All Your Network Are Belong to Us: A Transport Framework for Mobile Network Selection

Shuo Deng, Anirudh Sivaraman, and Hari Balakrishnan
MIT Computer Science and Artificial Intelligence Lab
Cambridge, Massachusetts, U.S.A.
{shuodeng, anirudh, hari}@csail.mit.edu

ABSTRACT

Mobile devices come with an assortment of networks: WiFi in two different frequency bands, each of which can run in infrastructure-mode, WiFi-Direct mode, or ad hoc mode; cellular radios, which can run in LTE/4G, 3G, or EDGE modes; and Bluetooth. But how should an app choose which network to use? There is no systematic solution to this problem today: in current practice the choice is almost always left to the user, who usually has no idea what's best. In fact, what's best for a user depends on the app's performance objectives (throughput, delay, object load time, etc.) and the user's constraints on cost and battery life. Besides, what's best for a single user or app must be balanced with what's best for the wireless network as a whole (individual optimality vs. social utility). This paper introduces Delphi, a transport-layer module to resolve these issues. Delphi has three noteworthy components: "local learning", in which a mobile device estimates or infers useful properties of different networks efficiently, "property sharing", in which mobile devices share what they learn with other nearby devices, and "selection", in which each node selects a network using what it has observed locally and/or from its neighbors.

CATEGORIES AND SUBJECT DESCRIPTORS

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network Management*

KEYWORDS

Multi-Network, Mobile Device

1. INTRODUCTION

Today's mobile devices come equipped with a variety of networks, but without any systematic framework that allows apps to use the best network that fits their needs. For example, it is all too common to see a "smart" phone associate with a WiFi access point located inside a building even when the user is outdoors and walking, simply because of a static setting preferring WiFi over cellular networks. Even inside a building, it is common today for many devices to all use the 2.4 GHz WiFi network in a building, when in fact some of them using the cellular LTE network, the 5 GHz WiFi band, or even Bluetooth would provide better aggregate performance. When two devices want to exchange files with each other, for example a device backing up its data to another or streaming a movie to another, the choice between a direct WiFi link (using WiFi-Direct or ad hoc mode) and a path via an access point has to be made; by default, the latter option is usually made today, which may be inferior to the former.

In all these cases, what's best for an interactive videoconference may be different from what's best for a web app, and what's best for uploading photos to a server may be different from what's best for sharing a picture between two mobile devices—because these apps all have different performance objectives and the networks have different connectivity properties. The starting point for our solution is that the right answer to this question depends on what the app wants from the network; i.e., the *objective*. In addition, constraints on monetary costs and energy must be taken into account. Last but not least, what's best for an individual might be at odds with what's socially optimal for a group of concurrent users.

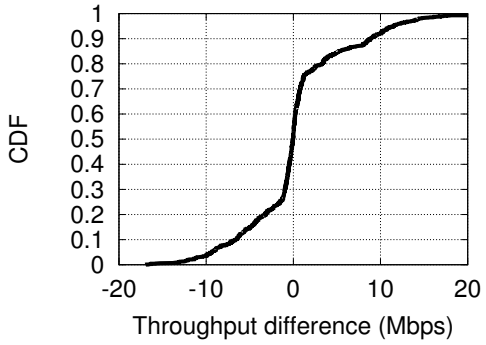
The problem of network selection is not a subnetwork or link-layer problem because it must operate across different subnetworks. Nor is it a network-layer problem—although traditionally the networking community has viewed it as such—because that layer lacks the ability to infer end-to-end properties of a path. Our position is that a good *transport architecture* for mobile systems must provide *network selection* as a core module, that it must consider objectives and costs, and that it must operate efficiently.

We propose a transport module that makes this choice for apps. This module, named *Delphi*, includes (1) components to predict the properties of each network by both learning from easy-to-measure observations and active probes; (2) a simple protocol to share properties learned between nearby nodes; and (3) a selector that makes the best network choice at each node taking both app objectives and social utility into consideration. We believe that Delphi is a significant step toward allowing a mobile device to fully harness the capabilities of all the networks at its disposal.

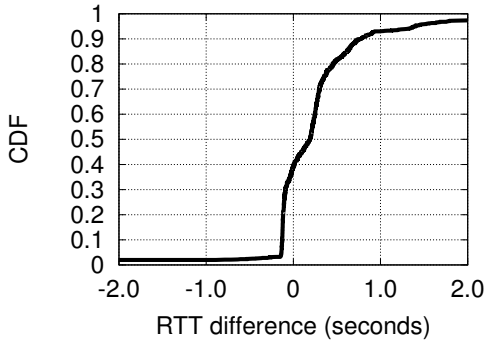
The question of choosing the best wireless network interface has received prior attention, as has the question of ensuring that TCP connections survive changes in this choice (see §4). To our knowledge, no prior work solves this problem when the candidate networks achieve throughput and delay values that are both variable and comparable in magnitude. In the past, it was almost always the case that WiFi performed better than 2G, EDGE, or 3G, allowing a static preference between these networks. Based on this assumption, previous work contributed to one of the following problems: 1) designing mechanisms to seamlessly switch between interfaces; 2) using multiple networks concurrently, striping packets between them; 3) specifying policies, e.g. delay background traffic, saving energy, and offloading data under the assumption that "WiFi is faster than cellular" or "WiFi is more energy-efficient than cellular".

However, increasing cellular speeds are fast invalidating these assumptions and make the problem of network selection more challenging. Today, WiFi and LTE throughputs are often comparable, but each varies significantly with time as channel conditions and offered loads change, so the choice isn't straightforward (see Figure 1). To better utilize the network capacity, an adaptive, online selection method is now required. Delphi provides a general-purpose wireless

ACM HotMobile'14, February 26–27, 2014, Santa Barbara, CA, USA.



(a) Throughput difference.



(b) RTT difference.

Figure 1: WiFi and LTE Throughput/RTT difference. $x < 0$ **means WiFi has lower throughput or RTT than LTE, and vice versa. This figure shows that neither WiFi nor LTE is categorically better.**

network selection scheme applying ideas from machine learning using features obtained using both passive observations and active probes. Delphi selects network intelligently when it is unclear which network performs best, and the right answer changes with time because the networks are variable in time or space, and are often of comparable quality.

Besides, prior research has not developed a systematic framework for network selection that addresses two conflicting concerns:

1. Optimizing individual objectives such as high throughput, low delay, or low page load time, while simultaneously

2. paying heed to a social objective such as good performance of the network as a whole.

Our work is a synthesis of new techniques for wireless network selection together with a plan for sharing information to achieve social objectives when network conditions are voltaile enough to preclude a static preference for one network over another.

2. SYSTEM OVERVIEW

Delphi's design is shown in Figure 2. Delphi runs on each device as a transport module to select the network for each data stream. The app specifies an objective to Delphi when it creates the stream.

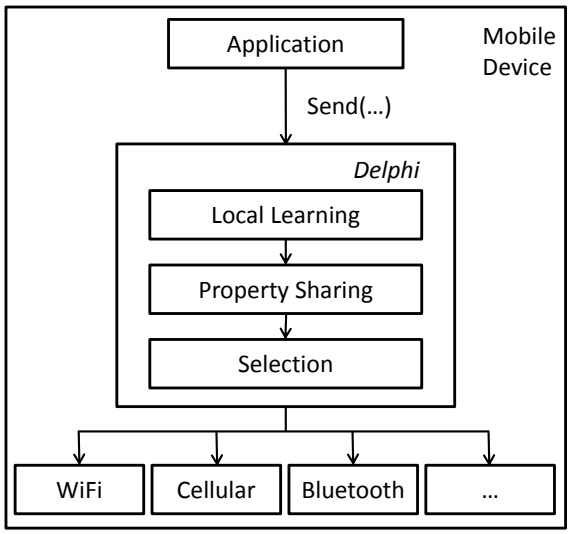


Figure 2: Delphi system overview.

For applications that use TCP, Delphi makes this choice once at the beginning, and does not change it for the stream because TCP does not handle roaming. For applications that embed a different transport library, e.g., one capable of roaming [17], or one where the session is made of request-response traffic with the possibility of changing IP addresses between requests, Delphi may change the network selected mid-stream.

Delphi strives to optimize the app-specified objective for the stream; Table 1 gives examples of objectives. Delphi also attempts to provide fairness across the different users sharing a wireless LAN.

Here are some examples of intended outcomes:

1. For apps that share the common objective of maximizing some property like throughput or throughput/delay, Delphi should select the network at each node that achieves an α-fair throughput allocation [15]. I.e., the allocation maximizes $\sum_i \frac{r_i^{1-\alpha}}{1-\alpha}$, and particularly the special case of $\alpha = 1$ (proportional fairness), which corresponds to maximizing $\sum_i \log r_i$, where r_i is the property of interest.

2. For an app wishing to minimize energy, Delphi should schedule the app's packets on a low-power interface such as EDGE.

3. For an app looking to minimize data usage, Delphi should delay transmissions until it has access to a free WiFi network.

In general, to realize these objectives systematically, Delphi's design consists of three components:

1. *Local learning* collects data to estimate network conditions:

   (a) Channel information, e.g., WiFi RSSI, WiFi link rate, and cellular signal strength.

   (b) Transport-layer performance statistics, such as average throughput, per-packet delay, and ping RTT.

   (c) Location and user activity [1].

2. *Property sharing* distributes the results of the local learning stage to all nearby nodes.

3. *Selection* is responsible for objective maximization, using the information from the previous stages.

Objective	Target apps
Maximize $\frac{throughput}{delay}$	Video streaming apps (YouTube, Netflix, Hulu)
Maximize throughput	Backup apps, system updates
Minimum delay	VOIP apps such as Viber and Skype
Minimize energy	Apps that periodically synchronize with a server like email
Minimize data usage	Apps that aggressively prefetch such as web search interfaces
Maximize interactivity	Apps that have both foreground and background modes. In this case, messages belonging to foreground apps are scheduled preferentially.

Table 1: Objectives and apps that use them

3. DESIGN

There are several ways in which the three components—local learning, property sharing, and selection—may be implemented. This section describes some alternatives in detail.

3.1 Local Learning

For each cellular and local-area network, Delphi must estimate properties of interest to apps, such as network throughput, delay, time to deliver a file of a certain size (which would additionally depend on the packet loss rate), the tail of the packet delay distribution, etc. Some of these parameters are obtainable by inference from passive measurements, while for others, some active measurements are required.

Property estimation for cellular networks is a tricky problem because channel conditions vary with time, especially for moving users. One might consider active probing techniques [18], but these incur significant overhead on cellular networks. Passive approaches to estimate throughput [7] are of greater practical interest, but our goal is not just to infer the properties of the cellular network, but to make decisions relative to other choices such as WiFi.

Local learning includes both passive and active measurements of the available networks, gathering the following statistics: signal strength of each available wireless link, cellular network type (GPRS/3G/LTE), ping RTT, DNS lookup time, number of observed WiFi APs, WiFi link rates on different WiFi options (frequency band, WiFi-Direct, infrastructure-mode, etc.), packet loss rates, offered load observed on the network from passive measurement (for WiFi, the fraction of time a node had data to send over the network) and, in some cases, measured throughput from active data probes.

During an active probe, which is useful to calibrate how well a throughput estimated from passive or lightly active observations (such as RTT) reflects reality, a node simply initiates a stream (ideally from an existing app workload, rather than with new packets). It tracks the observed throughput and logs the start and end times of the experiment. Later, when nodes share their locally-learned information, they can make the appropriate selection decisions using this information.

At the end of the local learning stage, a node has enough information to make its own local decision about which network to use. For instance, it can infer whether to use WiFi or cellular, assuming (for the moment) that it does not care about other users. We approach this problem by developing a simple machine learning classifier over the observed local features to determine whether one network is better than the other for any given objective, with a confidence metric reflecting the accuracy of the prediction.

We use a random forest classifier in our prototype experiments. We collected TCP throughput data from twenty different locations (each location involving multiple experiments lasting several minutes) over both WiFi and multiple cellular networks. We trained the classifier on 10% of the data and tested on the remaining 90%. Figure 3(a) shows the result for maximizing throughput in a dataset consisting of WiFi and Verizon LTE samples (the approach works for other cellular networks as well). The learning classifier improves median throughput by 29% over Cell-Only, and by 11% over WiFi-Only across different transfer sizes. We also used a similar learning process to pick a network to minimize packet RTT. Figure 3(b) shows that the classifier reduces delay by 18 ms (32%) over Cell-Only (Figure 3). These results indicate that it may be possible to use machine learning to quickly predict which network choice optimizes a property such as throughput, delay, as well as objectives of the form in Table 1. We are refining this approach, making sure not to overfit to the trained data.

Instead of each node making a greedy decision, in Delphi, nodes share information between each other (within a single local-area network) so they can make balanced decisions. We assume cooperating nodes; handling non-cooperating nodes is an interesting area of future work, but notice that the damage done by a non-cooperating node is mitigated by the sharing mechanisms at the MAC and network layers in WiFi and cellular networks.

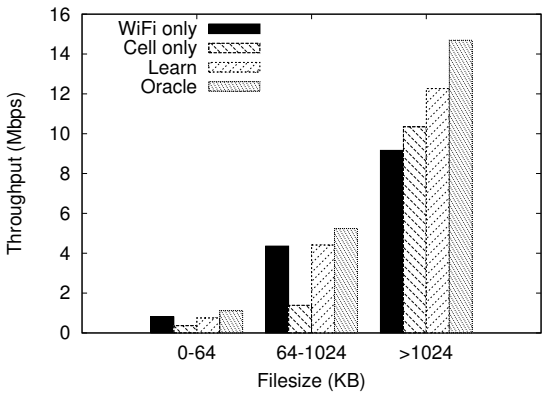
We also note that in some cases a purely local decision might be sufficient or even required. For instance, if an app has data usage or energy constraints, switching to a lower-power or cheaper network is possible even in the absence of global knowledge. Furthermore, the energy and cost constraints might even be at odds with the social optimum. As an example, the social optimum might dictate that the node transmit on a higher-power or more expensive interface for the greater good. In such cases, Delphi respects app constraints on energy and cost, even if it results in a socially inefficient solution.

3.2 Property Sharing

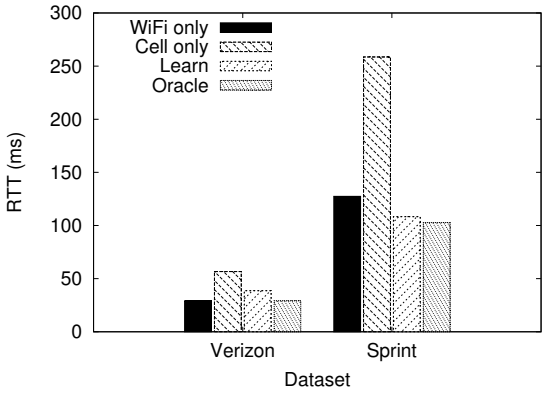
Property sharing is a simple protocol that uses link-layer WiFi broadcast to share information between nodes in the same WiFi broadcast domain about the locally-learned parameters. Every several seconds, each node broadcasts the locally-learned information about its property estimates, as well as estimates of offered load and the time-throughput observations of any active experiments. This protocol allows any pair of nodes in the same broadcast domain (e.g. connected to the same access point) to not only know its own properties, but also those of its neighbors. This ability is critical because it allows the nodes to cooperate to improve network selection. At the end of the property sharing stage, each node has its own property estimates, as well as those of its neighbors.

3.3 Selection

The goal of the selection stage is for each node to determine its choice of network; our key insight is that each node runs an algorithm over both its data *and* those of its neighbors, so the expectation is that each node can compute both its choice *and* those of its neighbors. This idea is an example of Hofstadter's superrationality [10]: when multiple agents face a choice and are interested in a cooperative outcome, each node knowing that the other will also cooperate and



(a) Throughput



(b) Delay

Figure 3: Median throughput and delay using random forest learning ("learn") compared to using WiFi/cellular and an "oracle" with future knowledge.

will run the same algorithm (for the social good) can lead to a better outcome than in the classic game-theoretic setting where no such explicit cooperation is assumed. Applying this notion to our context requires each node to run the same algorithm over a common set of inputs, available everywhere.

To illustrate this concept with a simple example, consider two nodes, *A* and *B*, with one app running on each node, interested solely in high throughput. Suppose *A* and *B* both get 10 Mbits/s when each is alone on WiFi, but on LTE they get 6 Mbits/s and 7 Mbits/s, respectively. Further, suppose that if both *A* and *B* used WiFi concurrently, they get only 5 Mbits/s each (Table 2). If each node chose greedily, solely based on local WiFi link rates and LTE throughputs, they would both pick WiFi giving each 5 Mbits/s. But each knowing about the other user and sharing properties, *A* can select WiFi and *B* LTE, which maximizes the total throughput across all users[1]. This outcome is easy to compute if each node had access to this table (the intended outcome of the previous two stages).

We are investigating two approaches to the selection problems. The first involves active probe data and is less optimal, but we have implemented and tested it. The second uses entirely passive

[1]This corresponds to α-fairness with $\alpha = 0$. Other values of α can be used as well

A/B	A:WiFi	A:LTE
B:WiFi	(5, 5)	(6, 10)
B:LTE	(10, 7)	(6, 7)

Table 2: Example: Throughput for A, B on WiFi, LTE.

observations and we believe can provider better results, but involves more computation at each node. We describe these below.

FCFS approach. In the first-come, first-served (FCFS) approach, a new sender *S* that wishes to transmit first broadcasts an *intent to transmit*. This broadcast tells all senders in the vicinity that *S* is about to initiate a transmission. On receipt of this broadcast, all nodes including *S* send out active probes to estimate their throughput on all possible networks. We call this throughput the "shared throughput" for each sender on a particular network, because it is the throughput any of the senders would see if all of them utilized the network at the same time. Once a node estimates these shared throughput values, it broadcasts them to everyone else, and each node simply picks the network that maximizes the aggregate shared throughput.

As implemented currently, each node works to maximize total shared throughput, but it can (should) be extended to achieve different global objectives, such as *proportional fairness*, which maximizes the sum of the logarithms of throughput (or other app objectives), or the more general α-fairness metric.

All-permutations approach. The first step is to determine what happens if any subset of nodes were transmitting on the same shared wireless LAN (e.g., WiFi). This computation sounds daunting, but it is in fact tractable for modest numbers of heavily active (say 10-20) nodes on the shared network. For each subset of nodes, because we know their offered load and link rates, we can compute the throughput each would get if the given subset were active. This computation can be carried out offline using a WiFi link-layer simulator, and stored in a lookup table that can be looked up at run time. Then, by comparing with the cellular property estimates, each node can decide which network it should use so as to maximize a global utility function. We have not tested this method experimentally yet, though we have reason to believe that it is feasible: the time to look up the results for a given subset is minuscule, and even with 20 active nodes, there are only about a million combinations to consider!

To evaluate the FCFS approach, we implemented a time-slot-based simulator that randomly generates the amount of data to be transmitted between pairs of phones and the time at which each pair starts transmitting. For calibration, we measured the actual throughput in a real wireless network, with two nodes *A* and *B*, under the following three scenarios:

1. Only node *A* is transmitting.
2. Only node *B* is transmitting.
3. Both nodes *A* and *B* are transmitting.

While running the simulation, we plug in the right value of the achieved throughput (one of item 1, 2, or 3 above) based on the set of concurrent transmitters in the simulated network in every time-slot.

We compare this method with other schemes, including WiFi 2.4 GHz only, WiFi 5 GHz only, and a greedy algorithm that the senders choose the best network by probing once (shown as Greedy in Figure 4), without notifying other phones sharing the same network. Figure 4 shows the average aggregate throughput for three pairs of

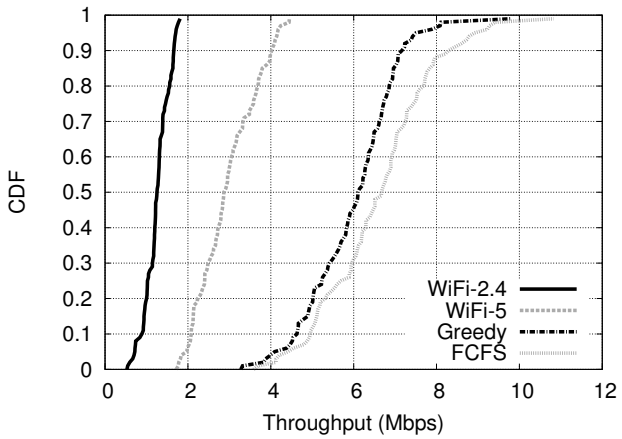


Figure 4: CDF of average throughput across 3 pairs of phones doing P2P transmission choosing among 3 networks (WiFi 2.4 GHz, WiFi 5 GHz and Bluetooth). The median value for each CDF is shown in the legend (in Mbps).

phones. FCFS achieves 5.1× gain over using WiFi 2.4 GHz only, and 2.3× gain over using WiFi 5 GHz only, and outperforms the greedy algorithm.

4. Related Work

The idea of using multiple networks has attracted significant attention from researchers over the past decade. Early work showed the benefits of switching between or combining multiple networks in different scenarios. Bahl et al. [5] present cases where selecting the proper network can reduce energy consumption, enhance network capacity, and improve mobility.

Several papers aim to achieve specific goals by combining multiple wireless networks. CoolSpots [13] and SwitchR [2] reduce mobile device power consumption. FatVAP [11] and MultiNet [8] improve throughput by allowing a single WiFi card to connect to multiple WiFi APs. COMBINE [4] improves individual device throughput by pooling the network capacity of neighboring devices. Multipath TCP [19] allows one TCP connection to be split across multiple paths. Our problem is different: Delphi does not stripe packets from a single connection across multiple networks. Instead, it picks one network interface for an app and sends all packets over that interface, switching to a new interface if conditions change dramatically and if permitted by the app.

Contact Networking [6] provides localized network communication between devices with multiple networks, focusing on mechanisms for neighbor discovery, name resolution, and routing. Airdrop [3], a feature of Apple OS X, allows users to share files over both WiFi and Bluetooth, but it is designed explicitly for the purpose of file sharing and does not extend to other applications (it is also not clear that it adaptively selects the network).

A substantial amount of prior work focuses on switching between different networks. Zhao et al. [20] designed network layer mechanisms for interface switching using Mobile IP [14]. Their mechanisms adapt to network topology changes, striving to minimize routing hop count. Stemm et al. [16] (and subsequent papers by others) developed vertical handoff techniques for heterogeneous networks. These works provide mechanisms to gracefully switch between different networks, whereas Delphi addresses the policy problem of "which network to pick".

MultiNets [12] proposes a mechanism to allow smartphones to use multiple networks based on certain policies (such as energy saving, data offloading, and performance). However, users need to manually configure the policy, which in turn affects all running apps. In contrast, our design requires no user interaction and is able to distinguish between different apps automatically based on app objectives. Intentional Networking [9] provides APIs that allow apps to label their network flows. The labels specify *background* or *foreground* to indicate if the flow is delay-tolerant, and *large* or *small* to specify the amount of data to be transmitted. This design is beneficial when there is a concurrent mix of foreground and background traffic. In contrast, our design provides apps with the capability to specify a quantitative metric to optimize directly. Both Multinets and Intentional Networking assume that WiFi has better performance than cellular wireless networks, which we have shown earlier to no longer be true.

5. Conclusion and Future Work

This paper introduced Delphi, a transport-layer module for mobile network selection. Applications on a mobile device express their desired objectives that Delphi optimizes by selecting the best network, balancing individual optimality with social utility. Delphi has three key components: "local learning", in which a node estimates or infers useful properties of different networks efficiently, "property sharing", in which nodes share what they learn with other nearby nodes, and "selection", in which each node selects a network using what it has observed locally and from its neighbors. An implementation and experimental evaluation of Delphi is in progress; some pieces have been built and experimented with as reported in this paper.

6. Acknowledgements

We are grateful to the HotMobile reviewers, and Mary Baker (the shepherd) in particular, for many helpful comments. We thank Amy Ousterhout, Lenin Ravindranath and Somak Das for their thoughtful suggestions. We thank the members of the MIT Center for Wireless Networks and Mobile Computing (Wireless@MIT), including Amazon.com, Cisco, Google, Intel, Mediatek, Microsoft, ST Microelectronics, and Telefonica, for their support. This work was also supported in part by NSF grant 0931550.

References

[1] Recognizing the user's current activity. `http://developer.android.com/training/location/activity-recognition.html`.

[2] Y. Agarwal, T. Pering, R. Want, and R. Gupta. SwitchR: Reducing System Power Consumption in a Multi-Client, Multi-Radio Environment. In *Wearable Computers*, 2008.

[3] iOS: Using AirDrop. `http://support.apple.com/kb/HT5887`.

[4] G. Ananthanarayanan, V. N. Padmanabhan, L. Ravindranath, and C. A. Thekkath. Combine: Leveraging the Power of Wireless Peers through Collaborative Downloading. In *MobiSys*, 2007.

[5] P. Bahl, A. Adya, J. Padhye, and A. Walman. Reconsidering Wireless Systems with Multiple Radios. *SIGCOMM CCR*, 2004.

[6] C. Carter, R. Kravets, and J. Tourrilhes. Contact Networking: a Localized Mobility System. In *MobiSys*, 2003.

[7] A. Chakraborty, V. Navda, V. N. Padmanabhan, and R. Ramjee. Coordinating cellular background transfers using loadsense. In *MobiCom*, 2013.

[8] R. Chandra and P. Bahl. MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card. In *INFOCOM*, 2004.

[9] B. D. Higgins, A. Reda, T. Alperovich, J. Flinn, T. J. Giuli, B. Noble, and D. Watson. Intentional Networking: Opportunistic Exploitation of Mobile Network Diversity. In *MobiCom*, 2010.

[10] D. Hofstadter. *Metamagical Themas: Questing for the Essence of Mind and Pattern*. Basic books, 1985.

[11] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi. Fatvap: Aggregating ap backhaul capacity to maximize throughput. In *NSDI*, 2008.

[12] S. Nirjon, A. Nicoara, C.-H. Hsu, J. Singh, and J. Stankovic. Multinets: Policy oriented real-time switching of wireless interfaces on mobile devices. In *RTAS*, 2012.

[13] T. Pering, Y. Agarwal, R. Gupta, and R. Want. Coolspots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces. In *MobiSys*, 2006.

[14] C. E. Perkins. Mobile IP. *Communications Magazine, IEEE*, 1997.

[15] R. Srikant. *The Mathematics of Internet Congestion Control*. Birkhauser, 2004.

[16] M. Stemm. Vertical Handoffs in Wireless Overlay Networks. Technical Report csd-96-903, University of California at Berkeley, May 1996. Master's Thesis.

[17] K. Winstein and H. Balakrishnan. Mosh: an interactive remote shell for mobile clients. In *USENIX ATC*, 2012.

[18] K. Winstein, A. Sivaraman, and H. Balakrishnan. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. In *NSDI*, 2013.

[19] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *NSDI*, 2011.

[20] X. Zhao, C. Castelluccia, and M. Baker. Flexible network support for mobility. In *MobiCom*, 1998.