A Multiagent System for Self-organisation of an 802.11 Mesh Network

John Debenham and Ante Prodan

Centre for Quantum Computation & Intelligent Systems,
University of Technology, Sydney, Australia
{debenham,aprodan}@it.uts.edu.au

Abstract. The self-organisation of telecommunications networks has to confront the two challenges of the scalability and the stability of the solution. This paper describes a distributed, co-operative multiagent system in which agents make decisions based only on local knowledge — that guarantees scalability. Extensive simulations indicate that stability is ensured by the agent's making improvements to the network settings that improve the social performance for all agents in a two-hop range. Our overall goal is simply to reduce maintenance costs for such networks by removing the need for humans to tune the network settings.

1 Introduction

Recent work on 802.11 Mesh Networks, such as [1], is predicated on a network whose prime purpose is to route traffic to and from nodes connected to the wired network — in which case there is assumed to be no traffic between end-user nodes. This introduces the conceptual simplification that mesh nodes can be seen as being grouped into clusters around a wired node where each cluster has a tree-like structure, rooted at a wired node, that supports the traffic. This is the prime purpose of 802.11 Mesh Networks in practice. Where possible, we move away from any assumptions concerning tree-like structures with the aim of designing algorithms for the more general classes of "wireless ad-hoc networks" or "wireless mesh networks". This paper is based on previous work in the area of mesh networking, and in particular in distributed algorithms at Columbia University, Microsoft Research, University of Maryland and Georgia Institute of Technology. See also: [2], [3], [4] and [5].

There are three principal inputs that we assume are available to the methods described: a load model, a load-balancing algorithm and an interference model. The work described below makes no restrictions on these three inputs other than that they are available to every node in the mesh. The load model, and so too the load balancing algorithm, will only be of value to a method for self-organisation if together they enable future load to be predicted with some certainty. We assume that the load is predictable. We assume that if the external demands on a set of nodes S are known and that there is a *load balancing algorithm* — that may or may not be intelligent — that determines how the load is routed through S. We assume that the load balancing algorithm will determine how the load is allocated to each link in the mesh.

V. Mařík, T. Strasser, and A. Zoitl (Eds.): HoloMAS 2009, LNAI 5696, pp. 244–253, 2009.

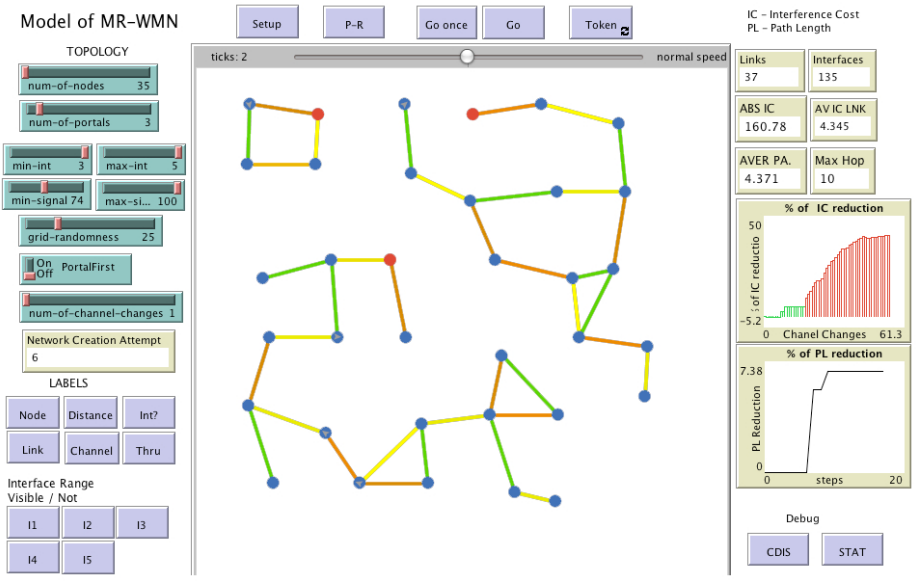

Fig. 1. The implementation of the algorithms

The multiagent system described in the paper has been simulated; the simulation is available[1] on the World Wide Web. A screen shot is shown in Figure 1. The "Setup" button establishes a random topology in line with the TOPOLOGY settings on the left side. The "P-R" button reduces the path length, and the "Go" button reduces the interference cost using the algorithms described in this paper. The colours on the arcs denote which of the eleven 802.11 channels is used. The work has been conducted in collaboration with Alcatel-Lucent, Bell Labs, Paris.

The measurement of interference cost is discussed in Section 2. Methods for the adjusting the channels in a multi-radio mesh networks for predictable load are described in Section 3, and for adjusting the links in Section 4. Future plans are described in Section 5.

2 Measuring Interference Cost

Suppose that during some time interval Δt two interfaces a and b are transmitting and receiving on channels Γ_a and Γ_b. During Δt, the *interference limit* that interface x imposes on interface y, $\tau_{y|x}$, is a ratio being the loss of traffic volume that interface y could receive if interface x were to transmit persistently divided by the volume of traffic that interface y could receive if interface x was silent:

$$\tau_{y|x} = \frac{(m_y \mid \text{ interface } x \text{ silent}) - (m_y \mid \text{ interface } x \text{ persistent})}{m_y \mid \text{ interface } x \text{ silent}}$$

[1] http://www-staff.it.uts.edu.au/~debenham/homepage/holomas/

where m_y is the mean SNIR observed by interface y whilst listening on channel Γ_y, where as many measurements are made as is expedient in the calculation of this mean[2]. The *interference load* of each interface, v_a and v_b, is measured as a proportion, or percentage, of some time interval during which that interface is transmitting. Then the *observed interference* caused by interface b transmitting on channel Γ_b as experienced by interface a listening on channel Γ_a is: $\tau_{a|b} \times v_b$, and the *observed interference cost* to interface a is[3]:

$$f(a \mid b) \triangleq \tau_{a|b} \times v_b \times (1 - v_a)$$

and so to interface b:

$$f(b \mid a) = \tau_{b|a} \times v_a \times (1 - v_b)$$

Now consider the interference between one interface a and two other interfaces c and d. Following the argument above, the *observed interference* caused by interfaces c and d as experienced by interface a is[4]: $\tau_{a|c} \times v_c + \tau_{a|d} \times v_d - \tau_{a|\{c,d\}} \times v_c \times v_d$. The observed interference cost to interface a is:

$$f(a \mid \{c,d\}) = (1 - v_a) \times \left(\tau_{a|c} \times v_c + \tau_{a|d} \times v_d - \tau_{a|\{c,d\}} \times v_c \times v_d\right)$$

If interfaces at agents c and d are linked then they will transmit on the same channel Γ_β, and we ignore the possibility of them both transmitting at the same time[5]. Further suppose that v_β is the proportion of Δt for which either interface c or interface d is transmitting. Then for some κ_β, $0 \leq \kappa_\beta \leq 1$: $v_c = \kappa_\beta \times v_\beta$, and $v_d = (1 - \kappa_\beta) \times v_\beta$. Thus:

$$f(a \mid \beta) = (1 - v_a) \times v_\beta \times \left(\tau_{a|c} \times \kappa_\beta + \tau_{a|d} \times (1 - \kappa_\beta)\right)$$

Now suppose that interfaces a and b are linked, and that v_α is the proportion of Δt for which either interface a or interface b is transmitting. Then for some κ_α, $0 \leq \kappa_\alpha \leq 1$: $v_a = \kappa_\alpha \times v_\alpha$, $v_b = (1 - \kappa_\alpha) \times v_\alpha$. Then as a will only receive interference when it is listening to b transmitting:

$$f(a \mid \beta) = v_b \times v_\beta \times \left(\tau_{a|c} \times \kappa_\beta + \tau_{a|d} \times (1 - \kappa_\beta)\right)$$

and so:

$$\begin{aligned} f(\alpha \mid \beta) = (1 - \kappa_\alpha) \times v_\alpha \times v_\beta \times \left(\tau_{a|c} \times \kappa_\beta + \tau_{a|d} \times (1 - \kappa_\beta)\right) \\ + \kappa_\alpha \times v_\alpha \times v_\beta \times \left(\tau_{b|c} \times \kappa_\beta + \tau_{b|d} \times (1 - \kappa_\beta)\right) \end{aligned} \quad (1)$$

Note that v_α, v_β, κ_α and κ_β are provided by the load model, and the $\tau_{x|y}$ are provided by the interference model.

[2] For $\tau_{y|x}$ to have the desired meaning, m_y should be a measurement of *link throughput*. However, link throughput and SNIR are approximately proportional — see [6].

[3] We assume here that whether or not interfaces a and b are transmitting are independent random events [7]. Then the probability that a is transmitting at any moment is v_a, and the probability that b is transmitting and a is listening at any moment is: $(1 - v_a) \times v_b$.

[4] That is, the interference caused by either interface c or interface d.

[5] The probability of two linked interfaces transmitting at the some time on an 802.11 mesh network can be as high as 7% — see [8], [9].

3 Adjusting the Channels

Each node is seen as an agent. The multiagent system exploits the distinction between proactive and reactive reasoning [10]. Proactive reasoning is concerned with planning to reach some goal. Reactive reasoning is concerned with dealing with unexpected changes in the agent's environment. The reactive logic provides an "immediate fix" to serious problems. The proactive logic, that involves deliberation and co-operation of nearby nodes, is a much slower process.

An agent (i.e. a node, or physically a router) with omnidirectional interfaces has three parameters to set for each interface: [1] The channel that is assigned to that interface; [2] The interfaces that that interface is linked to, and [3] The power level of the interface's transmission. Methods are describe for these parameters in the following sections. The following section describes how these three methods used combined in the proactive logic algorithm. The following methods all assume that there is a load balancing algorithm.

Informally the proactive logic uses the following procedure:

- *Elect* a node a that will manage the process
- *Choose* a link α from a to another node — precisely a trigger criterion (see below) permits node a to attempt to improve the performance of one of its links $\alpha \ni a$ with a certain priority level.
- *Measure* the interference
- *Change* the channel setting if this leads to a mean improvement for all the agents in a's interference range

The following is a development of the ideas in [2].

choose node a **at** time $t-2$;
set $V_a = \cup_{n \in S_a} S_n$;
$\forall x \in V_a$ **transmit** "**propose organise**$[a,x,p]$";
unless $\exists x \in V_a$ **receive** "**overrule organise**$[a,x,q]$" in
$[t-2,t-1]$ where $q > p$ **do** {
  $\forall x \in V_a$ **transmit** "**propose lock**$[a,x,t,t+1]$";
  if $\forall x \in V_a$ **receive** "**accept lock**$[a,x,t,t+1]$" in $[t-1,t]$
  then {
    unless $\exists x \in V_a$ **receive** "**reject lock**$[a,x,t,t+1]$"
      do {improve a;}
  }
}
where: improve $a = \{$
  choose link $\alpha \ni a$ on channel Γ^t_α;
  set $B \leftarrow \sum_{\beta \in S_\alpha} f(\alpha \mid \beta) + \sum_{\beta \in S_\alpha} f(\beta \mid \alpha)$;
  if (feasible) **re-route** α's traffic;
  for $\Gamma_\alpha = 1, \ldots, K, \Gamma_\alpha \neq \Gamma^t_\alpha$ **do**{
    if $\sum_{\beta \in S_\alpha} f(\alpha \mid \beta) + \sum_{\beta \in S_\alpha} f(\beta \mid \alpha) < B \times \varepsilon$ **then**{
      $\Gamma^{t+1}_\alpha \leftarrow \Gamma_\alpha$;
      selflock node a **in** $[t+1, t+k]$;

```
            break;
        };
    };
    ∀x ∈ V_a transmit "α's interference test signals";
    apply load balancing algorithm to S_a;
}
```

The statement **selflock** is to prevent a from having to activate the method too frequently. The constant $\varepsilon < 1$ requires that the improvement be 'significant' both for node a and for the set of nodes S_a. The stability of this procedure follows from the fact that it produces a net improvement of the interference cost within S_a. If a change of channel is effected then there will be no resulting change in interference outside S_a.

The above method reduces the net observed inference cost in the region V_a. It does so using values for the variables that appear on the right-hand side of Equation 1. If those values are fixed then the method will converge. The method above suggests the possibility that traffic is re-routed during the reassignment calculation — this is not essential.

3.1 Results and Discussion

The interference cost reduction for a link discussed herein is measured as the difference between absolute interference (AI) values obtained before the channel assignment process and after the channel assignment process. For example, if $AI_{before} = 5$ and $AI_{after} = 4$ the absolute difference is $AD = 1$ which is 20% decrease in the absolute interference. Consequently, the performance is always expressed as a percentage of the decrease. Our simulation studies consider realistic scenarios of different node densities and topologies in a typical wireless mesh network hence are more reflective of evaluating the true performance of the algorithm. In these studies the mean of interference cost (IC) reduction across all topologies and network (node) densities obtained is 36.7.

Impact of typical topologies on the interference cost. Figure 2(a) shows the variation in the interference cost reduction as a function of network topology across different node densities. It can be deduced that the impact of the topologies on the performance of the algorithm (i.e. in terms of interference cost reduction) is insignificant. The mean of IC reduction calculated from the data obtained shows that the topology with the smallest average IC reduction is the completely random with a mean of 36.02 and topology with the most IC reduction is the random grid with a mean of 37.12. The difference in performance between best and worst case is just 1.1 which confirms that the performance of the algorithm is almost completely independent of the type of topology.

Performance Comparison across the Network. In this study, we obtained interference cost (IC) in different regions of the MR-WMN for the same set of links before and after the self-organisation algorithm is invoked. Comparison of the results obtained is shown in Figure 2(b) where the Interference cost is on the X-axis. From Figure 2(b) we can see that there were no nodes (red dots) that caused more interference after the self-organisation than it had caused before (blue dots) the self-organisation was invoked.

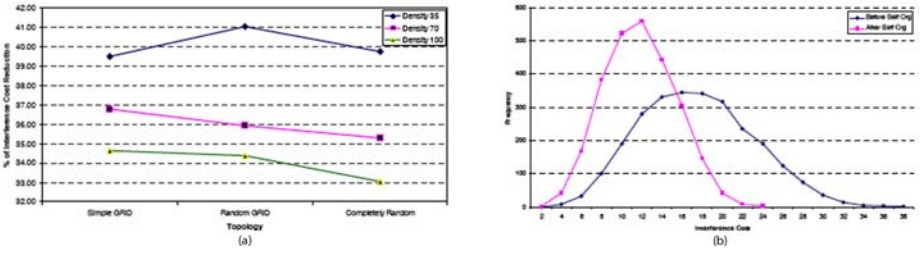

Fig. 2. (a) Interference cost reduction as a function of topologies. (b) Comparison of *IC* across the network before (blue) and after (red) self-organisation.

4 Adjusting the Links

The the first path-reduction algorithm is precisely the same as the algorithm in Section 3 but with the following 'improve' methods.

Link adjustment with known traffic load. Suppose that node a has interference range S_a. Let M_a be the set of nodes in S_a excluding node a. Then use the method in Section 3 with the following 'improve' method:

```
improve a = {
   for link α ∋ a, where α = [a,b]
   suppose α is on channel Γ^t_α;
   set B ← Σ_{β∈S_α} f(α | β) + Σ_{β∈S_α} f(β | α);
   if (feasible) re-route α's traffic;
   set γ ← α;
   for y ∈ M_a do {
      for Γ_[a,y] = 1,...,K, do {
         if Σ_{β∈S_a} f([a,y] | β) + Σ_{β∈S_a} f(β | [a,y]) < B × ε
          then {
              set γ ← [a,y];
              selflock node a in [t+1, t+k];
              break;
         };
      };
   };
   ∀x ∈ V_a transmit "γ's interference test signals";
   apply load balancing algorithm to S_a;
}
```

Trigger for attempting to adjust a link with known traffic load. Consider a mesh with known traffic load. Suppose that the load balancing algorithm has allocated load to links on the mesh, and let link $(a,b) = \arg\max_{x \in N_a^l} \rho(x)$. If replacing (a,b) with (a,x) would mean that there exists a cut through the mesh that traverses (a,x) and that all other links on that cut have a load $< \rho(a,b)$ then let node a initiate the link adjusting procedure. Likewise if replacing (a,b) with (y,b).

4.1 Reactive Logic

The relationship between the reactive and proactive logics is determined by:

if event [link α is broken] **then** $\{$
  activate [activate the *Reactive Method* for link α];
  $\forall x \in \alpha$ **if state** [node x locked by "**accept lock**$[a,x,s,t]$"
  then $\{$**transmit** "**reject lock**$[a,x,s,t]$";$\}$
$\}$

where the *Reactive Method* is as follows; it simply fixes disasters as they occur possibly with a configuration that is less satisfactory than the prior. It has no implications for neighbouring interfaces, and so it presents no instability issues.

Link adjustment with unknown traffic load. Suppose that node a has interference range S_a. Let M_a be the set of nodes in S_a excluding node a. For nodes $x, y \in S_a$, let $c(x,y)$ denote the cost[6] of the least cost path that connects x and y. We assume that: $(\forall x,y)c(x,y) = c(y,x)$, and that if the least cost path between nodes u and v is a subset of the least cost path between x and y then $c(u,v) \leq c(x,y)$. Let N_a^t be the set of links in S_a at time t, and $N_a^t(\ominus[a,x], \oplus[a,y])$ denotes the network configuration with link $[a,x]$ replaced by $[a,y]$. Let $C(N_a^t)$ denote the cost of the path of greatest cost in S_a: $C(N_a^t) \triangleq \max_{x,y \in S_a} c(x,y)$. Choose the pair of nodes b and c by:

$$(b,c) = \arg \min_{(x,y)|[a,x] \in N_a^t, y \in M_a} C(N_a^t(\ominus[a,x], \oplus[a,y]))$$

and swap link $[a,b]$ for link $[a,c]$ if:

$$C(N_a^t(\ominus[a,b], \oplus[a,c])) < C(N_a^t) \times \varepsilon$$

where $\varepsilon < 1$ is a threshold constant [11].

4.2 Second Path Reduction Algorithm

In our second path reduction algorithm only link substitution method is used which makes it different from our previous PR algorithm in which link substation is used along with link addition. Another distinction is that previously we selected only those substituted links that would not increase the IC whereas in the current algorithm we select links irrespective of their effect on the IC.

The outline of our algorithm is succinctly given here: An initiator node selects one of its available radio interfaces on the basis of strongest transmission power, the selected interface creates a list of available interfaces in its communication range (locality principle), from these interfaces those that have a path length to the portal node longer than the shortest path are filtered out. The remaining ones are short listed and an interface

[6] The precise meaning of this cost function does not matter. It could be simply the number of hops, or some more complex measure involving load and/or interference.

For node $n_x \in N$ select one of its interfaces i_x that is free and has the strongest signal of all its free interfaces;
if $i_x = \varnothing$
    for n_x set *blockFree* ← *bc;*
    end;
else
    set *blockBusy* ← *bc*;
set I_{free} ← *communicationRange*(i_x , *I*) ;
if $I_{free} = \varnothing$
    endAction.
set i_y ← *bestSNR*(I_{free})
if $i_y = \varnothing$
    endAction.
$l_{i_x i_y}$ ← *createLink*(i_x , i_y)
for n_x *linkCounter* *linkSubCounter* + 1;
remove ($l_{i_u i_v}$);
reset (i_u , i_v);
set p ← *newShortestPath*(n_x)
end.

N	a set of all nodes
I	a set of all interfaces
L	a set of all links
I_{free}	a subset of interfaces
i_x	an available interface.
n_x	a node which contains i_x.
i_y	an available interface.
n_y	a node which contains i_y.
i_u	an interface with shortest path on n_x
n_v	a node that contains i_v.
i_u	an interface on n_z that is linked to i_u
p	a shortest path for a n_x
$l_{i_u u_v}$	a link between i_u *and* i_v
bc	a blocking constant

endAction
    for n_x
        set *blockBusy* ← 0;
        set *blockFree* ← *bc;*
end.

Fig. 3. Second path reduction algorithm

from these that offers the best SNIR is selected; the new link between the two interfaces is created and the previous shortest path link is switched off and its interfaces operational attributes are reset to their default values. This process occurs simultaneously across the multiagent system. A more formal description of the algorithm is given in Figure 3. The functions in the algorithm are:

- cummuncationRange(interface, set of interfaces): This function selects all interfaces from the set of interfaces that are free and within the range of interface.
- shortestPath(p, interface, set of interfaces): This function selects a set of interfaces that has a shortest path from the interface to the set of interfaces. If the path between the interface with a shortest path and the interface is not shorter than path p the function returns ∅. Otherwise it returns a set of interfaces with a shortest path.
- bestSNIR(set of interfaces): This function selects an interface with a best SNIR from the set of interfaces. If there is more than one such interface this function randomly selects one.
- createLink(interface *A*, interface *B*): This function creates a link between agent *A* and agent *B* and returns a newly created link.
- remove(link): This function removes a link from the set *L*.
- reset(interface *A*, interface *B*): This function resets attributes of agent *A* and agent *B* to default attributes.
- newShortestPath(node): This function returns a shortest path value for the node.
- A node is either *locked* or *unlocked*. A locked node is either locked because it has committed to lock itself for a period of time on request from another node,

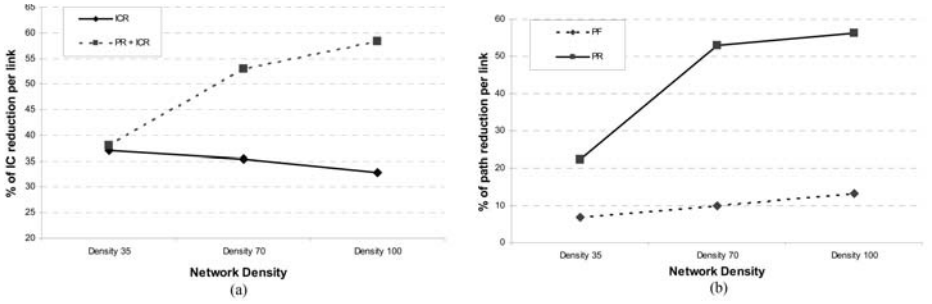

Fig. 4. (a) Result (comparative study) showing the increase in IC reduction in comparison to ICR only algorithm. (b) First versus the second algorithm for path reduction through link substitution.

or it is self-locked because it has recently instigated one of the self organisation procedures. A locked node is only locked for a "very short" period. This is simply to ensure that no more than one alteration is made during any one period which is necessary to ensure the stability of the procedures.

4.3 Results and Discussion

We present below some of the key results that we have obtained to illustrate the performance of the second path reduction algorithm. Figure 4(a) shows a graph of ICR vs. network density when just the algorithm for ICR has been invoked and both the path reduction and ICR algorithms has been invoked. This comparative study clearly shows that without the invocation of the path reduction algorithm the effect of ICR process results in much higher IC. A reason for this is twofold. Firstly, because of an increased number of interfaces that the initiator node can use and secondly, because of the additional mechanism that selects the interface based on the best SNIR value. Furthermore, as the network density increases the performance of the path reduction followed by ICR significantly increases whereas just the performance of the ICR algorithm on its own slightly decreases. Figure 4(b) compares the path length reduction that is achieved by using the first algorithm and the second path length reduction algorithm. The second algorithm is much more effective than the first by a factor of 2 to 5 times.

5 Conclusion and Future Work

We have described a multiagent system based self-organising algorithm for multi-radio wireless mesh networks (MR-WMN) that can operate on any radio technology. The multiagent system ensures scalability by progressively assigning the channels to nodes in clusters during the WMN system start up phase. The stability is offered by means of the proactive and reactive logic of the system. These attributes were validated through analysis and simulation. Through the work described in this report we have examined motivation and developed a multiagent system for the topological control of MR-WMN. The goal of this algorithm is to increase the number of shortest paths to the portal nodes

without adversely effecting interference cost. In addition to interference cost reduction implementation of this algorithm on MR-WMN further improve the system capacity.

Our future work will be focused on the development of our Java framework that is multi threaded so each node is represented as an independent thread. We believe that this will enable us to develop algorithms for tuning the capacity of the network links according to fluctuations in demand by mobile users.

References

1. Raniwala, A., Chiueh, T.c.: Architecture and Algorithms for an IEEE 802.11-based Multi-channel Wireless Mesh Network. In: Proceedings IEEE Infocom 2005. IEEE Computer Society, Los Alamitos (2005)
2. Ko, B.J., Misra, V., Padhye, J., Rubenstein, D.: Distributed Channel Assignment in Multi-Radio 802.11 Mesh Networks. Technical report, Columbia University (2006)
3. Mishra, A., Rozner, E., Banerjee, S., Arbaugh, W.: Exploiting partially overlapping channels in wireless networks: Turning a peril into an advantage. In: ACM/USENIX Internet Measurement Conference (2005)
4. Mishra, A., Shrivastava, V., Banerjee, S.: Partially Overlapped Channels Not Considered Harmful. In: SIGMetrics/Performance (2006)
5. Akyildiz, I.F., Wang, X., Wang, W.: Wireless mesh netowrks: a survey. Computer Networks, 445–487 (2005)
6. Vasudevan, S.: A Simulator for analyzing the throughput of IEEE 802.11b Wireless LAN Systems. Master's thesis, Virginia Polytechnic Institute and State University (2005)
7. Leith, D., Clifford, P.: A self-managed distributed channel selection algorithm for wlans. In: Proceedings of RAWNET, Boston, MA, USA, pp. 1–9 (2006)
8. Duffy, K., Malone, D., Leith, D.: Modeling the 802.11 Distributed Coordination Function in Non-saturated Conditions. IEEE Communication Letters 9, 715–717 (2005)
9. Tourrilhes, J.: Robust Broadcast: Improving the reliability of broadcast transmissions on CSMA/CA. In: Proceedings of PIMRC 1998, pp. 1111–1115 (1998)
10. Fischer, K., Schillo, M., Siekmann, J.: Holonic multiagent systems: The foundation for the organization of multiagent systems. In: Mařík, V., McFarlane, D.C., Valckenaers, P. (eds.) HoloMAS 2003. LNCS, vol. 2744, pp. 71–80. Springer, Heidelberg (2003)
11. Ramachandran, K., Belding, E., Almeroth, K., Buddhikot, M.: Interference-aware channel assignment in multi-radio wireless mesh networks. In: Proceedings of Infocom, Barcelona, Spain, pp. 1–12 (2006)