, Portland, OR, Aug. 2004.

[9] A. Akella, G. Judd, S. Seshan, and P. Steenkiste, "Self-management in chaotic wireless deployments," in *Proceedings of ACM MobiCom*, Cologne, Germany, Sept. 2005.

[10] J. Zhao, H. Zheng, and G.-H. Yang, "Distributed coordination in dynamic spectrum allocation networks," in *Proceedings of IEEE DySPAN*, Baltimore, MD, Nov. 2005.

[11] M. J. Marcus, "Real time spectrum markets and interruptible spectrum: New concepts of spectrum use enabled by cognitive radio," in *Proceedings of IEEE DySPAN*, Baltimore, MD, Nov. 2005.

[12] M. Alicherry, R. Bhatia, and L. Li, "Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks," in *Proceedings of ACM MobiCom*, Cologne, Germany, Aug. 2005.

[13] M. Kodialam and T. Nandagopal, "Characterizing the capacity region in multi-radio multi-channel wireless mesh networks," in *Proceedings of ACM MobiCom*, Cologne, Germany, Aug. 2005.

[14] A. Brzezinski, G. Zussman, and E. Modiano, "Enabling distributed throughput maximization in wireless mesh networks-a partitioning approach," in *Proceedings of ACM MobiCom*, Los Angeles, CA, Sept. 2006.

[15] A. Raniwala and T. Chiueh, "Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network," in *Proceedings of IEEE InfoCom*, Miami, FL, Mar. 2005.

[16] S. Nelakuditi, S. Lee, Y. Yu, J. Wang, Z. Zhong, G. Lu, and Z. Zhang, "Blacklist-aided forwarding in static multihop wireless networks," in *Proceedings of IEEE SECON*, Santa Clara, CA, Sept. 2005.

[17] S. Chen and K. Nahrstedt, "Distributed quality-of-service routing in ad hoc networks," *IEEE JSAC*, vol. 17, no. 8, 1999.

[18] L. Qiu, P. Bahl, A. Rao, and L. Zhou, "Troubleshooting multi-hop wireless networks," in *Proceedings of ACM SigMetrics (extended abstract)*, Alberta, Canada, June 2005.

[19] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott, "Experimental evaluation of wireless simulation assumptions," Dept. of Computer Science, Dartmouth College, Tech. Rep. TR2004-507, 2004.

[20] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campus-wide wireless network," in *Proceedings of ACM MobiCom*, Philadelphia, PA, Sept. 2004.

[21] M. Buddhikot, P. Kolodzy, S. Miller, K. Ryan, and J. Evans, "DIMSUMnet: New directions in wireless networking using coordinated dynamic spectrum access," in *Proceedings of IEEE Symposium on a World of Wireless Mobile and Multimedia Networks*, Naxos, Italy, June 2005.

[22] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol (rsvp)," Internet Request for Comments 2205 (rfc2205.txt), Sept. 1997.

[23] D. S. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proceedings of ACM MobiCom*, San Diego, CA, Sept. 2003.

[24] C. Perkins, E. Belding-Royer, and S. Das, "Ad-hoc on-demand distance vector routing," Internet Request for Comments 3561 (rfc3561.txt), July 2003.

[25] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *the Book of Mobile Computing*. Kluwer Academic Publishers, 1996, vol. 353.

[26] G. Holland, N. Vaidya, and P. Bahl, "A rate-adaptive MAC protocol for multi-hop wireless networks," in *Proceedings of ACM MobiCom*, Rome, Italy, Sept. 2001.

[27] J. L. Gross and J. Yellen, "Graph theory and its applications, 2nd edition," Chapman & Hall/CRC, 2006.

[28] A. P. Subramanian, H. Gupta, S. R. Das, and J. Cao, "Minimum interference channel assignment in multi-radio wireless mesh networks," *IEEE Transaction on Mobile Computing*, Dec. 2008.

[29] A. S. Tanenbaum and M. V. Steen, "Distributed systems," Pearson Education.

[30] Q. Xue and A. Ganz, "Ad hoc QoS on-demand routing (AQOR) in mobile ad hoc networks," *ACM Journal of Parallel and Distributed Computing*, vol. 63, no. 2, pp. 154–165, 2003.

[31] K.-H. Kim and K. G. Shin, "Accurate and asymmetry-aware measurement of link quality in wireless mesh networks," *To appear in IEEE/ACM Transactions on Networking*, 2009.

[32] Netfilter, http://www.netfilter.org.

[33] MADWiFi, http://www.madwifi.org.

[34] Atheros Communications, http://www.atheros.com.

[35] Soekris Engineering, http://www.soekris.com.

[36] Iperf, "Network measurement tool," http://dast.nlanr.net/Projects/Iperf/.

[37] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *Proceedings of ACM MobiCom*, Philadelphia, PA, Sept. 2004.

[38] "*ns-2* network simulator." http://www.isi.edu/nsnam/ns.

[39] T. S. Rappaport, "Wireless Communications: Principles and Practice," Prentice Hall, 2002.

[40] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *ACM SIGCOMM*, London, UK, Sept. 1994, pp. 234–244.

[41] M. M. Carvalho and J. J. Garcia-Luna-Aceves, "A scalable model for channel access protocols in multihop ad hoc networks," in *Proceedings of ACM MobiCom*, Philadelphia, PA, Sept. 2004.

[42] ——, "Delay analysis of IEEE 802.11 in single-hop networks," in *Proceedings of IEEE ICNP*, Atlanta, GA, Nov. 2003.

[43] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE JSAC*, vol. 18, no. 3, Mar. 2000.

[44] H. Wu, X. Wang, Y. Liu, Q. Zhang, and Z.-L. Zhang, "SoftMAC: Layer 2.5 mac for VoIP spport in multi-hop wireless networks," in *Proceedings of IEEE SECON*, Santa Clara, CA, Sept. 2005.

[45] S. Lee, S. Banerjee, and B. Bhattacharjee, "The case for a multi-hop wireless local area network," in *Proceedings of IEEE InfoCom*, Hong Kong, Mar. 2004.

APPENDIX

A. Link Capacity

Although there are a considerable number of analytical models for link capacity [41]–[44], ARS uses a simple and realistic model based on both MAC behavior and actual measurement results similar to the derivation in [45]. Link capacity (C) can be derived by estimating the expected packet transmission latency (t_l), which consists of back-off time (t_b) and actual transmission time (t_s). First, the initial value of t_b is determined based on a uniformly-chosen random value in [0, CWMin]. On average, t_b is $\frac{CWMin \times slotTime}{2}$. However, because the window size exponentially increases every time the transmission attempt fails, t_b^i becomes $2^i \times \frac{CWMin \times slotTime}{2}$ on i consecutive failures.

Next, the packet-transmission time t_s includes inter-frame time (i.e., SIFS, DIFS) and RTS/CTS, DATA/ACK frame transmission time. While inter-frame times and control/ACK frames consume a fixed amount of time, a data frame takes different amounts of time due to different transmission rates (c). Thus, t_s can be represented as $\frac{RTS+CTS+ACK}{2.0 \times 10^6} + \frac{DATA}{c} + 3SIFS + DIFS$.

Given t_b, t_s, and the measured data-delivery ratio d, the expected latency or t_l for a packet transmission can be derived as:

$$t_l = \sum_{i=0}^{n_r} (1-d)^i \times d \times \{t_b^i + (i+1) \times t_s\}, \quad C = \frac{DATA}{t_l} \quad (2)$$

where n_r is a retry limit at the MAC layer, and d and c are obtained from ARS's monitoring protocol.

Kyu-Han Kim (S'02-M'09) received the BS from the Korea University, Seoul, Korea in 2000, the MS from the from the Georgia Institute of Technology, Atlanta, GA in 2003, and the PhD from the University of Michigan, Ann Arbor, MI in 2009, all in computer science. He is currently a senior research scientist at Deutsche Telekom Inc., R&D Lab U.S.A. His research interests include performance, Quality-of-Service, fault-tolerance, and manageability in mobile/distributed systems and wireless networks. He is a recipient of the ACM MobiCom Best Student Paper Award (Co-author, 2003) and the government scholarship (2001–2005) from the Ministry of Information and Communication, Republic of Korea.

Kang G. Shin (S'75-M'78-SM'83-F'92) is the Kevin and Nancy O'Connor Professor of Computer Science and Founding Director of the Real-Time Computing Laboratory in the Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, Michigan.

His current research focuses on computing systems and networks as well as on embedded real-time and cyber-physical systems, all with emphasis on timeliness, security, and dependability. He has supervised the completion of 67 PhDs, authored/coauthored more than 740 technical articles (263 of which are published in archival journals) and more than 20 patents or invention disclosures. He has co-authored (with C. M. Krishna) a textbook "Real-Time Systems," McGraw Hill, 1997.

He has received numerous best paper awards, including the Best Paper from the 2010 USENIX Annual Technical Conference, the IEEE Communications Society William R. Bennett Prize Paper Award in 2003, the Best Paper Award from the IWQoS'03 in 2003, and an Outstanding IEEE Transactions of Automatic Control Paper Award in 1987. He has also coauthored papers with his students which received the Best Student Paper Awards from the 1996 IEEE Real-Time Technology and Application Symposium, and the 2000 UNSENIX Technical Conference. He has also received several institutional awards, including the Research Excellence Award in 1989, Outstanding Achievement Award in 1999, Service Excellence Award in 2000, Distinguished Faculty Achievement Award in 2001, and Stephen Attwood Award in 2004 from The University of Michigan (the highest honor bestowed to Michigan Engineering faculty); a Distinguished Alumni Award of the College of Engineering, Seoul National University in 2002; 2003 IEEE RTC Technical Achievement Award; and 2006 Ho-Am Prize in Engineering (the highest honor bestowed to Korean-origin engineers).

He received the B.S. degree in Electronics Engineering from Seoul National University, Seoul, Korea in 1970, and both the M.S. and Ph.D degrees in Electrical Engineering from Cornell University, Ithaca, New York in 1976 and 1978, respectively. From 1978 to 1982 he was on the faculty of Rensselaer Polytechnic Institute, Troy, New York. He also chaired the Computer Science and Engineering Division, EECS Department, The University of Michigan for three years beginning January 1991.

He has held visiting positions at the U.S. Airforce Flight Dynamics Laboratory, AT&T Bell Laboratories, Computer Science Division within the Department of Electrical Engineering and Computer Science at UC Berkeley, and International Computer Science Institute, Berkeley, CA, IBM T. J. Watson Research Center, Software Engineering Institute at Carnegie Mellon University, HP Research Laboratories, Hong Kong University of Science and Technology, Ewha Womans University in Korea, and Ecole Polytechnique Federale de Lausanne (EPFL) in Switzerland.

He is Fellow of IEEE and ACM, and overseas member of the Korean Academy of Engineering, has chaired numerous major conferences, including 2009 ACM Annual International Conference on Mobile Computing and Networking (MobiCom'09), 2008 IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'08), the 3rd ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys'05), 2000 IEEE Real-Time Technology and Applications Symposium (RTAS'00), and 1987 IEEE Real-Time Systems Symposium (RTSS). He also chaired the IEEE Technical Committee on Real-Time Systems during 1991-93, was a Distinguished Visitor of the Computer Society of the IEEE, an Editor of *IEEE Trans. on Parallel and Distributed Computing*, and an Area Editor of *International Journal of Time-Critical Computing Systems*, *Computer Networks*, and *ACM Transactions on Embedded Systems*. He has also served or is serving on numerous government committees, such as the US NSF Cyber-Physical Systems Executive Committee and the Korean Government R&D Strategy Advisory Committee.