12-2010

A Policy-Based Constraint-Solving Platform Towards Extensible Wireless Channel Selection and Routing

Changbin Liu
University of Pennsylvania

Xiaozhou Li
University of Pennsylvania

Shiv Muthukumar
University of Pennsylvania

Harjot Gill
University of Pennsylvania

Taher Saeed
University of Pennsylvania

See next page for additional authors

A Policy-Based Constraint-Solving Platform Towards Extensible Wireless Channel Selection and Routing

Abstract
This paper presents PUMA, a novel declarative constraintsolving platform that achieves efficient policy-based channel selection and routing for multi-radio wireless mesh networks. PUMA is based on declarative networking, a databaseinspired extensible infrastructure using query languages to specify behavior. In PUMA, users specify high-level declarative policies that dictate their channel selection constraints and routing protocol behavior. We demonstrate that channel selection can be expressed in a compact fashion and implemented efficiently. We have developed a PUMA prototype based on the RapidNet declarative networking engine with enhancements to handle multi-channel communication and integration with an open-source constraint solver. We perform preliminary evaluation of PUMA using the emerging ns-3 network simulator, and describe our ongoing research in ORBIT testbed deployment, distributed channel selection protocols, and distributed optimizations that combine routing and channel selection.

Disciplines
Computer Sciences

Comments
Liu, C., Li, X., Muthukumar, S., Gill, H., Saeed, T., Loo, B., & Basu, P., A Policy-Based Constraint-Solving Platform Towards Extensible Wireless Channel Selection and Routing, *ACM Workshop on Programmable Routers for Extensible Services of Tomorrow (PRESTO), in conjuction with ACM CoNEXT*, Dec. 2010

Author(s)
Changbin Liu, Xiaozhou Li, Shiv Muthukumar, Harjot Gill, Taher Saeed, Boon Thau Loo, and Prithwish Basu

A Policy-based Constraint-solving Platform Towards Extensible Wireless Channel Selection and Routing

Changbin Liu* Xiaozhou Li* Shiv Muthukumar* Harjot Gill*
Taher Saeed* Boon Thau Loo* Prithwish Basu†
*University of Pennsylvania †Raytheon BBN Technologies
{changbl, xiaozhou, mshivk, gillh, taher, boonloo}@seas.upenn.edu, pbasu@bbn.com

ABSTRACT

This paper presents PUMA, a novel declarative constraint-solving platform that achieves efficient policy-based channel selection and routing for multi-radio wireless mesh networks. PUMA is based on *declarative networking*, a database-inspired extensible infrastructure using query languages to specify behavior. In PUMA, users specify high-level declarative policies that dictate their channel selection constraints and routing protocol behavior. We demonstrate that channel selection can be expressed in a compact fashion and implemented efficiently. We have developed a PUMA prototype based on the RapidNet declarative networking engine with enhancements to handle multi-channel communication and integration with an open-source constraint solver. We perform preliminary evaluation of PUMA using the emerging ns-3 network simulator, and describe our ongoing research in ORBIT testbed deployment, distributed channel selection protocols, and distributed optimizations that combine routing and channel selection.

1. INTRODUCTION

Recently, the following trends have emerged in wireless networking: (1) transceivers supporting multiple tunable RF channels are becoming common; (2) devices with multiple wireless interfaces are becoming ubiquitous; (3) software defined radio technologies have developed into an active area of research with commercial uses [19]; and (4) the Federal Communications Commission (FCC) has opened up "white spaces" spectrum to unlicensed devices. Therefore, it is now more desirable (and feasible) to develop intelligent network protocols that simultaneously control parameters for dynamic (or agile) spectrum sensing and access, dynamic channel selection and medium access, and data routing with a goal of optimizing overall network performance. This is referred to as *Cognitive Radio Networking* [11].

Given the above technology trends, several architectures and designs for dynamic spectrum access/sharing [16, 19] and integrated channel selection and routing in cognitive radio networks (e.g. [18, 10, 9]) have been proposed for mitigating the impact of harmful interference and thus improving overall network performance. For reasonable operation of large mesh networks with nodes strewn over a large area with heterogeneous interference and traffic characteristics and constraints, we believe that a *one-size-fits-all* channel selection and/or routing protocol may be difficult to find.

To address the above needs, we propose PUMA[1], a novel declarative constraint-solving platform that achieves efficient policy-based routing and channel selection for multi-radio wireless mesh networks. PUMA is based on *declarative networking* [15], a database-inspired extensible infrastructure using *query languages* to specify behavior. Declarative programming allows programmers to say "what" they want, without worrying about the details of "how" to achieve it. PUMA allows wireless network providers to succinctly specify (or *declare*) the "policy rules and constraints of operation" for channel selection and routing. This allows the service providers a great degree of flexibility in the specification and enforcement of local and global policy constraints, and run-time configurability of policies.

In PUMA, users declaratively specify a rich set of communication goals, and communication policies and/or constraints on the procedure of selecting channels. The specifications attempt to capture *goals* such as minimize interference, maximize connectivity, maximize spectrum diversity, subject to one or more *constraints* specific to the deployment scenario. These specifications are then used as input to a constraint solver for determining channel assignments. PUMA's approach removes the dependence on *stove-piped* channel assignment and routing protocols programmed into various nodes in the network. For instance, we utilize our declarative framework to encode different interference models (e.g. one-hop vs two-hop), and further allow individual nodes to customize local policy constraints on selecting channels.

We have developed a PUMA prototype based on the RapidNet declarative networking engine [5] with enhancements to handle multi-radio multi-channel communication and integration with an open-source constraint solver [1] and the emerging ns-3 network simulator [2]. Our evaluations using wireless simulations demonstrate that PUMA can flexibly and efficiently implement centralized channel selection pro-

[1]Stands for *Policy-based Unified Multi-radio Architecture*.

ACM PRESTO 2010, November 30, 2010, Philadelphia, USA.

tocols that significantly outperform single-channel and naïve identical channel assignment.

We conclude this paper by describing some of our ongoing work, which includes deployment on the ORBIT [3] wireless testbed, enabling distributed channel selection protocols by leveraging the distributed capabilities of declarative networking, and a distributed protocol that optimizes across routing and channel selection policy decisions.

2. OVERVIEW

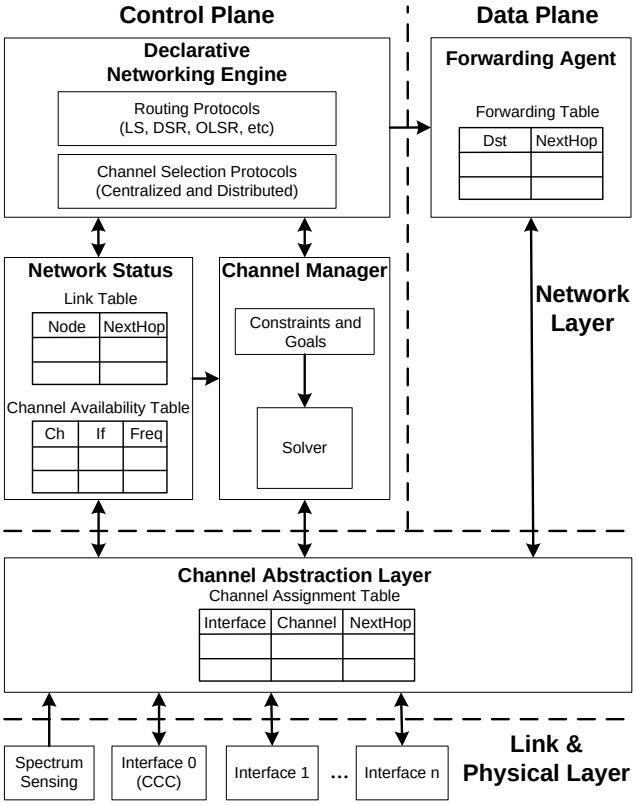

Figure 1: Components of a PUMA node

Figure 1 shows an overview of PUMA from the perspective of a single PUMA node. Each node runs a number of multi-channel wireless radio devices (interfaces). Typically, the first interface runs on the *common control channel* (CCC), reserved solely as a control channel for routing and channel selection protocol messages. A *spectrum sensing* component is able to detect channels available for each interface by periodically scanning a wide range of spectrum. The set of available channel information is then made available to the *channel manager* through the *channel abstraction layer* [7], which interacts with multiple radios and presents upper layers with a uniform communication interface. Using this information, the channel manager assigns a channel for communication with each of the node's neighbors.

At the network layer, a *declarative networking engine* [5] is deployed within the control plane to implement a variety of declarative routing protocols. These protocols can either be specified dynamically by the user, or pre-programmed as a library of declarative routing protocols (e.g. OLSR, DSR, and epidemic routing), and can themselves be selected via a series of policy rules [14]. The output of a declarative routing protocol is a forwarding table that indicates for a destination, the next hop to which the packet should be routed.

The *channel manager* selects and assigns channels that are used for data communication between pairs of neighbors. A typical goal of channel assignment is to ensure that pairs of adjacent nodes within communication range use a common channel such that the overall network interference is minimized. The channel assignment is performed using a *constraint solver*, that takes as input existing network state information, which consists of a *link table* that stores the network topology, and a *channel availability table* that contains the set of channels available to nodes.

The channel manager can be deployed either as *centralized* or *distributed*. In the centralized model, all nodes send their local neighborhood and channel availability information to a centralized channel manager whose "solver" takes the entire topology and per-node information into account to perform channel assignment. In the distributed model, each node makes the channel assignment decisions using its own "solver", with only information gathered from immediate neighbors or neighbors within the vicinity. In this paper, we focus on the centralized model, and briefly discuss the distributed model in Section 6. After channel selection, channel assignments are then exchanged with neighboring nodes via *channel selection protocols* implemented using declarative networking engine to initialize the channel assignment table at the channel abstraction layer of each node.

In addition to the input topology and available channel information, the solver can take additional user-defined constraints and policy goals and determines a channel assignment that meets the goal. In the PUMA framework, policy constraints and optimization goals are all specified in a declarative language.

PUMA focuses primarily on routing and channel selection, and assumes the presence of a correctly functioning spectrum sensing component. Hence, we consider the challenges of spectrum sensing to constitute a set of separate and orthogonal problems that are outside the scope of this paper.

3. DECLARATIVE NETWORKING ENGINE

PUMA uses a declarative networking engine as a basis for executing routing and channel selection protocols. Based on Figure 1, the outputs of PUMA's declarative networking component are the `link` and `forwarding` tables, as well as any other network state required in routing and channel selection protocols.

Declarative networking [15] aims to build extensible architectures that achieve a good balance of flexibility, performance and safety. Declarative networks are specified using *Network Datalog* (*NDlog*), which is a distributed recursive query language. Declarative queries such as *NDlog* are a natural and compact way to implement a variety of routing protocols and overlay networks [15]. In our prior work [14], we have further demonstrated on the ORBIT wireless testbed, the feasibility of declarative mobile ad-hoc network (MANET) routing protocols, and capabilities to adapt among these protocols at runtime based on specified policies. When com-

piled and executed, these declarative networks perform efficiently relative to imperative language implementations, as shown by open-source implementations [4, 5].

NDlog is based on Datalog [17]: a Datalog program consists of a set of declarative *rules*. Each rule has the form `p :- q1, q2, ..., qn.`, which can be read informally as "`q1` and `q2` and ... and `qn` implies `p`". Here, `p` is the *head* of the rule, and `q1`, `q2`,...,`qn` is a list of *literals* that constitutes the *body* of the rule. Literals are either *predicates* with *attributes* (which are bound to variables or constants by the query), or boolean expressions that involve function symbols (including arithmetic) applied to attributes.

Datalog rules can refer to one another in a mutually recursive fashion. The order in which the rules are presented in a program is semantically immaterial; likewise, the order predicates appear in a rule is not semantically meaningful. Commas are interpreted as logical conjunctions (*AND*). Conventionally, the names of predicates, function symbols, and constants begin with a lowercase letter, while variable names begin with an uppercase letter. Function calls are additionally prepended by `f_`. Aggregate constructs are represented as functions with attribute variables within angle brackets (`<>`). We illustrate *NDlog* using a simple example below, which is link-state (LS) routing:

```
ls1 lsu(@X,X,Y,C,X) :- link(@X,Y,C).
ls2 lsu(@M,X,Y,C,Z) :- link(@Z,M,C1),lsu(@Z,X,Y,C,W),M!=W.
```

`lsu(@M,X,Y,C,Z)` is a link state update (LSU) corresponding to `link(X,Y,C)`, which indicates a link between node `X` and `Y` with a cost of `C`. The `link` table forms the basis of routing protocols – it represents the neighborhood information gathered at each node, and can itself be generated via a neighbor discovery process expressible in *NDlog* [5]. The cost of each link can also be customized, e.g. based on link RTT or expected transmission count (ETX) [8] or expected transmission time (ETT) [9].

This LSU tuple is flooded in the network starting from source node `X`. During the flooding process, node `M` is the current node it is flooded to, while node `Z` is the node that forwarded this tuple to node `M`. Rule `ls1` *generates* an `lsu` tuple for every link at each node. Rule `ls2` states that each node `Z` that receives an `lsu` tuple *recursively forwards* the tuple to all neighbors `M` except the node `W` that it received the tuple from. *NDlog* supports a *location specifier* `@` in each predicate to denote the source location of each corresponding tuple. Datalog tables are set-valued, meaning that duplicate tuples are not considered for computation twice. This ensures that no similar `lsu` tuple is forwarded twice.

In link-state routing, once the `lsu` information is available at each node, additional rules can be written to compute routes for each node, and then select the routes with least hop count, as shown in [5]. This is in turn used to generate the `forwarding(@S,D,N)` table at each node `S`, which denotes that the optimal route to destination `D` is via immediate neighbor `N`. In addition to computing min-hop paths, *NDlog* rules on route selection can be customized to consider channel diversity, for instance, by utilizing the *Weighted Cumulative Expected Transmission Time (WCETT)* metric [9].

4. DECLARATIVE CHANNEL SELECTION

In this section, we describe how to declaratively specify channel selection rules and constraints. The role of the channel manager in Figure 1 is to assign available channels to wireless links to satisfy one of several potential objectives, e.g., minimize interference in the network, minimize the number of unique channels etc, while subject to additional channel assignment constraints. In the interference minimization example, the channel manager needs to assign channels to communication links between neighboring nodes in order to improve *channel diversity*, i.e. communication links that are within transmission range should ideally use non-conflicting channels that do not interfere with each other. This problem has been shown to map into the well-known *graph-coloring* problem [12], an NP-hard problem that aims to minimize the number of conflicting links.[2]

4.1 CSP Formulation

Our channel manager utilizes a *constraint solver* to perform channel selection formulated as a *constraint satisfaction problem* (CSP) [21]. A CSP formulation takes as input a set of constraints, and attempts to find an *assignment* of *values* chosen from an input *domain* to a set of *variables* to satisfy the constraints. Often, there can be multiple possible assignments; hence an optimization *goal* is supplied to the solver, to return the optimal assignment. The goal is typically expressed as a minimization over a cost function of the assignments. For example, in the context of channel assignment, the variables are the channels assigned to each communication link, while the values to be assigned are chosen from candidate channels available for each link. The goal in this case is to minimize the likelihood of interference among conflicting links.

To mathematically illustrate how channel selection maps into CSP, we consider the following example that avoids interference based on the *one-hop interference* model [22]. In this model, any two adjacent links are considered to interfere with each other if they are both using channels whose frequency bands are closer than a certain threshold. The formulation is as follows:

Input domain and variables: Consider a network $G = (V, E)$, where there are nodes $V = \{1, 2, \ldots, N\}$ and edges $E \subseteq V \times V$. Each node x has an available set of candidate channels A_x to select from, and a set of channels P_x currently occupied by primary users which own exclusive right to a certain spectrum band within its vicinity. The number of interfaces of each node is i_x. For simplicity of exposition, we assume interfaces are homogeneous and are able to use any candidate channel.

Optimization goal: For any two adjacent nodes $x, y \in V$, l_{xy} denotes the link between x and y. Channel assignment selects channel c_{xy} for each link l_{xy} to meet the following optimization goal:

$$\min \sum_{l_{xy}, l_{xz} \in E, y \neq z} cost(c_{xy}, c_{xz}) \qquad (1)$$

[2]The high-level approach is to model each wireless link as a vertex in a conflict graph, and each edge in the conflict graph denotes two links within interference range of each other. The equivalent graph coloring problem becomes minimizing the number of edges in conflict graph.

where $cost(c_{xy}, c_{xz})$ assigns a unit penalty if adjacent channel assignments c_{xy} and c_{xz} are separated by less than a specified frequency threshold $F_{mindiff}$:

$$cost(c_{xy}, c_{xz}) = \begin{cases} 1 & \text{if } |c_{xy} - c_{xz}| < F_{mindiff} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Constraints: The optimization goal has to be achieved under the following four constraints:

$$\forall l_{xy} \in E, c_{xy} \in A_x \quad (3)$$

$$\forall l_{xy} \in E, c_{xy} \notin P_x \quad (4)$$

$$\forall l_{xy} \in E, c_{xy} = c_{yx} \quad (5)$$

$$\forall x \in V, |\bigcup_{l_{xy} \in E} c_{xy}| \leq i_x \quad (6)$$

(3) ensures that each channel assignment c_{xy} is selected from the available channel domain A_x. (4) expresses the constraint that a node should not use channels currently occupied by primary users within its vicinity. (5) requires two adjacent nodes to communicate with each other using the same channel. (6) guarantees the number of assigned channels is no more than interfaces.

While the above CSP formulation can be hard-coded into existing constraint solvers, PUMA instead provides a declarative interface using *NDlog* for customizing the formulation in the form of policy rules. These policy rules are then compiled into actual implementation usable by a constraint solver.

PUMA also allows a good degree of flexibility in the definition and configuration of the CSP formulation. For instance, the problem of minimizing the number of unique channels in a network while ensuring no link conflicts [13] can be formulated in PUMA by making the cost function (2) a "hard constraint" which incurs infinite cost if violated and by replacing the optimization goal with $\min |\bigcup_{l_{xy} \in E} c_{xy}|$. *NDlog* not only facilitates policy customization, but also enables us to naturally integrate with declarative networking rules, to enable channel selection protocols.

Our implementation is based on the widely used *Gecode* [1] solver, although our framework is general enough to be applied to most constraint solvers. We next demonstrate the compilation process, by describing the mapping from the above CSP formulation to *NDlog* rules.

4.2 Declarative Specification

Our exploration of channel selection is based on a centralized channel selection protocol [18, 6]. In this protocol, we consider channel selection to be carried out separately from routing. The constraint solver is executed on a single node in the network. Typically, this node is pre-determined, or is chosen via a separate leader election protocol. The centralized solver collects the network status information from each node – this includes neighborhood information, available channels, and any additional local policies. Such status information is collected using link-state dissemination or its variants, which are themselves expressible in *NDlog* (see Section 3). Alternatively, if a route to the centralized solver has already been computed, each node can forward the status information directly to the centralized solver.

CSP	*NDlog*
symbol A_x	`availChannel(X,C,F,St)`
symbol c_{xy}	`assignChannel(X,Y,C)`
symbol i_x	`numInterface(X,K)`
symbol l_{xy}	`link(X,Y)`
symbol P_x	`primaryUser(X,C)`
equation (1)	rules `goal` and `s2`
equation (2)	rule `s1`
equation (3)	constraint `c1`
equation (4)	constraint `c2`
equation (5)	constraint `c3`
equation (6)	constraint `c4`

Table 1: Mappings from CSP to *NDlog*.

The following *NDlog* program specifies the one-hop interference model CSP formulation. Since the constraint solver is centralized and the entire topology and network-wide channel availability information is available at the solver, we omit the use of location specifiers in *NDlog*. Table 1 summarizes the mapping from CSP symbols to *NDlog* tables, and CSP equations to *NDlog* rules/constraints identified by the rule labels. In cases when symbols refer to sets, e.g. A_x and P_x, it matches more than one entry in the respective tables. For instance, A_x is represented by all `availChannel` entries where the first attribute is `x`.

```
// goal declaration
goal minimize C in totalcost(C)

// variable declaration
var assignChannel(X,Y,C) forall link(X,Y)

// cost assignment rules
s1 cost(X,Y,Z,C) :- assignChannel(X,Y,C1),
   assignChannel(X,Z,C2), Y!=Z, C=1,
   f_freqDist(C1,C2)<F_mindiff.
s2 totalCost(COUNT<C>) :- cost(X,Y,Z,C).

// Input domain constraint for assignChannel
c1 assignChannel(X,Y,C) -> link(X,Y), availChannel(X,C,F,St).

// primary user constraint
c2 availChannel(X,C,F,St) -> !primaryUser(X,C).

// channel symmetry constraint
c3 assignChannel(X,Y,Cx) -> assignChannel(Y,X,Cx).

// interface constraint
c4 uniqueChannel(X,Count) -> numInterface(X,K), Count <= K.
s3 uniqueChannel(X,UNIQUE<C>) :- assignChannel(X,Y,C).
```

The above rules take as input two tables `link(X,Y)` and `availChannel(X,C,F,St)`. The `link` table stores the gathered network topology information, as described in Section 3. The `availChannel` table is supplied by the channel abstraction layer via known spectrum sensing mechanisms, where each entry denotes that node `X` has an available channel `C` with frequency `F` and signal strength `St`. `C` is typically a globally known channel identifier. If interfaces are heterogeneous, i.e. different interfaces run on different range of channels, one additional attribute interface `I` can be added to the `availChannel` predicate.

The output of the solver is the table `assignChannel(X,Y,C)`, that indicates for neighbor `Y` of `X`, the channel `C` that is used for communication. The channel abstraction layer uses this information to select an unused interface to run on channel

C, and then update its internal forwarding state (i.e. the channel assignment table in Figure 1) to ensure that all messages forwarded to neighbors are directed to the selected interface.

4.2.1 Optimization Goal

Two reserved rules `goal` and `var` specify the goal and variables used by the constraint solver. The goal in this case is to minimize the cost attribute `C` in `totalCost`, while assigning channel variables for communication of all links.

Rules `s1-s2` provide the definitions of `totalCost` as follows. Rule `s1` sets cost `C` to 1 for each `cost(X,Y,Z,C)` tuple if the chosen channels that `X` uses to communicate with adjacent nodes `Y` and `Z` are interfering with each other. Rule `s2` counts the number of interfering channels among adjacent links in the entire network, and stores the value in `totalCost`. Note that the cost aggregation function is highly customizable, e.g., by expressing rule `s1` differently or using a different aggregation function other than `COUNT`.

In some wireless deployments, e.g. IEEE 802.11 network, the *two-hop interference model* [22] is often considered a more accurate measurement of interference. This model considers interference that results from any two links using similar channels within two hops of each other, and can be achieved via only minor modification to rule `s1`.

4.2.2 Channel Assignment Constraints

In addition to the optimization goal above, the solver can take as input additional *constraints*, that are used to remove from channel assignments that are considered illegal for the purpose of correct assignments. These constraints can be globally applied to all nodes, or customized at the node-level.

Our constraints are of the form F_1 `->` F_2, which denotes the logical meaning that whenever F_1 is true, then F_2 *must* also be true in order for the constraint not be violated. Unlike a rule, which derives new values for a predicate, a constraint *restricts* a predicate's allowed values, hence representing an invariant that must be maintained at all times.

The constraints `c1-c4` encodes the four constraints introduced in CSP formulation in Section 4.1. Constraint `c1` restricts the domain of `assignChannel(X,Y,C)` to only valid channel assignments for existing links `link(X,Y)` and ensures that only available channels are considered. Constraint `c2` applies to the input `availChannel` table, and states that a channel `C` at node `X` is only available, if there does not exist a primary user within the vicinity of `X`. Constraint `c3` enforces channel symmetry on the output `assignChannel` table. The fourth constraint `c4` requires that nodes must use at most K unique channels, where K is the number of usable interfaces. The number of unique channels is derived in rule `s3` using aggregate keyword `UNIQUE`.

In addition, one can declare additional constraints, e.g., avoid channels that have low SNR (a straightforward filter condition on `availChannel` table); ensure channel diversity along each path (by having the cost assignment take into account of interference along each best path).

5. PRELIMINARY EVALUATION

We have developed a prototype for PUMA based on the RapidNet declarative networking engine [5]. The RapidNet system is implemented as an add-on to ns-3 [2], an emerging discrete-event network simulator aimed to replace ns-2. ns-3 emulates all layers of the network stack and supports (configurable) packet loss, packet queuing, and network topology models. ns-3 supports a simulation mode based on the IEEE 802.11b PHY+MAC model thus enabling the controlled examination of PUMA's performance under various network topologies and conditions. As extensions to realize our PUMA prototype, we have enhanced ns-3 to support multi-radio multi-channel capabilities via the use of the *channel abstraction layer* described in Section 2. Moreover, we have integrated the RapidNet engine with the Gecode [1] constraint solver, in order to implement channel selection in Section 4.

5.1 Experimental Setup

In our setup, nodes utilize multiple interfaces consisting of homogeneous multi-channel radios. For simplicity, we limit the set of usable channels to "orthogonal channels", i.e. channels with sufficient frequency gap between them to incur minimal or no interference when active in each other's vicinity. This limits interference to situations where nearby links use the same channel.

We evaluate our declarative channel selection protocol in Section 4 and compare it with two baselines: *1-Interface* where all nodes communicate with each other using one interface and hence a common channel, and *Identical-Ch* [9], where the same set of channels are assigned to the interfaces to every node (e.g. channel 1 to the first interface, channel 2 to the second), and a centralized constraint solver then assigns each link to use one of these interfaces. In contrast, *PUMA* does not hardcode the mapping from interfaces to channels, but instead allows the solver to determine the mapping that best meets the specified optimization goal. In all our experiments, routing is based on the WCETT metric.

To determine the effectiveness of channel assignments, we inject traffic load with increasing sending rate, and then measure the *aggregate network throughput* (in terms of aggregate packets that are successfully received by their destinations), and *loss rate* (fraction of packets that are not delivered successfully at the receiver). Since loss rates can be derived by dividing the receiving over the sending rates for each workload, we present only the throughput graphs. In addition, to quantify the overhead of resources required for channel selection, we measure the solver execution time (dominating factor) for *PUMA*.

5.2 Simulation-based Results

Our experiments are carried out by running RapidNet over a simulated network based on ns-3's 802.11b PHY+MAC model modified to increase the number of orthogonal channels. Our simulations do not use RTS/CTS among nodes, but permit up to 3 retries at the MAC layer to transmit each packet. The simulated wireless mesh network consisting of 12 nodes randomly located in a $450m \times 450m$ arena. The transmission range of each node is approximately $100m$, and each node has an average degree of 4. Each node is equipped with one interface reserved for CCC, and two additional homogeneous data interfaces with 4 orthogonal channels each. The small network size enables us to achieve an optimal solution for centralized solver. We will discuss scaling up to larger networks in Section 6.

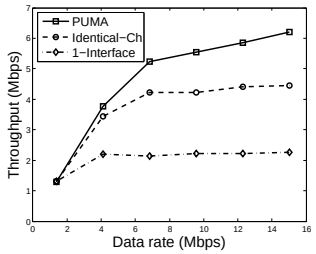

Figure 2: Aggregate network throughput.

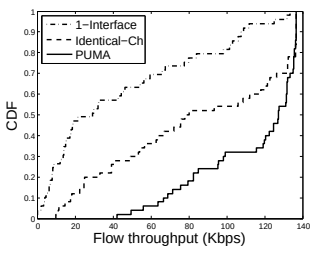

Figure 3: CDF of flow throughput.

Figure 2 shows the aggregate network throughput for all three protocols as the data sending rate increases. Our traffic load consists of packets sent from random sources to random destinations. We observe that the throughput for all protocols (expectedly) first increases linearly, then becomes sub-linear, and finally goes flat when network saturation is reached due to high interference. Comparing across protocols, *1-Interface* is the first to saturate (at $4.0Mbps$), followed by *Identical-Ch* (at $6.5Mbps$). In comparison, *PUMA* saturates at a much higher sending rate.

Figure 3 shows the CDF of network throughput breakdown by individual network flows. Here, instead of using a random traffic model, we selected 50 random source/destination pairs, and generate a steady stream of packets at a bidirectional sending rate of $70Kbps$ (in each direction) between each pair routed along the computed best path. Under this workload, *PUMA* still achieves the best performance.

In terms of the performance overhead, *PUMA* requires less than 10 seconds to perform channel selection on a Intel Quad core 2.33GHz PC with 4GB RAM running Ubuntu 10.04, demonstrating the efficiency of the solver in finding the optimal solution for a small network. Overall, our preliminary evaluation in simulations demonstrates that PUMA is able to implement, in a flexible and efficient manner, centralized channel selection and routing protocols that significantly outperform single-channel and the identical channel assignment in terms of high throughput and low loss rate.

6. ONGOING WORK

Scaling channel selection: While our initial evaluation results are encouraging, given that graph coloring is NP-hard, the centralized approach typically does not scale beyond small networks. One approximation method that we have explored is a *divide-and-conquer* strategy. The basic idea is to divide the whole network into roughly equal-sized subnetworks (a heuristic breadth-first search is used to partition the network), so that for each subnetwork the solver performs channel selection optimization and finishes in reasonable time.

Distributed channel selection: Like the divide-and-conquer strategy, distributed channel selection protocols provide approximations to the optimal centralized solution, and hence scale better for large networks. Furthermore, fully distributed protocols have the added advantages of not introducing single points of failure and are amenable to incremental computations (e.g., late joiners). One protocol that we are currently exploring is that proposed in [20]. In a nutshell, in this protocol each node periodically randomly selects one of its links to start a *channel negotiation* process with its neighbor. The negotiation process solves a *local* CSP in PUMA and assigns a channel such that interference is minimized. The process repeats until all links have been assigned channels. The use of location specifiers in declarative networking enables one to naturally capture constraints and policies with nearby neighbors within policy rules.

Integrated policies for route and channel selection: While certain specific resource allocation architectures and designs related to channel assignment and routing in multi-channel wireless networks have been proposed [18, 10], the problem of distributed channel selection and routing under a generic set of user-specified policy goals and constraints has not received much attention. We plan to study this problem in PUMA, by developing a distributed protocol that optimizes across routing and channel selection policies. Optimal routes are selected to avoid interference, and based on expected traffic loads along selected routes, the constraint solver re-optimizes channel assignments which are in turn used to find further fine-tuned routes. Leveraging the compact specifications of *NDlog*, this cross-layer protocol requires minor extensions to existing declarative protocols, to encode mutual dependencies between route and channel selection.

ORBIT testbed deployment: We are currently in the process of evaluating PUMA on the ORBIT wireless testbed. To this end, we have enhanced RapidNet with socket implementation that is capable of multi-radio multi-channel wireless communication on ORBIT. PUMA will be made able to toggle between ns-3 simulations and ORBIT deployment without changing the underlying system or *NDlog* programs.

7. ACKNOWLEDGMENTS

This work is supported in part by NSF grants CNS-0831376, CNS-0845552, and CCF-0820208.

8. REFERENCES

[1] Gecode constraint development environment. `http://www.gecode.org/`.
[2] Network Simulator 3. `http://www.nsnam.org/`.
[3] ORBIT Wireless Network Testbed. `http://www.orbit-lab.org/`.
[4] P2: Declarative Networking. `http://p2.cs.berkeley.edu/`.
[5] RapidNet. `http://netdb.cis.upenn.edu/rapidnet/`.
[6] V. Brik, E. Rozner, S. Banerjee, and P. Bahl. DSAP: a protocol for coordinated spectrum access. In *DySPAN*, 2005.
[7] C. Chereddi, P. Kyasanur, and N. H. Vaidya. Design and implementation of a multi-channel multi-interface network. In *REALMAN*, 2006.
[8] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *MobiCom*, 2003.
[9] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *MobiCom*, 2004.
[10] M. A. et. al. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In *MobiCom*, 2005.
[11] J. M. III and G. Maguire. Cognitive Radio: Making Software Radios More Personal. *IEEE Personal Communications*, August 1999.
[12] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. In *MobiCom*, 2003.
[13] F. Kuhn and R. Wattenhofer. On the complexity of distributed graph coloring. In *PODC*, 2006.
[14] C. Liu, R. Correa, X. Li, P. Basu, B. Loo, and Y. Mao. Declarative policy-based adaptive MANET routing. In *ICNP*, 2009.
[15] B. T. Loo, T. Condie, M. Garofalakis, D. E. Gay, J. M. Hellerstein, P. Maniatis, R. Ramakrishnan, T. Roscoe, and I. Stoica. Declarative Networking. In *Communications of the ACM (CACM)*, 2009.
[16] F. Perich. Policy-based Network Management for NeXt Generation Spectrum Access Control. In *DySpan*, 2007.
[17] R. Ramakrishnan and J. D. Ullman. A Survey of Research on Deductive Database Systems. *Journal of Logic Programming*, 23(2), 1993.
[18] A. Raniwala, K. Gopalan, and T.-c. Chiueh. Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 8(2):50–65, 2004.
[19] C. Santivanez, R. Ramanathan, C. Partridge, R. Krishnan, M. Condell, and S. Polit. Opportunistic spectrum access: Challenges, architecture, protocols. In *ACM WiCon*, Boston, MA, 2006.
[20] A. Subramanian, H. Gupta, and S. Das. Minimum Interference Channel Assignment in Multi-Radio Wireless Mesh Networks. In *SECON*, 2007.
[21] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, 1993.
[22] Y. Yi and M. Chiang. Wireless Scheduling Algorithms with O(1) Overhead for M-Hop Interference Model. In *IEEE ICC*, 2008.