A Distributed Power Management Algorithm for a Self-optimizing WiFi Network

Abheek Saha
Hughes Systique Corp.
Gurgaon, India
Email:abheek.saha@hsc.com

***Abstract*—In this paper, we propose a fully distributed algorithm for a dense unplanned self-optimizing network of 802.11 access points. As opposed to the traditional method of having a centralized controller to collect information from all access points, our algorithm runs independently on each AP to create a cooperative network of access points, with no explicit inter-node communication. We present simulation and actual laboratory data which shows how our algorithm can significantly improve network performance in a robust and scalable manner.**

***Keywords*—Self optimizing networks; distributed algorithms; cooperative games; femtocells**

I. INTRODUCTION

A self optimizing network (SON) of wireless nodes represents a collection or network of co-located network nodes which can jointly set their own operating configuration so as to maximize network performance using suitably chosen parameters. SON controllers for WiFi Access Points (APs) have been launched by Aruba [1], Ruckus [2] and others. The CISCO suite of Wireless LAN (WLAN) applications includes the Cisco Unified Wireless Network module, which incorporates various SON features. In the LTE domain, SON has been included into the 3gPP specifications [3] and is being actively pursued by Nokia Networks, Ericsson, AT&T and others.

In a typical self-optimizing network, we are interested in optimizing some global performance metric, subject to constraints in another global metric, by controlling specific operational parameters at each individual node in the system. For example, a SON can try to minimize handovers in the network as a whole, subject to average call drop probability being above a certain threshold; both of these are global, user-visible metrices. The optimization is achieved by setting certain policy parameters in each network node; for the above example, it may be the Signal and Interference to Noise Ratio (SINR) threshold at which a handover is triggered.

In the standard self-optimizing networks as discussed in [4]–[7], the SON function resides in a central entity (or cluster of entities) called the controller. The controller receives feedback from the network nodes (base-stations, access points, or node B, depending on the Radio access technology) and in turn computes and sends configurations to the network nodes; hence translating global network state information to local control values for each network node. We call this the *central control* model.

The object of this paper is to propose an alternate, distributed model for self-optimizing algorithms in modern wireless systems. Our work is on the same lines as [4] [8] but our model and technique vary significantly. We take a fully distributed approach to our problem by borrowing elements from the most successful distributed control applications that we are aware of. The result is a simple, scalable algorithm, which works in a multitude of conditions. We have simulated this and implemented it in real life using a network of WiFi access points, by using the basic interface provided by the Linux *hostapd* application. With the recent release of the Femto API by the small cell forum, we believe that SON applications can be implemented very easily on the network cell nodes themselves. A fully distributed algorithm is a significant step to solving the problems of connectivity, scalability and robustness which limits centralized algorithms.

The rest of this paper is organized as follows. In Section II, we describe the details of transmit power control in a WiFi network. In Section III, we describe the problem to be solved in the context of the network topology and network node properties, for which our algorithm is presented. In Section IV, we describe the basis for the distributed algorithm and its adaptation for our particular problem. In Section V, we describe the design of the test-bed and the validation of the algorithm. Finally, in Section VI, we conclude our analysis and areas of future work.

II. SETTING TRANSMIT POWER IN A WIFI NETWORK

While there are many wireless network management functions which are suitable for SON applications, a commonly used one is interference and transmit power management. This single function covers a great deal of ground; by adjusting beacon power, we can adapt coverage, by adjusting traffic channel power, we can adjust SINR and throughput and finally, by a combination of the two, we can also adjust handoff performance.

A. The impact of transmit power setting

What are the implications of adjusting transmit power? A network node broadly transmits three kinds of waveforms. First is the beacon and/or pilot channel. This transmission is necessary for mobile devices in the idle state to detect the presence the network node and associate with it - thus, the pilot/beacon power controls the coverage or reach of the signal. This signal is also used by user terminals during handover as a way of identifying the relative strength of a network node compared to the one it is currently *camped on*. Pilots are typically at significantly higher power than the rest of the transmission, but occupy a relatively small part of the overall

 ISBN: 978-1-61208-279-0

power budget, because they occupy a small portion of the available spectrum.

The second power level is that used for transmission on the traffic channels; this is spread over embedded pilot, data and control bits. Typically, the embedded pilot bits are transmitted at a certain level above/below that of the traffic bits , since they have to be received reliably in order to do accurate channel estimation. A power budgeting function is used to allocate energy per bit so as to maintain these relative ratios as per network configuration. Control bits are usually protected through coding, rather than power setting and thus are at the same power level as data bits. The quality of the service that a given user terminal receives is a function of the SINR on the data channels (subject to the signal level itself being above the receiver sensitivity threshold; but this condition is very mild thanks to the astonishing sensitivity of modern receivers). As the SINR rises, the individual bit error rate (BER) drops, and consequently the user terminal can increase the transmission rate (by reducing protection and/or using more aggressive modulation schemes); this is known as adaptive modulation and coding. It should be noted, however, that the relationship between SINR and transmission rate is not linear. For example, at a given coding rate and scheme, the block error rate (BLER) is only affected by the SINR once the number of bits in error per block cross a certain threshold value known as the *free distance* of the coding scheme. While there are many studies on this topic, a log-linear curve has been shown to hold true in a number of cases [9].

III. ALGORITHM CONTEXT

We place our algorithm in the fairly generic context of a network of 'network nodes', 'mobile/user devices' and an allocated set of frequencies, which are the shared physical resource. The purpose of the network is to allocate frequencies to individual nodes, so as to offer the maximum quantum of service (measured both in terms of throughput and QoS) to active users. The number of frequencies available is typically fixed, whereas the network can scale indefinitely, both in terms of network nodes and in terms of actual users. Gupta and Kumar point out in [10] that in this kind of capacity limited, interference constrained network, the aggregate throughput can at best scale at $\sqrt{n}$, whereas the throughput per user will actually scale at $\frac{1}{\sqrt{n}}$. We wish to find an algorithm to achieve this target.

A. Properties of network components

We use the term network node to describe eNodeB/Access Points. The common characteristics of these nodes are:

- They transmit a mix of pilot and data signals on the downlink.
- The total power required for data traffic is determined by the mix of users they are supporting and their distances from the node itself
- The network nodes use a fair allocation scheduler, so that the traffic transmission is not dominated by the best placed user devices, but the average.
- The network node is aware of its own position.
- The utility of the network node is determined by the average bit-rate it supports and the average number of active users.
- The network node has a transmit power level u_i which allocates the combined power for each resource it is using. It is power-limited in the sense that the value u_i must be less than some maximum power level P_u. This comparision may be done instantaneously, i.e., $u_i(t) \leq P_u \forall t$ or averaged over a slot or a frame. A power allocation of zero for a given network node indicates that the node is switched off.
- The network node has fine grained control over its transmit power level, i.e., it can adjust transmit power on a frame by frame basis [11] and allocate variable power to control and data frames; however, it has also been pointed out that this has limitations based on the technology and realization thereof [12]. The network node is free to manage the individual allocations to frames/users/channels as long as the overall power budget is maintained.

We further make the following reasonable assumptions about the topology

- The network nodes are randomly placed in a 'dense' environment, i.e., theoretically, any of the clients may attach to any of the nodes
- Each network node is assigned one channel on startup; more channels can be allocated depending on availability

Each network node has a set of users (WiFi clients) to whom they are offering data transport services. The user population is assumed to have the following general principles

- The user devices are located randomly in the area; for simplicity's sake a given user device can pick absolute any network node in this area, subject to operational constraints, i.e., sufficient RSSI and SINR
- The user devices are homogenous; they are characterized by a data generating process with common properties

We say that a network node is *backlogged* if at least one of its users is continuously backlogged, i.e., has a non-zero queue for the duration of the measurement T. In practical terms, if a network node has no backlogged users then it should reduce its traffic load, because it is possibly offering more service than its users are demanding. Given the set of backlogged users, we assume that the network computes a target SINR value τ_i which represents the ensemble of backlogged users. The utility for the network is given by the function

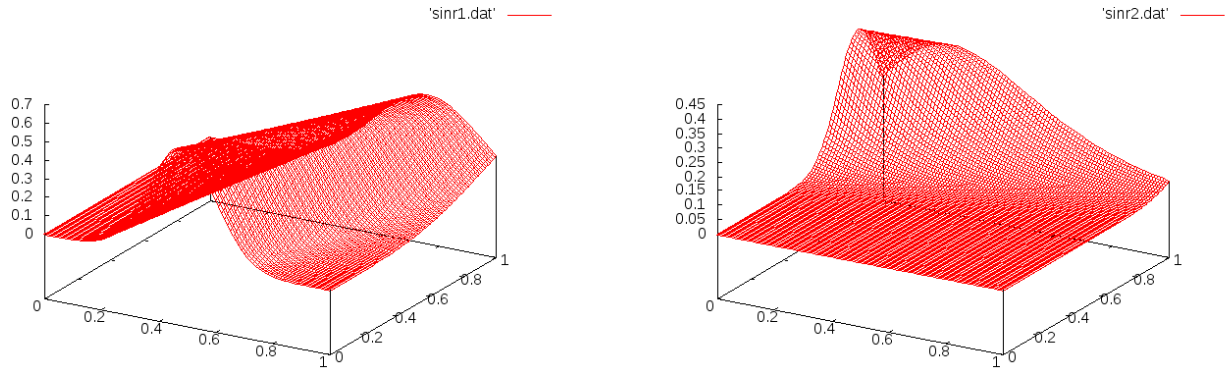
$$U(u_i) = \mathbb{E}_k e^{-\left[(s_{i,k}-\tau_{i,k})^2\right]} \quad (1)$$

As we can see in Fig. 1, the utility function for a given network node is quasi-concave.

Each network node transmits at a level $u_i(t)$, which is the control variable for our algorithm. The signal received by the kth user of the ith network node thus becomes

$$S_{k,i} = \frac{u_i F(d_{k,i,i})}{\sigma_i + \sum_{j \neq i} \gamma_{i,j} u_j F(d_{k,i,j})} \quad (2)$$

 ISBN: 978-1-61208-279-0



(a) AP 1 (b) AP 2

Figure 1. Utility functions for two access points

where $d_{k,i,j}$ is the distance between the kth user of the ith Access Point from the jth access point and $F()$ represents the fading. $\gamma_{a,b}$ is the interference coupling between access points a and b. If the two are using orthogonal frequencies, it is 0, if the two are using the same frequency, it becomes 1. For a given network node, we assume the existence of a metric function $G(i) = g(S_{1,i}, S_{2,i}, \ldots)$, which is monotonic in each of its elements. A simple metric function is the average or infimum.

B. Formulation of the standard problem

We start with the easiest form of the problem. Assume that $\alpha_{i,j}$ represents the marginal impact of increasing transmit power u_j for the ith Access Point. Therefore $\alpha_{i,j} = \mathbb{E}_k(\gamma_{i,j}F(d_{k,i,j}))$. The formulation in (2) can be rewritten as:

$$S_i = \mathbb{E}_k(S_i) = \frac{\alpha_{i,i}u_i}{n_i + \sum_{j\neq i}\alpha_{i,j}u_j} = u_i\kappa_i \quad (3)$$

For an equilibrum solution, we need to solve $\frac{\partial U_i(u_i)}{\partial u_i} = 0 \forall 0 \leq i < n$. To do this, we note that

$$\frac{\partial U_i}{\partial u_i} = \frac{\partial U_i}{\partial s_i}\frac{\partial s_i}{\partial u_i}$$

$$= -2.0(s_i - \tau_i)e^{-(s_i-\tau_i)^2}\frac{\alpha_{i,i}}{n_i + \sum_{j\neq i}\alpha_{i,j}u_i} \quad (4)$$

(5)

This has one solution where $s_i = \tau_i \forall i$. We can write the resultant set of equations as a matrix equation

$$\begin{bmatrix} u_0 \\ u_1 \\ \cdots \\ u_{n-1} \end{bmatrix} \begin{bmatrix} \alpha_{0,0} & -\tau_0\alpha_{0,1} & \cdots & -\tau_0\alpha_{0,n-1} \\ -\tau_1\alpha_{1,0} & \alpha_{1,1} & \cdots & -\tau_1\alpha_{1,n-1} \\ \cdots & \cdots & \cdots & \cdots \\ -\tau_n\alpha_{n-1,0} & \tau_{n-1}\alpha_{n-1,1} & \cdots & \alpha_{1,n-1} \end{bmatrix}$$

$$= \begin{bmatrix} n_0\tau_0 \\ n_1\tau_1 \\ \cdots \\ n_{n-1}\tau_{n-1} \end{bmatrix}$$

The optimization problem as formulated in (6) is solvable under some basic conditions; specifically, the matrix has to be invertible. However, this solution requires the solver to have full state information of p_i and s_i for all network nodes. This can be acquired by querying individual network nodes, but requires a robust communication channel, scaling with the number of access points. A bigger challenge is to compute the values of $\alpha_{0\leq i<n, 0\leq j<n}$ and the values of the individual noise terms. In fact, there is no possible way to compute n, the noise terms and the values of $\alpha_{i,j}$, even under the fairly reasonable assumption that $\alpha_{i,j} = \alpha_{j,i}$, since we have a total of $2 * n$ readings and $n_{C_2} + n^2 + n$ unknowns. Most of the available solutions assume that the values of the noise term are known by external means [5].

Is it possible to compute the value of $\alpha_{i,j}$ using external means? Note that $\alpha_{i,j}$ consists of two terms, one being the attenuation caused by distance and the second being the coupling. For the latter we need to know the channels allocated to each node and how much ACI/CCI is being generated - this is relatively easy to get. However for the former, we need a channel model which captures the relationship between geographical position and the attenuation and other channel model parameters specific to that particular environment and geography. OFDM traffic models are complex and sensitive to environmental artefacts and this typically requires a lot of intervention from the user/administrator. In commercial models, the relative coupling between nodes are typically embodied in the ANR and are based on a combination of channel modeling and continuous measurements.

A second challenge is to compute a value of τ_i for each access point. Clearly, an excessively low value of τ will cause bad service, whereas an excessively high value of $\vec{\tau} = \{\tau_0, \tau_1, \ldots, \tau_{n-1}\}$ means that the matrix given in (6) is no longer positive definite, hence the program isn't solvable. In a centralized environment, we will be able to set $\vec{\tau}$ to the maximum set of values so as to allow a solution to (6). This is a semidefinite programming problem and is not very easy to solve either. Alternate game theoretic formulations have the

 ISBN: 978-1-61208-279-0

same results and similar issues.

We would thus like to avoid any solution which depends on static analysis or pre-configuration to compute the above values; ideally, our algorithm would be an iterative solution, capable of using empirical measurements to fine tune its estimate of the network state. If we can design an algorithm that only uses empirical measurements from individual network nodes, it lends itself to distributed implementation. In our experience, a distributed algorithm will typically score over any centralized solution in terms of scalability (the ability to handle larger and larger numbers of nodes), robustness and flexibility. The classic paper by Kleinrock and Tung [13] demonstrates a elegant and remarkable solution to what seems to be an intractable optimization problem. On the other hand, distributed algorithms need to be carefully designed so as to be stable and converge sufficiently quickly to the appropriate solution. Convergence time for different categories of distributed algorithms is an enormous research topic in its own right.

IV. A DISTRIBUTED ALGORITHM FOR POWER CONTROL

We propose a distributed power control algorithm loosely modeled on the well-studied TCP congestion control algorithm. There are some crucial similarities and some crucial differences, as we shall demonstrate in the subsequent sections.

A. TCP congestion control as a distributed optimization algorithm

The TCP congestion control algorithm is a extremely well studied and widely deployed example of network wide congestion control [14]. It has been extremely succesful in practice, and has benefited from many years of continuous refinement and innovation (TCP New Reno, TCP Vegas, TCP Westwood, Eifel and TCP CUBIC). Its strengths lie it its flexibility (starting from switched links of 56kb/s, it has been deployed in every concievable environment, including wireless, gigabit Ethernet, terabit optical fiber and satellite), its scalability (literally thousands of individual nodes and tens of thousands of connections) and its robustness. It is also easy to deploy, requiring practically no intervention or configuration from the user. Based on our own extensive experience of TCP, we find its success arises from three factors.

- It constantly switches between stability and network probing. This ensures that it never enters a sub-optimal local equilibrium and can also automatically adjust to changes in global conditions (network load, network bandwidth availability, etc.)
- It acts aggressively before it discovers congestion and conservatively on discovering congestion (fast attack/slow retreat). This allows new TCPs to enter and adjust rapidly to existent network conditions
- It constantly updates its estimates of the two critical parameters in the algorithm; the round trip latency (to avoid loop gain) and the congestion buffer size. There are no fixed or externally programmed thresholds.

TCP works by transmitting a number of packets into the network destined to a peer TCP and waiting for a set amount of time. If the packets are positively acknowledged within that time, it assumes that there is bandwidth/buffering available in the network and transmits a few more packets, this time a slightly higher number. If the pre-set time expires, it assumes that the transmitted packets were lost due to congestion in the network. So it reduces its transmission rate and tries resending the packets. Each TCP computes its own estimate of the network measures independently and applies its congestion algorithm independently of the others in the same shared network. The implementation is completely distributed; no TCP needs to exchange any information with any other TCP. Yet the system performs stably and scales, due to the features built into the algorithm itself.

B. A brief review of TCP

Leaving aside *congestion avoidance* for the moment, there are two variables which govern a particular TCP instance. The first is the transmission rate, measured by the number of packets to be injected into network per roundtrip time. TCPs call this variable *cwnd*; it is the internal estimate of the amount of buffering available in the network. The higher the *cwnd*, the faster the TCP transmits. The second is the estimate of roundtrip time, which is a statistical measure $\hat{rtt} = \mu_{rtt} + 2\sigma_{rtt}$, constantly updated by the measured timegap between a packet being transmitted and the acknowledgement being received for it. The variables represent the tension in the system; if the TCP underestimates the round trip time, it will time out prematurely and inject unnecessary packets into the network, adding to congestion. If the TCP over-estimates the round trip time, its reaction to network conditions and ability to recover from congestion is diminished.

There is a further purpose to the round trip measurement; it allows the TCP to estimate a *baseline* round trip time, below which the TCP cannot possibly expect feedback. This is the base latency of the network, the actual time taken for a packet to traverse the empty network. In addition, a component of the baseline rtt is the latency induced by background network traffic; network traffic which is independent of the actions of this TCP. TCP algorithms such as VEGAS [15] use variations of the measured round trip time, in conjunction to the knowledge about the baseline delay to detect congestion.

Further, the TCP congestion control algorithm uses *Additive Increase and Multiplicative Decrease*. Even though multiple TCPs may be transmitting in parallel, each TCP reacts to a congestion signal as if it was the sole contributor for this congestion and executes a multiplicative decrease of its traffic rate; this bypasses the distributed coordination issue. Empirical data shows that this conservatism is as the heart of TCP's success in maintaining network stability.

It is important to understand the parallels between TCP and our situation. As in TCP, we also have the loop time, which is basically the time taken for the rest of the network to detect and respond to any unilateral changes in power; its value depends on the frequency of measurement updates and the averaging period for the other Access Points in the network. The analogue to the congestion window is of course, the transmit power setting of the network node.

 ISBN: 978-1-61208-279-0

C. A local algorithm for power adjustment

The algorithm that we have devised is a mixture of the approaches suggested in the various TCP algorithms. As in NewReno, we use a congestion signal for backing off transmission power; in our case, the congestion event occurs when the measured noise is greater than the baseline noise by a fixed Δ amount. Like New Reno, we use 'bandwidth hunting', by constantly trying to adjust transmit power onwards. We have used some of the ideas in TCP CuBic to make the algorithm self timing.

There is an important issue to be solved, however, which is not addressed by New Reno. This is the problem of background noise - noise which is independent of the network nodes transmission levels or those of its neighbours. In our New Reno analogy, this is akin to having an additional amount of packet drops which operate independent of network node induced congestion. The assumption in New Reno is that all packet drops are caused by congestion - a valid assumption in normal wired networks, but one which causes some issues in high delay wireless networks [16].

In our case, baseline noise exists and it varies over time and networks. A network node, on entering the system, has to be able to detect baseline noise dynamically and adjust to it. An analogy exists in TCP Vegas, which works by mapping congestion to variations in round-trip delay. For this to work well, the Vegas algorithm needs to be able to estimate the base delay in the network. We adapt the same to our system, except we measure baseline noise. The measurement of baseline noise has an interesting ramification - specifically, the higher the estimate of baseline noise, the more aggressive an endpoint is going to be. This gives an additional incentive for network nodes to minimize their transmission power ; otherwise, network nodes which enter (or re-enter) the system are going to over-estimate the baseline noise initially and act more aggressively. This effect has been noted in Mo et al [17].

Secondly, there is no directly analogue of a utility function for a TCP connection - at least, one which is explicitly built into the algorithm. Rather a TCP connection attempts to maximize throughput, subject to a complex set of rules. However, there have been many studies of what utility function TCP is *effectively* using [18], [19]. It is obvious, that TCP pays more attention to the bandwidth delay product, than pure bandwidth; however, packet drops also impact it. In our case, we are attempting to maximize an explicit utility function $U(., \tau_i)$, which is driven by our set point SINR target.

In the following, u_i is the transmit power of the ith network node, and $r_{i,k}$ and $s_{i,k}$ are the RSSI and SINR reported by the ith backlogged mobile. We have chosen these two metrices because they are directly available from most existing WiFi chipsets; in some cases SINR is replaced by the channel quality, which can, however be converted back to SINR units. The noise as measured by the node is given by $N_i = \frac{\sum_k s_{i,k} - r_{i,k}}{\sum_k I_{s_{i,k} > \tau_i}}$ The state transition diagram is given in I. The values ∇, T_s are user provided and can be adjusted depending on the type of the network. As can be seen here, the states *Stable*, *Ramping* and *Backoff* are rough analogues of the different phases of the TCP connection. A separate procedure is implemented for measuring the 'background' noise threshold; in our case, we simply measured the average signal energy of the system when the AP wasn't transmitting. However, specific air interfaces may support direct noise measurement; in defined guard periods, for example, which provide a good approximation of the current system noise.

V. Design and verification

A. Simulation results

Initially, the algorithm was verified in a customized simulation environment. The simulation setup allows us to test the algorithm with a very large number of network node and UE combinations. N network nodes cater to U user devices placed randomly in a fixed area; each network node has a dynamic transmission range of 1dBW to 20dbW and a startup transmit power of 10dbW. User devices attach to the network nodes using measured RSSI and then receive downlink data. The data transmission rates are derived from the measured SINR, so as to cause a BER of 0.01%. Each user equipment receives a random amount of data drawn from a Pareto distribution. Once the data queue goes idle for a user, it can select a new device to campon using either RSSI measurements or a combination of load and SiNR measurements. The setpoint τ_i is configured externally.

The graph in Fig. 2 below shows a comparision of the same system for different combinations of N and U; in one case active interference management is being carried out using the algorithm described above and in the other, there is no interference management. Each line shows the percentage improvement in average throughput for a given number of network nodes (depicted by the variable *Nn*) for the SON algorithm, versus the baseline average throughput when no algorithm is used. The X-axis shows the number of user terminals (distributed randomly over a unit square space) for a given simulation with a particular value of *Nn*. The Y-axis shows the percentage difference in throughput. We can see that the interference management algorithm easily outperforms the baseline when the system is relatively uncrowded and gradually degenerates to the baseline performance as the number of user devices per network node increases.

B. Real life results

We subsequently have implemented and tested our algorithm on a laboratory setup comprising of 8 access points and 16 Wifi clients, all operating on a single WiFi channel in a closed and sanitized laboratory environment. The WiFi network nodes were Linux workstations with attached WiFi cards; the SON algorithm was implemented in an application which controlled the WiFi driver (for power measurements and power settings) using the standard Linux interface. The user terminals were a mixture of commercially available laptops and WiFi enabled mobile phones; the test was carried out in a controlled lab atmosphere and loading was generated using a mixture of artificially generated traffic and smartphone applications, such as web-browsing and game apps. The overall network was monitored using a mixture of inhouse tools and a commercial tool called Ekahau, which dynamically measures network RSSI and SINR from multiple positions.

The results are captured in the graphs in Fig. 3a and Fig. 3b. Since the network setup was fairly dense and the APs were power capped, the average throughput for APs in the

 ISBN: 978-1-61208-279-0

TABLE I. Congestion management - state/event matrix

Current State	Measured event	Action	New State
Stable	Noise measurement is stable or reducing for T_s time periods	tx pwr increases by ∇	Ramping
Stable	Noise measurement is crosses threshold	Reduce tx power to last known stable value	Backoff
Stable	Noise baseline has changed	Adjust tx power accordingly	Stable
Backoff	Noise measurement is stable or reducing for T_s time periods	Save current tx power as last stable tx power	Stable
Ramping	Noise measurement changes by less than ∇	Adjust tx power	Stable
Ramping	Noise threshold changes	Adjust tx power	Stable
Ramping	Noise measurement crosses threshold	Reduce tx power to last known stable tx pwr	Backoff

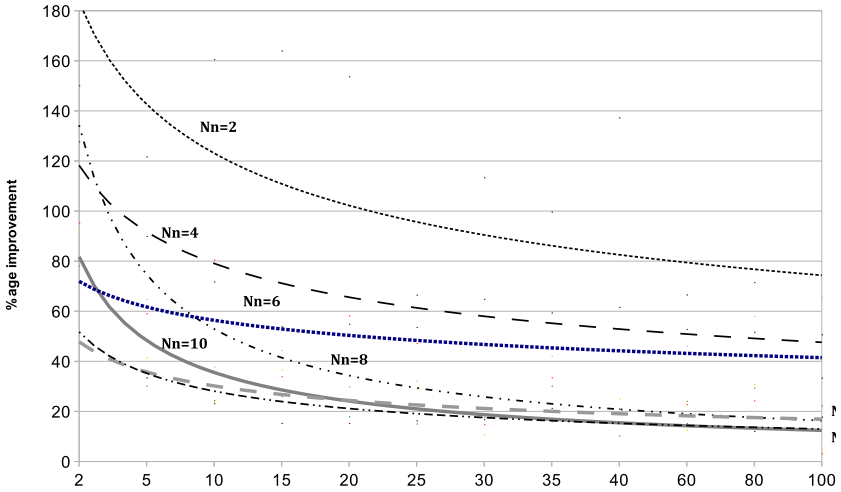
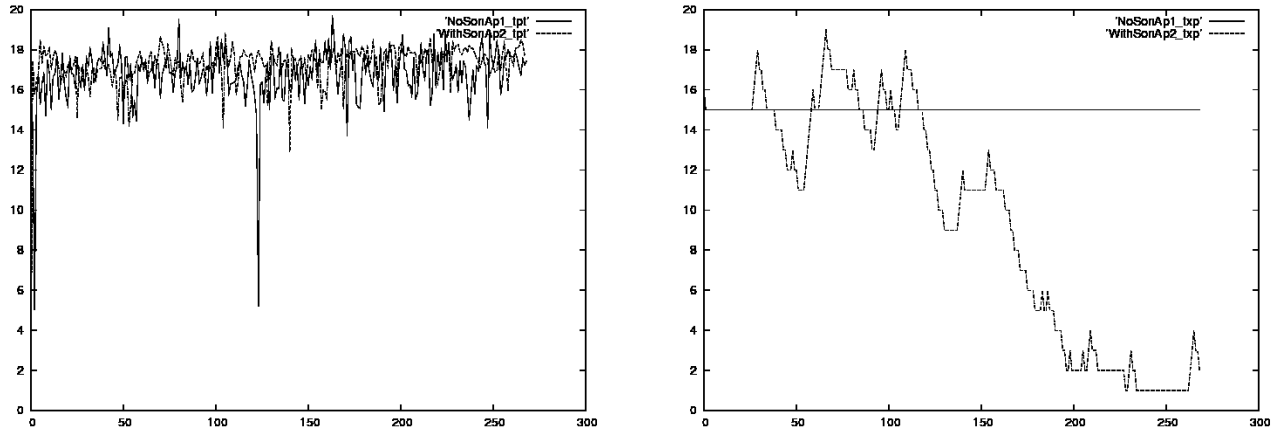


Figure 2. Simulation output; throughput improvement using distributed SON algorithm



(a) AP 1 (b) AP 2

Figure 3. Real life - setup and results

SON and no SON environment were very similar. However, the APs in the network with SON active ran at *substantially* lower transmit power, a full 3-6 dB below the ones with no SON; even though both sets of APs started at the exact same transmit power settings. This arises from the conservative behaviour of the SON algorithm (borrowed from the original TCP). As the number of APs are reduced, the contention drops and individual APs start scaling their power adaptively, leading to substantial improvements in average throughput.

A further observation was that the set point τ_i plays a very important role in the stability of the algorithm. A high value of τ_i actually acts as a damping factor, because all network nodes tend to converge slowly towards this. On the other hand, a low set point allows network nodes to be more responsive to load conditions (since all network nodes achieve the set point easily), at the cost of network fairness;

 ISBN: 978-1-61208-279-0

there is substantial variations in the mean throughput achieved by different network nodes. A second metric is the relative percentage of time a network node spends in stable state; it can be seen that there are substantial variations in this when τ_i is reduced and it reduces as the value of τ_i increases.

VI. Conclusion and Future Work

In this paper, we have formulated a simple, yet robust and generic distributed algorithm for network wide optimization of transmit power levels for a self-optimizing network of WLAN APs. The principles can, in general, be translated to other networks such as LTE - in fact, any network which is power limited can use the algorithm. Simulation and real life results show that our algorithm provides significant improvements over the unmanaged network.

Our laboratory experiments show that the algorithm is adaptive to network conditions, but can lead to unstable equilibria at times. It also suffers from the reverse of the Stackelberg phenomenon; late starting access points tend to drive the equilibrium of the algorithm away from an optimal equilibrium. Future work focuses on mechanisms to limit this, perhaps using supervisory mechanisms.

References

[1] "Aruba adaptive network management," http://www.arubanetworks.com/products/arubaos/adaptive-radio-management, [Accessed January 12, 2013].

[2] "Smartmesh networking," http://www.ruckuswireless.com/technology/smartmesh, [Accessed January 12, 2013].

[3] W. G. 3, ""3rd generation partnership project: Technical report: Self-configuring and self-optimizing network: Use cases and solutions"," Tech. Rep., March 2003.

[4] N. Ahmed and S. Keshav, "Smarta: a self-managing architecture for thin access points," in *Proceedings of the 2006 ACM CoNEXT conference*, ser. CoNEXT '06. New York, NY, USA: ACM, 2006, pp. 9:1–9:12. [Online]. Available: http://doi.acm.org/10.1145/1368436.1368449

[5] I. Viering, M. Dottling, and A. Lobinger, "A mathematical perspective of self-optimizing wireless networks," in *Communications, 2009. ICC'09. IEEE International Conference On*, June 2009, pp. 1–6.

[6] B. Kauffmann, F. Baccelli, A. Chaintreau, K. Papagiannaki, and C. Diot, "Self Organization of Interfering 802.11 Wireless Access Networks," INRIA, Rapport de recherche RR-5649, 2005. [Online]. Available: http://hal.inria.fr/inria-00070360

[7] S. Bhaumik, G. Narlikar, S. Chattopadhyay, and S. Kanugovi, "Breathe to stay cool: adjusting cell sizes to reduce energy consumption," in *Proceedings of the first ACM SIGCOMM workshop on Green networking*, ser. Green Networking '10. New York, NY, USA: ACM, 2010, pp. 41–46. [Online]. Available: http://doi.acm.org/10.1145/1851290.1851300

[8] T. Moscibroda, R. Chandra, Y. Wu, S. Sengupta, P. Bahl, and Y. Yuan, "Load-aware spectrum distribution in wireless lans," in *Network Protocols, 2008. ICNP 2008. IEEE International Conference on*, vol. 3, no. 3, October 2008, pp. 137 –146.

[9] D. Divsalar, "A simple tight bound on error probability of block codes with application to turbo codes," *Journal of Programming Languages*, November 1999.

[10] P. Gupta and P. Kumar, "The capacity of wireless networks," *Information Theory, IEEE Transactions on*, vol. 46, no. 2, pp. 388 – 404, March 2000.

[11] D. Qiao, S. Choi, and K. G. Shin, "Interference analysis and transmit power control in ieee 802.11a/h wireless lans," *IEEE/ACM Transactions on Networking*, vol. 15, no. 5, pp. 1007–1020, October 2007.

[12] K. Kowalik, M. Bykowski, K. B, and D. M, "Practical issues of power control in ieee 802.11 wireless devices," in *IEEE International Conference on Telecommunications (ICT 2008), Proceedings of*, June 2008.

[13] L. Kleinrock and B. Tung, "Distributed control methods," in *Proceedings of the 2nd International Conference on High Performance Distributed Computing*, July 1993, pp. 206 – 215.

[14] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behaviour of the tcp congestion avoidance algorithm," *SIGCOMM Computer Communication Review*, vol. 27, no. 3, July 1997.

[15] L. Brakmo and L. Peterson, "Tcp vegas: end to end congestion avoidance on a global internet," *Selected Areas in Communication, IEEE Journal on*, vol. 13, no. 8, pp. 1465–1480, October 1995.

[16] H. Balakrishnan, V. PAdmanabhan, S. Seshan, and R. Katz, "A comparision of mechanisms for improving tcp performance over wireless links," *Networking, IEEE/ACM Transactions on*, vol. 5, no. 6, pp. 756–769, December 1997.

[17] J. Mo, R. La, V. Ananthanram, and J. Walrand, "Analysis and comparision of tcp reno and vegas," in *IEEE Infocom, 99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, March 1999, pp. 1556–1563.

[18] S. H. Low, "A duality model of tcp and queue management algorithms," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, August 2003.

[19] S. Kunniyur and R. Srikant, "End to end congestion control schemes: Utility functions, random drops and ecn marks," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, October 2003.

 ISBN: 978-1-61208-279-0