Dyson: An Architecture for Extensible Wireless LANs

Rohan Murty[†], Jitendra Padhye[‡], Alec Wolman[‡], Matt Welsh[†]
[†]Harvard University,[‡]Microsoft Research

1. INTRODUCTION

Although wireless LANs have evolved over time, including significant improvements to the PHY and MAC layers, one critical aspect of the WLAN architecture that has remained unchanged is the interface between clients and access points. Each node makes independent decisions about AP associations, PHY data rates, transmission power level, and other parameters that have a substantial impact on overall network performance. These decisions are typically based on *local* observations of the network state, with no explicit coordination between nodes. Further complicating the matter, different vendors introduce very different policies in their WLAN firmware and software, as this is viewed as an opportunity for innovation and competition. However, wresting control over the network is difficult given this "every station for itself" mentality.

The current trend in commercial WLANs is moving toward the use of a central controller that manages access points across an enterprise [1,2]. However, this approach does not incorporate the perspective of the *clients* in the network. We argue that effective network management must involve observation and control in a holistic manner, involving both APs and clients. While some research systems [4, 6, 8] and standardization efforts [5] try to address this issue, they are hamstrung by limitations of the 802.11 design.

In this poster, we present *Dyson*, a new wireless network architecture that is designed to support *global network observation* and *historical knowledge*, *deep control*, and *extensibility* to meet future needs. Put together, these features provide the controller with extensive visibility into the network's state, including information that can only be gleaned from clients, such as the presence of hidden terminals and signal strength from multiple APs. Dyson's control framework can yield better overall network efficiency, such as optimizing client/AP associations using knowledge of channel utilization and interference. Finally, Dyson enables a high degree of extensibility of the network's policies, making it easy to customize behavior, such as by providing user-specific airtime reservation, or shifting VoIP clients to a separate channel to avoid interference. In an earlier position paper [7] we outlined the concept and various potential benefits one may derive from it. This poster describes the system we have been building since then, and our preliminary experiences with it. We have considerably expanded the architecture, fleshed out the API for extensibility, and developed policies for managing infrastructure wireless networks.

2. DYSON ARCHITECTURE

The Dyson network architecture, shown in Figure 1, consists of a number of wireless *clients*, *access points* (APs), and a single *central controller* (CC). As described below, both APs and clients report measurements to the infrastructure, which are used to construct a dynamic *network map* representing the state of the network. Measurements are also logged to a database for historical analysis, and static information on AP location and MAC addresses are stored in a separate AP database. A set of administrator-defined *policies* are used to trigger network configuration changes via *commands* delivered to APs and clients by the CC. These commands form the core of the Dyson framework and permit the CC to control every AP and client in the system.

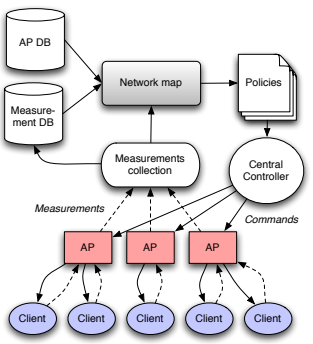

Figure 1: The Dyson network architecture.

Dyson builds upon existing 802.11 standards, including CSMA MAC and the format of the data and management frames. As a result, Dyson can be implemented entirely using existing 802.11-compatible hardware. The key difference between Dyson and existing enterprise WLANs is the manner in which network management and control is performed. The Dyson architecture requires both clients and APs to be Dyson-aware. However, a Dyson-enabled client can operate both in 802.11 and Dyson modes.

Dyson significantly augments the standard CC design by extending both observation and control to the wireless clients as well as the APs. Dyson clients are responsible for collecting periodic *measurements* of channel and traffic conditions and reporting them to the CC, as well as responding to *commands* from the CC that control many aspects of transmission parameters, as described below.

A key question that arises in this regime is how much control clients should yield to the infrastructure. At one extreme, the CC could control clients at a very fine-grained level, for example, by dictating individual packet transmission timings. However, this design would require substantial control overhead, and would fail to respond rapidly to local changes in channel conditions (e.g., interference) at the client. In Dyson, we opt to affect control at a

higher level, namely that of channel allocations, client-AP associations and throttling. Although cruder than packet-level control, this design strikes a balance between the overhead for command issue and the ability of the network to drive towards more efficient configurations. Due to lack of space we omit details regarding the exact API we use in the system. The power of the CC derives from its global knowledge of the state of the network and ability to control both APs and clients at fine granularity. The CC also maintains a database to store received measurements, permitting long-term historical analysis of network performance.

A key benefit in Dyson is the ability to collect *client-side* measurements, providing the CC with greater visibility and control over the network state. Client-side information can be used to resolve sources of ambiguity that would arise with AP-only observations. Examples include detection of hidden terminals, awareness of mutual connectivity between APs and clients, and mapping channel airtime utilization. While client participation has been explored by several previous systems, such as MDG [4] and SMARTA [3], Dyson provides a flexible framework in which a wide range of policies can be specified programmatically.

3. IMPLEMENTATION AND POLICIES

We have implemented a prototype of the Dyson architecture using the ALIX 2c2 single-board computer (500 MHz AMD Geode processor with 256 MB DRAM) running FreeBSD 7, coupled with dual CM 9 Atheros-based 802.11a/b/g radios. We have deployed a testbed to 30 nodes across one floor of an academic office building. Each node is connected to an Ethernet network for control. The CC is implemented on a separate machine running FreeBSD with 2 GB of RAM. We run most experiments using 802.11a to avoid interference with existing 802.11b/g networks in the building.

To support Dyson, it was necessary to modify the FreeBSD Atheros driver to add support for statistics collection and the Dyson command API, as well as to disable local autorating. Each node runs a Python-based daemon that exposes the Dyson measurements and command API via an XML-RPC interface, and communicates with the modified Atheros driver through *ioctl* calls. The CC is also implemented in Python; policies are loaded as Python modules at startup.

To illustrate the power of the Dyson framework, we outline six separate policies that demonstrate the key facets of our design. In particular, we present the following policies implemented using Dyson's policy API:

- **Capacity aware associations:** that assigns clients to APs based on airtime availability [6].
- **Interference mitigation:** using connectivity information from clients and APs.
- **VoIP-aware handoffs:** that causes VoIP and bulk clients to be assigned to different APs.
- **User-specific airtime reservations:** that gives priority to certain clients over others.
- **Uplink/downlink load balancing:** to mitigate the impact of bulk upload clients on throughput fairness.
- **Predicting handoffs:** based on historical knowledge of a user's mobility patterns.

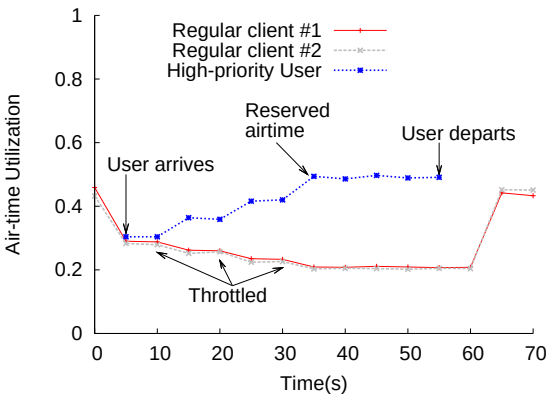

Figure 2: A time series plot of the airtime reservation policy in action. The setup consisted of two clients in close proximity associated with the same AP. The high-priority user arrives shortly before 5s and departs at 55s. The throttle for regular users is released once the high-priority user departs.

We present a preliminary result from one of the policies mentioned above, the user-specific airtime reservation policy. The setup consisted of an AP and two clients placed in close proximity of each other. These are deemed to be regular clients. We then introduce a third high-priority client with an airtime reservation target of 50%. All clients in the system were performing downloads using *iperf*. For the purposes of exposition, the policy runs every 10 sec at the CC. As seen in Figure 2, the policy correctly detects the high-priority user has not received the requisite share of airtime and throttles the regular users to compensate.

4. ON-GOING AND FUTURE WORK

We are currently evaluating a host of policies on testbed which is embedded in a realistic setting consisting of consists of hallways, lecture rooms, conference rooms, offices, and common areas, each of which see different wireless usage patterns during the day. In particular, we are performing experiments to show (a) the extensibility offered by Dyson in deploying new policies and (b) the efficacy of combining global knowledge and deep control over clients.

5. REFERENCES

[1] Enterprise Solutions from Aruba Networks, http://www.arubanetworks.com/solutions/enterprise.php.

[2] Meru Networks, The Virtual Cell Architecture,.

[3] N. Ahmed and S. Keshav. SMARTA: A Self-Managing Architecture for Thin Access Points. In *CoNEXT*, 2006.

[4] I. Broustis, K. Papagiannaki, S. V. Krishnamurthy, M. Faloutsos, and V. Mhatre. MDG: Measurement-driven Guidelines for 802.11 WLAN Design. In *MobiCom*, 2007.

[5] IEEE. *IEEE 802.11k-2008 — Amendment 1: Radio Resource Measurement of Wireless LANs*. June 2008.

[6] R. Murty, J. Padhye, R. Chandra, A. Wolman, and B. Zill. Designing High-Performance Enterprise Wireless Networks. In *NSDI*, San Francisco, CA, April 2008.

[7] R. Murty, J. Padhye, A. Wolman, and M. Welsh. An Extensible Wireless Network Architecture. In *HotNets*, 2008.

[8] M. Vutukuru, K. Jamieson, and H. Balakrishnan. Harnessing Exposed Terminals in Wireless Networks. In *NSDI*, 2008.