Link-alike: Using Wireless to Share Network Resources in a Neighborhood

Szymon Jakubczak MIT
David G. Andersen CMU
Michael Kaminsky Intel Research
Konstantina Papagiannaki Intel Research
Srinivasan Seshan CMU

Asymmetric broadband connections in the home provide a limited upstream pipe to the Internet. This limitation makes various applications, such as remote backup and sharing high definition video, impractical. However, homes in a neighborhood often have high bandwidth wireless networks, whose bandwidth exceeds that of a single wired uplink. Moreover, most (wired and wireless) connections are idle most of the time.

In this paper, we examine the fundamental requirements of a system that aggregates upstream broadband connections in a neighborhood using wireless communication between homes. A scheme addressing this problem must operate efficiently in an environment that is: i) highly lossy; ii) broadcast in nature; and iii) half-duplex. We propose a novel scheme, Link-alike, that addresses those three challenges using opportunistic wireless reception, a novel wireless broadcast rate control scheme, and preferential use of the wired downlink. Through analytical and experimental evaluation, we demonstrate that our approach provides significantly better throughput than previous solutions based on TCP or UDP unicast.

I. Introduction

People are creating and sharing data in unprecedented quantities. Websites such as Flickr and YouTube provide easy portals through which users can share their photos and movies; peer-to-peer systems such as BitTorrent [14] offer a simple interface to share files; and several online storage companies offer users an easy way to backup their data remotely. These new modalities of information exchange, however, are voracious consumers of bandwidth. Most of this content is multimedia, which home users create megabytes and gigabytes at a time. The volume of data they create and share is growing at a staggering rate each year.

Increasingly, the main problem that users face in trying to share their data is the limited bandwidth they have at home. Typical broadband connections, such as cable modems or DSL, are asymmetric: ISPs provision their links such that users receive more downstream bandwidth (into the home) than upstream (out of the home). Thus, downloading rich content is feasible, but publishing such content is challenging.

In this paper, we present a solution to this problem that exploits wireless connectivity to aggregate the bandwidth of multiple broadband uplinks; our system then makes this aggregate bandwidth available to users for their uploads. This approach is motivated by two observations. First, most broadband connections are idle most of the time [18]. Second, wireless 802.11 connectivity is becoming commonplace in the home, and these wireless networks have much higher capacity than residential broadband connections. Current 802.11a/g links offer 54 Mbps and the upcoming 802.11n standard offers over 200 Mbps while cable modem or DSL links typically provide only 384–768 Kbps upstream. This significant difference in bandwidths provides a unique opportunity for users to harvest the idle upstream capacity of cooperating neighbors.

Three trends make this application scenario favorable in neighborhoods. First, community efforts like FON [3] and Meraki [22] clearly indicate the desire for users to share their wireless and wired connectivity in the presence of appropriate incentives (e.g., for a monetary reward or free access to other parts of the network). Second, Internet Service Providers like British Telecom (BT) seem to be leaning in this direction with products such as the BT Home Hub [5]. The BT Home Hub promotes wireless community networking and sharing of wired connectivity by supporting two simultaneous SSIDs, one of which forms part of the FON network. Finally, a number of manufacturers now offer affordable, open-firmware wireless routers (e.g., FON, OpenMesh Linksys, and Netgear [7, 4, 2]), which enables easy deployment of new functionality. Similarly, manufacturers like Broadband Bonding [1] and Wi-Boost [6] build custom hardware for the bonding of wired links through wireless connections.

A common approach to wired link aggregation through wireless is to stripe traffic over multiple access points to increase throughput [13, 19]. That work, however, mostly focuses on the mechanisms needed to perform the striping transparently. Such mechanisms

include deciding how long to stay on each wireless channel, minimizing the time needed to switch channels, handling the queues of multiple TCP connections, and interfacing with the kernel. A common assumption in these approaches is that the wireless links can deliver packets quickly and reliably to the access point. In contrast, our focus is on devising a solution that works well within the constraints of neighborhood wireless deployments: they are unplanned, they have a wide range of link quality, and obstructions and interference are common.

In this paper, we re-examine the problem of aggregating broadband connections, with a specific focus on the requirements imposed by the neighborhood environment. The three central challenges are:

1. **Work efficiently in a high-loss environment.** Wireless links between neighbors are unplanned and can exhibit a wide range of performance [18]. Transport protocols such as TCP may perform quite poorly in this environment, particularly when communicating with distant or occluded neighbors. The system should maintain high throughput despite this potentially high loss rate.

2. **Work efficiently in a broadcast environment.** Wireless is fundamentally a broadcast medium, in which all nodes have a chance to receive a particular transmission. If *any* neighbor with spare uplink capacity receives a transmitted packet, the source should not waste wireless capacity retransmitting the packet to a different neighbor.

3. **Make efficient use of the half-duplex, shared medium.** A node cannot send and receive simultaneously on a single channel. Given that downlink wired capacity is often abundant in the neighborhood environment, the system should favor using the wired downlinks instead of the wireless whenever possible. (For example, as we discuss later, our system uses the downlinks to schedule nodes and acknowledge received data.)

We present *Link-alike*, a system that uses *opportunistic wireless broadcast* to aggregate neighboring upstream broadband bandwidth. The source broadcasts data to everyone that can hear, and the participants in the system decide (using one of several techniques described later) how to most efficiently get that data to the intended recipient. Although the idea of wireless opportunism has been applied before to routing in mesh networks [11, 21, 12], Link-alike is the first system to use this technique to aggregate the uplink bandwidth of multiple access points.

The neighborhood scenario we consider is inherently *cooperative*: users share resources so that they individually receive improved throughput when they need it and idle bandwidth is available. Thus, in addition to maximizing throughput, Link-alike tries to minimize wasted transmissions on both the wired and wireless networks since other users may be using these resources simultaneously.

After reviewing background information and related work in Section II, we describe in Sections III and IV the design and implementation of Link-alike and its strategies for sending data through neighbors and for making efficient use of the wireless medium. We then provide a detailed theoretical comparison of the opportunistic approach versus unicast techniques (Section V), including the maximum achievable throughput given a set of system parameters. Section VI shows through analysis and experimentation that Link-alike achieves nearly optimal efficiency. We conclude by discussing practical deployment issues and future extensions to our work.

II. Background and Related Work

Our work builds upon related work in three areas: network connection sharing, opportunistic wireless delivery, and techniques for improving transport efficiency over lossy networks.

II.A. Network Connection Sharing

These schemes share Link-alike's goal of aggregating multiple broadband links to increase upload or download speeds. Particular proposals, such as [27, 17], advocate doing so using wireless, and define appropriate flow scheduling approaches. The recently proposed FatVAP [19] and at least two commercial products also offer network connection sharing: *Broadband Bonding* [1] and *WiBoost* [6]. These products appear to target a similar application scenario to that addressed by Link-alike. Based on the limited literature on their products, it appears that they also focus on flow load balancing and unicast distribution. As we show in Section VI, Link-alike can substantially out-perform these unicast schemes.

II.B. Opportunistic Wireless Delivery

Traditional network and application-layer approaches to wireless connectivity attempt to force it into the abstraction of a relatively reliable point-to-point link. Packets are transmitted via unicast to receivers, and 802.11 NICs mask losses by retransmitting the packet,

potentially using different modulation schemes, until a maximum number of retries or a successful reception. This approach provides system designers with an easy and familiar interface, but hiding the complexity of the wireless medium also hides some of its power.

The wireless medium is intrinsically broadcast. Because receivers are separated spatially, each receiver typically sees an uncorrelated set of bit-errors. Hence, even if one receiver is unable to decode a wireless frame, others may be able to do so. In recent years, this observation has been exploited by several opportunistic transmission schemes to substantially increase throughput in multi-hop wireless mesh networks.

MRD (Multi-Radio Diversity) [23] and SOFT [29] use reception variability to hide losses in transmissions from clients to an access point. ExOR [11] and others [21] use reception variability to opportunistically forward data as far as possible in a mesh network. MORE [12] combines the ExOR design with network coding to greatly simplify the coordination task. There have also been efforts [24, 15] that apply similar techniques to forwarding in sensor networks. Although these previous systems use data received at different nodes opportunistically, none of these have considered the particular problem domain that Link-alike addresses. The wide range of previous efforts demonstrates that the effective use of opportunistic forwarding in a particular context mandates solutions to a number of domain-specific issues.

In this work, we use the wireless medium to naturally provide an *anycast* communication model, where it is sufficient for any of the potential receivers to successfully receive a transmitted packet. In the scenario we examine, the opportunistic approach is key to allowing Link-alike to take advantage of neighbors with poor reception, where making explicit (and often unsuccessful) attempts to reach only that neighbor would simply waste network capacity.

II.C. Transport over lossy networks

TCP treats losses as an indication of congestion and reduces its sending rate in their presence. It has long been observed that as a result, TCP performs poorly when it experiences significant numbers of non-congestive losses—as is common on wireless links. Link-alike's approach is to use a custom transport protocol over the wireless link, while using TCP over the wired uplink from the neighbors to the destination. This approach is inspired by many proxy-based solutions such as Split-TCP and Snoop [8], among others. Because Link-alike assumes a modified sender as well as proxies on the neighboring APs, it does not need to force its changes into TCP as much prior work does. Link-alike still uses end-to-end acknowledgments for reliability, but it implements its own wireless broadcast rate-control scheme, described in Section III. We emphasize that Link-alike does require changes on the APs themselves; however, we feel that such a requirement is well justified in the studied environment. BT, for example, already controls what is running on their Home Hub, and re-configurable APs are shipped by several manufacturers (e.g., OpenMesh, Linksys, Netgear, Meraki, FON [7]).

III. Design

Link-alike's primary goal is to minimize the total transfer time by using neighbors in the most effective way possible. The system's secondary goal is to minimize wireless cost, i.e. the amount of airtime occupied by the transmission of each unique piece of information—a metric that will reach high values in the presence of unnecessary retransmissions.

This section presents the design of Link-alike, which achieves these two goals by meeting the three requirements set forth in Section I: it works efficiently in a high-loss environment, it works efficiently in a broadcast environment, and it makes efficient use of the half-duplex, shared medium.

Throughout our work, we focus on a single wireless transmitter in the network (treating other transmitters merely as interference and the causes of packet loss), and we target our solution to using neighbors who are one wireless hop away from the source. We assume that the transmitters use a single shared channel. We believe this focus is reasonable since most home networks are typically idle and the current ratio of wireless to wired bandwidth does not require a large number of neighbors to maximize performance. In Section VII, we discuss possibilities for extending our system to relax these constraints.

To illustrate how Like-alike's opportunistic broadcast strategy provides the best solution for the neighborhood environment, we compare it to two alternate strategies: TCP striping and UDP unicast.

III.A. Alternate 1: Unicast TCP striping

Many existing broadband aggregation techniques direct different TCP connections through different access points or neighbors. Although these schemes allow faster aggregate downloads (e.g., if a Web browser opens multiple TCP connections), they typically do not directly speed up the transfer of a single file. To

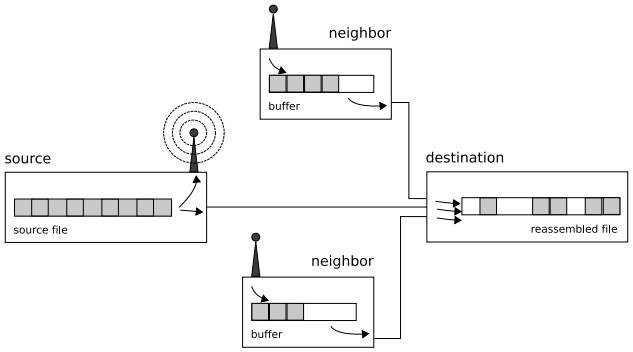

Figure 1: The basic data flow in the system. The source can transmit chunks of the file either on the wireless interface or the wired uplink. Each neighbor buffers received chunks before sending them upstream. The destination reassembles the chunks it receives into the output file.

compare with these schemes, we implement an aggregator where the sender opens a TCP connection to the destination through each neighbor. The source sends chunks of the file to the open sockets on a greedy first-come-first-served basis. No special queuing discipline is applied at the wireless interface.

This approach suffers from two weaknesses. First, when a packet received by the intended neighbor is corrupted, the link layer must retransmit the packet even if it was received correctly by other nodes. Second, if losses are severe enough that the link is unable to retransmit a packet successfully, TCP will be exposed to the loss. As many past studies have shown, TCP performs poorly when operating over such lossy links.

III.B. Alternate 2: UDP Unicast

Figure 1 shows the basic data flow for our UDP unicast setup (and for Link-alike, described below). The source divides the file into packet-sized chunks, which it sends either over its wired uplink or broadcasts on the wireless interface. Any neighbor that successfully receives a chunk can buffer it, ship it to the destination over its wired uplink, or drop it.

In UDP unicast, the sender addresses each wireless transmission to a specific neighbor. This protocol implements the optimal unicast strategy developed in Section V: it ships chunks through the neighbor with the best packet reception probability who has available bandwidth. Since 802.11-compliant interfaces support multiple modes (bit-rates) which trade error-protection for transmission time, the source picks the *most efficient* neighbor, i.e., that which offers the lowest expected transmission time (ETT) per packet [10]. Observe that this strategy matches the approach taken in FatVAP [19].

The unicast strategy requires estimates of reception probabilities. To obtain these estimates, all neighbors passively sniff the wireless interface and count the number of unique packets they observed from the source (uniqueness is determined by the packet sequence number). The neighbors report to the destination the number of packets they observed during the last measurement period. (A measurement period is defined as a sequence number range so that all neighbors provide a consistent count.) The destination uses this information to rank the neighbors in ascending ETT order. The neighbors also send their queue size and estimated uplink capacity with each upstream packet. Using this information, the destination computes the optimal transmission schedule and provides it to the source.

Finally, the destination sends ACKs to the source over the downstream wired connection. The neighbors do not need to query the destination to decide which chunks to send upstream: they simply upload each chunk that the source successfully transmits to them.

Like the TCP alternative, this approach must also retransmit packets unnecessarily when the intended recipient does not receive the packet. However, it does not suffer from the problems related to TCP performance over lossy links.

III.C. Link-alike: UDP broadcast

Link-alike is a *two-tier* file transfer protocol that is similar to the UDP unicast approach. On the first hop, a Link-alike source broadcasts chunks to all neighbors (as opposed to the UDP unicast scheme above which assigned chunks to specific neighbors and retransmited them until they arrived at that neighbor). Simultaneously, the source greedily transmits chunks on its own uplink and does not transmit those chunks over the wireless.

For the second hop, neighbors use TCP to ship their chunks over the wired uplink to the destination. Because Link-alike is a broadcast scheme, however, more than one neighbor might receive a given chunk. Thus, the core of Link-alike is a scheduling algorithm that determines which neighbors transmit which chunks over the wired uplink.

III.C.1. Scheduling

Our design for Link-alike results from the following design decisions:

Centrally scheduled. The most fundamental decision that the source and neighbors must make is which chunks to send to the destination and how. To use both the wired and wireless medium efficiently and

to minimize transfer time, these nodes must use an effective transfer strategy.

In designing Link-alike, we explored three such strategies: scheduled, random, and network coding. In the scheduled scheme, a single node decides which nodes send which chunks. Because this one node assigns chunks to neighbors, it can avoid redundancy and reduce transfer time.

The primary advantage of the random and coding schemes is that they do not require per-chunk coordination among the neighbors. In the random scheme, individual neighbors decide independently which chunks to send to the destination by randomly picking from among the chunks they have received. Our initial experiments showed, however, that the random scheme runs into the coupon-collectors problem (requiring $n \log n$ uploads) unless the chunks are acknowledged to every neighbor, thus requiring coordination. Additionally, the probability of two neighbors uploading the same chunk is inversely proportional to the number of chunks buffered at the neighbors; as a result, random scheduling is only efficient when the system is highly wired-bottlenecked (Section V). When wired-bottlenecked, the source can fill the neighbors' buffers quickly enough to avoid leaving their uplink capacity idle.

The network coding approach avoids these problems by generating random linear combinations of the received chunks. Such combinations are independent with very high probability. This scheme, however, has several downsides: the number of chunks combined together must be limited which requires batching; both the neighbors and the destination have to perform finite field algebra; finally, the neighbors must know how many coded chunks to generate [12], which requires solving the capacity problem defined in Section V. Based upon these limitations, we chose the centrally scheduled design for Link-alike.

The destination node is the scheduler. Because the neighbors are already communicating with both the source and the destination, these two nodes are natural choices to act as the scheduler.

The choice of the destination node stems from the fact that the scheduling node must know which chunks each neighbor has in order to produce an effective schedule. The neighbors send reception reports to the scheduler. Thus, if the source is the scheduler, the neighbors must transmit their reports over the wireless. These transmissions can cause interference, and small packet transmissions make less efficient use of the wireless capacity. By choosing the destination as the scheduler instead, the neighbors can send reception reports over their substantially more reliable upstream wired connection.

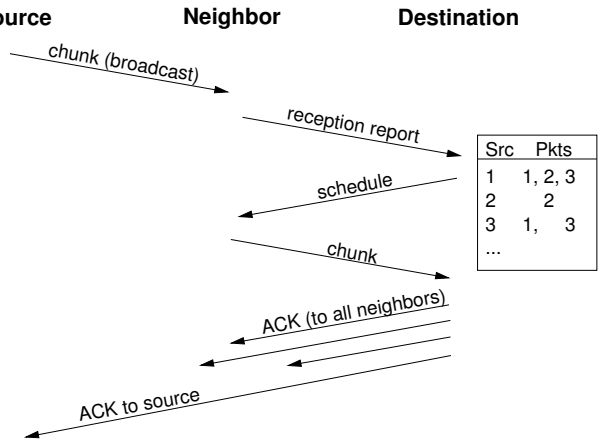

Figure 2: The basic communication via neighbors in destination-coordinated FCFS. For clarity, the diagram shows only a single chunk exchange; in practice, the neighbor would be uploading previously scheduled data while awaiting a new schedule.

Greedy, first-come, first-served scheduling. Given a set of neighbors and the list of chunks they have received from the source via broadcast wireless, the destination must schedule chunks to neighbors. Link-alike uses a greedy, first-come-first-serve (FCFS) strategy that schedules chunks to the first neighbor with capacity that has that chunk. (Neighbors piggyback their available capacity on the shipped chunks.) By scheduling chunks onto neighbors as soon as they have spare capacity, it ensures that at least *that* neighbor will efficiently use its bandwidth. Although we have not (yet) proved whether our greedy strategy is optimal, in practice it works well, as we show in Section VI.

III.C.2. The greedy scheduler algorithm

The details of the greedy scheduler are as follows (see Figure 2):

1. The source broadcasts the chunks of the file in order. When it completes, it cycles back and rebroadcasts chunks that have not yet been acknowledged (Step 6) by the destination. The source transmits chunks at a rate it estimates will keep its neighbors uplinks full without wasting wireless bandwidth (Section III.C.3).
2. Neighbors receive a wireless broadcast and send a *reception report* to the destination via their wired uplinks. They insert the received chunk into their buffer.
3. The destination maintains a FIFO list of the chunks reported by each neighbor. It declares a neighbor *schedulable* when that neighbor has

chunks to send that have not already been scheduled to another neighbor and when it believes that neighbor to have spare uplink capacity (Section III.C.3).
4. The destination assigns chunks to schedulable neighbors in FCFS order.
5. The destination notifies the schedulable neighbors which chunks they should send.
6. When the destination actually receives a chunk, it reliably sends an acknowledgment to the source and to every neighbor. These ACKs permit neighbors to flush useless chunks from their own buffers (Section III.C.4).

Note that for clarity, the figure shows the communication as if it were synchronous; in practice, both neighbors and the destination operate asynchronously and batch some of their communication for efficiency, as we describe later.

III.C.3. Capacity Estimation

Capacity estimation is required for two aspects of chunk scheduling in Link-alike. First, the neighbors must estimate their available uplink capacity. Second, in order to avoid wasting wireless transmissions and needlessly overflowing the neighbors' buffer space, the source should determine the slowest rate at which it can transmit on the wireless medium while ensuring that neighbors' uplinks are full.

Link-alike deals with capacities as bandwidth-delay products (BDP) instead of directly estimating rates.

Uplink capacity estimation using BDP. Link-alike neighbors send chunks to the destination using TCP. The FCFS strategy schedules new chunks to a particular neighbor only if it has spare capacity. To avoid an imbalance of outstanding requests to neighbors, the destination issues upload requests to each neighbor at a rate no greater than its upload capacity. Rather than measuring and enforcing a *rate*, Link-alike uses a simple per-neighbor sliding window-based approach: the number of *outstanding* uploads (assigned to the neighbor but not yet executed) is bounded by the window size. Clearly, to achieve full link capacity, the window size must exceed the bandwidth-delay product (BDP) of the wired link. An excessive window, however, could lead to imbalance as we discuss in the next subsection.

To estimate the BDP, the neighbors use TCP buffer auto-tuning and query the kernel for its estimate of the send socket buffer size each time they pass a chunk to the socket. The resulting value is a safe upper bound on the amount of data that the neighbor could upload to the destination (even if it is too low, the resulting low socket buffer size would itself limit the neighbor's capacity).[1] This value is sent to the destination along with the chunks, so the destination can adjust the window size before it issues the next schedule to fill the window.

Source broadcast rate. Unlike neighbor capacity estimation, there is no well-defined optimal source broadcast rate; in many scenarios, larger values continue to marginally increase the end-to-end transfer rate while considerably increasing the load on the wireless network.

To understand this trade-off, consider a source with neighbors $n_1, n_2, n_3, \cdots, n_k$ where the probability that these neighbors hear transmissions from the source decreases exponentially with increasing subscript. At a low transmission rate, the source can saturate n_1's uplink. At double that base rate, n_1 and n_2 both receive enough chunks to saturate their own uplinks, but the end-to-end rate is not doubled because these neighbors receive some overlapping chunks. At four times the base rate, a third receiver now receives enough chunks, and so on.

Link-alike sources therefore dynamically determine a broadcasting rate that will fill the uplink TCP windows of all neighbors *who account for 5% or more of the total throughput to the destination.* Although the choice of 5% is arbitrary, in our test topologies, it provided a good balance between keeping useful neighbors sending while not wasting wireless broadcasts to extremely poor neighbors. To ensure the uplinks do not go dry during transient variations in available wireless bandwidth, we aim to ensure that the buffer of each useful neighbor contains data amounting to 5x its estimated BDP.

To determine the sending rate, the destination counts the number of chunks shipped from each neighbor in every window of 200 chunks. It reports the maximum number of chunks that are needed to reach $5\times$BDP buffer occupancy among neighbors that shipped more than 10 chunks each (5% of the total). The source uses this feedback value, F, to drive a proportional-differential controller that attempts to keep that capacity occupied without overflowing. Every 0.2 seconds the controller computes the new packet broadcast rate

[1] This approach actually estimates the TCP fair-share capacity, not the *spare* capacity. In the future, we plan to investigate using techniques from TCP-Nice [28] or TCP-LP [20] to ensure that opportunistic transfers do not interfere with the uplink owner's own traffic.

as:

$$\begin{aligned} F &= \max_{\text{contributing neighbors}} (5 \times \text{BDP} - \text{buffered}) \\ \text{rate} &= \text{rate}_{\text{old}} + (0.1F + 0.4(F - F_{\text{old}})) \times 1/s \\ F_{\text{old}} &= F \end{aligned}$$

The rate is clipped to be between zero and the maximum packet rate for the wireless bitrate used.

Bit-rate selection. Link-alike's transmissions are not addressed to any one neighbor; hence, no traditional auto-rate algorithm is applicable. In general, the optimal bit-rate selection for opportunistic broadcast is an open problem, and is beyond the scope of this paper. Instead, Link-alike drafts behind the auto-rate algorithm used by unicast transmissions. First, we determine the long-term best bit-rate for each neighbor using the algorithm built into the driver (SampleRate [9]). Subsequently, the bit-rate for each transmission is set to the best bit-rate for the lowest expected time (i.e. ETT metric [16]) to a neighbor with available bandwidth. This heuristic essentially mimics the optimal unicast strategy.

III.C.4. Acknowledging Received Data

As noted above, the source transmits the entire file in order over the wireless medium. We assume that the file is large enough that the time needed to do so is long; as a result, the system uses a completely asynchronous, end-to-end acknowledgment scheme. Link-alike sends these acknowledgments only from the destination to ensure reliability.

Although we use TCP to provide reliable delivery of chunks over wired uplinks, chunks may be lost over the wireless medium or because a neighbor has failed after receiving them. In the approaches that use wireless unicast, the 802.11 transmitter retransmits the packet until it receives an 802.11 ACK from the intended receiver. Broadcast packets have no such mechanism (nor should they—multiple simultaneous responses would interfere with each other). Therefore, in Link-alike, the source does not receive immediate feedback from its neighbors.

When the source transmits a chunk, it selects the not-yet-acknowledged chunk that was broadcast the longest time ago. In practice, this means it cycles through the unacknowledged chunks of the file in order. By permitting a large gap between transmission and acknowledgment, this scheme ensures that the Link-alike source need never stop broadcasting packets to wait for feedback.

Link-alike neighbors do *not* ACK chunks when they receive them. Such an approach might improve efficiency (particularly when there are few chunks remaining) because it would let the source avoid redundantly transmitting a chunk. In practice, however, the combination of the large file size and careful buffer sizing at receivers (as a multiple of their uplink bandwidth-delay product) means that chunks will not reside at a neighbor for long periods of time prior to upload and acknowledgment. Using only end-to-end acknowledgments greatly simplifies handling neighbor failures. When the TCP connection to a neighbor is severed, the destination simply marks all outstanding chunks scheduled to this neighbor as unscheduled.

III.C.5. The End-Game

Like many load-balancing systems, Link-alike must deal with the *end-game* problem that occurs when all chunks of the file have been scheduled to neighbors, but those neighbors may upload at drastically different rates. The system must avoid being bottlenecked by the slowest of these neighbors while other neighbors are idle.

Link-alike deals with this problem in two ways. First, the destination always uses the TCP bandwidth-delay estimate to determine how many chunks to schedule on a neighbor. This mechanism avoids grossly over-committing to a slow receiver.[2] Second, Link-alike copes with failed neighbors by rescheduling all of their chunks (as discussed above). Triggered by a timeout, this method can also deal with neighbors whose capacity drastically drops during the last part of a transfer. As we explore in Section VI, the existing mechanisms perform well in most scenarios, and we leave more advanced end-game strategies for future work.

IV. Implementation

Link-alike is implemented completely in userspace using approximately 2300 lines of C++. It interfaces with the kernel network stack via regular sockets, and accesses the wireless network interface in monitor mode via a raw packet socket to support per-packet bit-rate control and sniffing for unicast. For TCP, the auto-rate algorithm implemented in the wireless driver tracks the best bitrate to communicate with each neighbor. For UDP and Link-alike, however, we use the long-term best bit-rate per neighbor estimated before the experiment.

The source splits the file into 1.4KB chunks so that each chunk fits in a single UDP packet. The source

[2] In fact, our commitment ensures that the end-game lasts no longer than the RTT spread.

assigns each chunk a unique sequence number that is used for acknowledgments and file reassembly. To cope with files larger than the amount of available memory, the Link-alike source keeps only a part of the file in its buffer, which it refills as the chunks are acknowledged by the destination.

IV.A. Neighbors

Each neighbor maintains a bit vector of all sequence numbers to avoid storing and shipping redundant chunks. Whenever a chunk is received over the wireless UDP or ACKed over the TCP from the destination, the corresponding bit is set in the vector. ACKed packets are also evicted from the buffer. Further UDP receptions of that sequence number are discarded.

IV.B. Capacity Estimation Implementation

Neighbors report their buffer state information to the destination with each chunk they upload. This information contains the number of chunks currently stored in the neighbor's buffer and an estimate of the bandwidth-delay product (BDP) from the neighbor to the destination. The BDP is measured in chunks and is estimated using a `getsockopt(SO_SNDBUF)` system call. The destination, in turn, attaches feedback information (Section III.C.3) to each ACK packet it sends to the source, so that the source can compute its broadcast rate.

V. Understanding potential gains

Before empirically evaluating Link-alike, we examine the potential benefit of an opportunistic wireless broadcast scheme for broadband link aggregation.

V.A. System Model

Our analytical model divides the network into two domains, the wireless and the wired. For simplicity, we assume that there are no shared bottlenecks in the wired domain and that the wireless transmitter operates on a fixed modulation (bit-rate) and power. The transmitter broadcasts each packet, after which any successful receiver can relay it to the recipient over its wired uplink. To find the maximum rate of unique packets that could be delivered to the destination, we use a specialized variant of the standard maximum flow problem.

In this flow formulation, the source can attempt to broadcast at most B units of flow (i.e., it is limited by the maximum rate of the wireless network). The source has a set of neighbors, N. Each neighbor $i \in N$ is characterized by its uplink capacity u_i and the probability p_i that it receives a packet from the source. Because wireless losses might be correlated, each *set* of neighbors K also has a probability q_K that at least one node in K receives a transmission. (If all receivers are independent, then $q_K = 1 - \prod_{i \in K}(1 - p_i)$.)

V.B. Opportunistic Capacity Bounds

Let x_i be the rate of flow relayed by neighbor i to the ultimate destination. The total amount of flow sent can be formulated as a linear program (LP), whose objective is to maximize the total number of unique packets that can be delivered to the destination via the neighbors:

$$\max \sum_{i \in N} x_i \tag{1}$$

subject to:

$$\textit{uplink limit} \qquad 0 \le x_i \le u_i \qquad \forall_{i \in N} \tag{2}$$

$$\textit{redundancy limit} \qquad \sum_{i \in K} x_i \le B\, q_K \qquad \forall_{K \subset N} \tag{3}$$

The first limit simply says that no neighbor can transmit faster than its uplink capacity u_i. The second constraint encodes the fact that if multiple neighbors receive a packet via the wireless medium, it is only useful for *one* of them to send the packet to the destination.

There is one instance of the second constraint for each set of neighbors K (the number of these constraints may be exponential in the size of the neighborhood N). The total expected flow of unique packets that reach at least one of them is Bq_K (the wireless bandwidth times the set reception probability). This value is the upper limit on the total amount of unique flow that can be delivered to the destination by *all* nodes in that set.

Note that this formulation does not include the uplink capacity of the source itself. This capacity simply augments the total throughput.

V.C. Unicast Capacity Bounds

To understand the benefits that can be achieved by exploiting wireless broadcast, we compare the opportunistic capacity to that which could be achieved by having the source select a single, best forwarder for each packet [17]. In this case, the total flow through a neighbor i is limited both by that neighbor's uplink capacity u_i and the total wireless capacity at the source, B. If z_i denotes the rate of packets sent to neighbor i, then the LP formulation is again to maximize the total flow sent through all neighbors:

$$\max \sum_{i \in N} x_i \tag{4}$$

subject to:

$$\textit{uplink limit} \quad 0 \le x_i \le u_i \quad \forall_{i \in N} \quad (5)$$

$$\textit{unicast limit} \quad x_i \le z_i p_i \quad \forall_{i \in N} \quad (6)$$

$$\textit{wireless limit} \quad \sum_{i \in N} z_i \le B \quad (7)$$

The first constraint again bounds the amount of flow sent by neighbor i to be smaller than its available capacity. The second constraint takes into account the effect of wireless losses on the amount of flow sent to neighbor i from the source. The neighbor can forward on its uplink the fraction p_i of z_i it receives from the source. The last constraint simply declares that the total amount of flow sent to all neighbors from the source must be smaller than the wireless capacity itself.

The LP formulation for unicast provides some insight on how to implement such a system in practice. Unlike the opportunistic formulation, this problem has a linear number of constraints that can be solved greedily by filling neighbors' capacities (x_i) in order of descending reception probability p_i. In practical terms, this means that the optimal strategy is to forward via the neighbor most likely to receive the packet who still has remaining capacity.

V.D. Potential Capacity Gain

Based on the formulation of Section V.B the potential gain from opportunism depends greatly on the wireless bandwidth B, uplink speeds u_i, and reception probabilities p_i. For simplicity, we examine the case when all neighbors have the same uplink capacity $u_i = u$, and all neighbors independently receive packets with probability $p_i = p$. In this restricted scenario, the capacity bounds become:

$$\text{broadcast} \quad \min\left[Nu, B\left(1-(1-p)^N\right)\right]$$

$$\text{unicast} \quad \min\left[Nu, Bp\right]$$

Figure 3 explores the potential gain from opportunism, i.e., the ratio of the broadcast and unicast bounds, as a function of the available wireless bandwidth B and the reception probability p. Notice that the wired upstream capacity u is set to 1. Thus, varying the value of B allows us to explore the benefits one can reap as the wireless medium capacity increases in relation to that of the wired medium. Figure 3 suggests three distinct operating regimes:

Wired-bottlenecked: As B increases, we eventually enter a region where the received wireless flow exceeds the neighbors' wired capacity. For example, in Figure 3, the line for $N = 10, p = 0.5$ when $B = 20$ may correspond to a very dense (e.g., large apartment

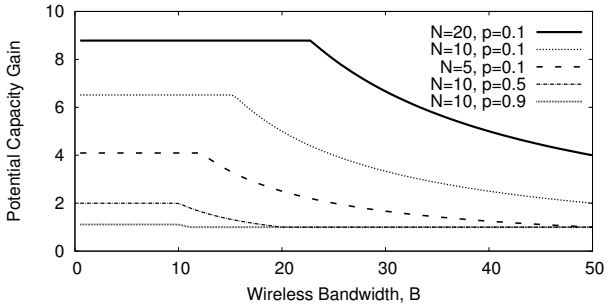

Figure 3: Assuming wired uplink capacity, $u_i = u = 1$ and probability of reception, $p_i = p$, the potential gain is always limited by $1/p$ and converges to 1 as the available wireless bandwidth B increases.

building) deployment (ten neighbors) using a very high performance wireless network (e.g., 802.11n), since the wireless bandwidth is 20x that of the wired bandwidth. At this point, we expect $B * p = 10$ units of traffic to arrive at the neighbors. If the total capacity available at the neighbors is $N * u = 10$, then additional wireless bandwidth would not increase traffic flow at the destination, since all broadband links are saturated. As a result, the gain at this point drops to 1. If wireless capacity is abundant, such that we can easily saturate the wired links, then opportunism provides little benefit.

Wireless-bottlenecked: When B is low, the potential gain from opportunism is $\left(1-(1-p)^N\right)/p$. In this case, opportunistic broadcast minimizes the bandwidth cost of delivering packets to the neighbors, who have excess wired capacity that can be used to then forward data to the destination. The gain from opportunism is larger when the loss rate is higher, since the system has nothing to gain from opportunism if all receivers perfectly hear every transmitted packet.

In-between: When B is not too low but not high enough to saturate the wired links, then the opportunistic gain reduces in response to increased wireless bandwidth availability (e.g., when $10 \le B \le 20$ for $N = 10, p = 0.5$). The length of this region depends on the actual values for N and p. The reason behind the decrease in gain is that as the wireless bandwidth increases, unicast can be more successful at bridging the gap and utilizing a higher fraction of the available wired resources.

Practical wireless deployments have been shown to suffer from high loss rates [10]. Consequently, we believe that opportunism will play a significant role in actual deployments. This simplified analysis suggests that in wireless-bottlenecked topologies, opportunistic broadcasting can significantly improve throughput. In the next section, we evaluate whether these gains hold

Parameter	Setting
one-way latency	25ms, 50ms, 75ms
bandwidth	400 Kbps, 800 Kbps, 1200 Kbps
maximum queue	bandwidth × 1s
maximum burst	15 KB

Table 1: Parameters for the wired uplink shaping in our experiments. Whenever there is a choice, we pick a value at random with equal probability.

in practice as well as in theory.

VI. Evaluation

In this section, we use measurements from our wireless testbed to evaluate the performance of Link-alike. First, we validate our implementation against the theoretical capacity computed according to the framework developed in the previous section. Second, we compare the performance of Link-alike against that of the unicast schemes. Finally, we study how the throughput gain is affected by changing topological features (e.g., the channel, the number of nodes, etc.) that would vary widely in actual deployments.

VI.A. Testbed

The testbed consists of 8 wireless nodes deployed on one floor of an office building. Each node is a small Soekris net4826 computer equipped with an Atheros 802.11a/b/g wireless card. These nodes and an additional machine that acts as the destination are connected to a 100 Mbps Ethernet switch. The wireless nodes run Linux 2.6.21 with the supplied Madwifi 0.9.4 drivers configured in ad-hoc (IBSS) mode (for TCP) or monitor mode (for UDP and Link-alike) in 802.11b. [3]

To emulate residential connectivity, the traffic between the destination and the nodes is shaped using the Linux bandwidth (`tbf`) and latency (`netem`) shaping modules. To match the characteristics of residential Internet access we pick the parameters according to Table 1.

VI.B. Methods

In each experiment, we transfer one file to completion. The size of the file is scaled with the aggregate wired capacity so that each transfer lasts at least 1 minute.

[3]Note that the special operating modes do not conflict with regular AP functionality since the driver supports software virtualization of the hardware NIC, a feature commonly exploited in wireless mesh networks.

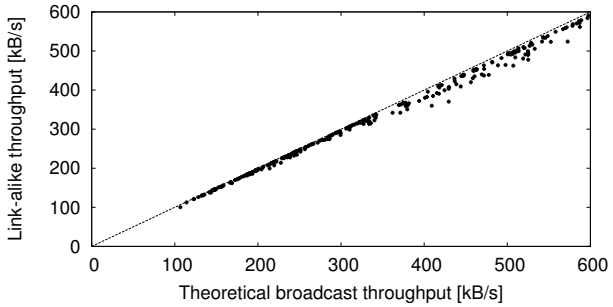

Figure 4: Observed experimental throughput compared to the theoretical prediction.

We assess the performance of the different distribution mechanisms in terms of two metrics: 1) **overall throughput**, defined as the number of unique chunks delivered per second to the destination, and 2) **wireless cost** per chunk, defined as the total amount of wireless "air-time" required for the file transfer divided by the number of chunks delivered via the neighbors. In the case of TCP, this includes both link-layer and end-to-end retransmissions and acknowledgments. Both metrics include the overhead of TCP/IP packet headers.

The computation of the theoretical capacity is based on actual measurements and depends on the accurate computation of B, u_i and q_K (for the LP problems (1) and (4)). B can be easily computed as the number of chunks sent in the wireless medium per second. u_i is computed from the nominal bandwidth applied at the wired uplink shaper accounting for the TCP/IP headers. Finally, q_K is computed from the number of times a wireless transmission was received by at least one neighbor in set K. Note that, for increased accuracy, we do not assume that the bit errors are independent.

Finally, in order to explore a breadth of network configurations using a single network testbed, we vary four parameters of the network configuration. We quantify the individual performance gains of each resulting topology. The four parameters we vary are: (i) the number of nodes, (ii) the channel (among the 802.11b orthogonal channels), (iii) the wired uplink capacity, and (iv) wired latency. We present below results from 200 runs that allow us to study the gains of Link-alike under a variety of conditions.

VI.C. Capacity Bounds

Before proceeding with the actual performance comparison, we first validate Link-alike's performance against the estimated potential gain from Section V. Because the theoretical analysis did not include bit-rate selection, we run only this experiment with a fixed bit-rate.

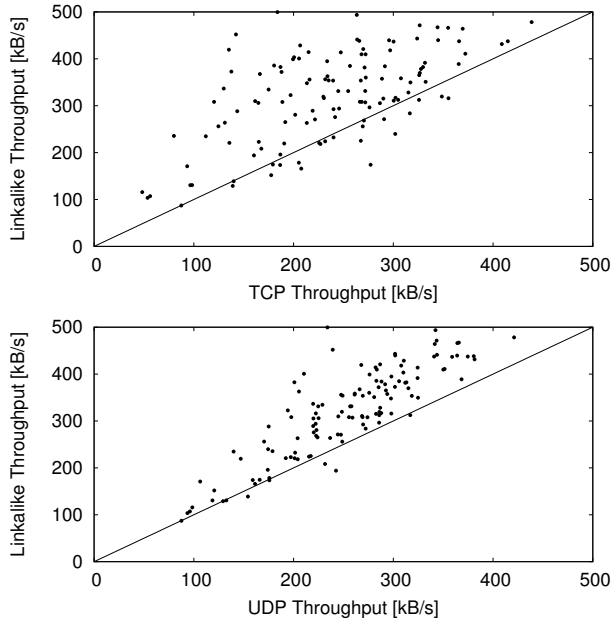

Figure 5: The measured throughput of Link-alike versus TCP and UDP. Each point represents two back-to-back experimental runs with the same topology parameters.

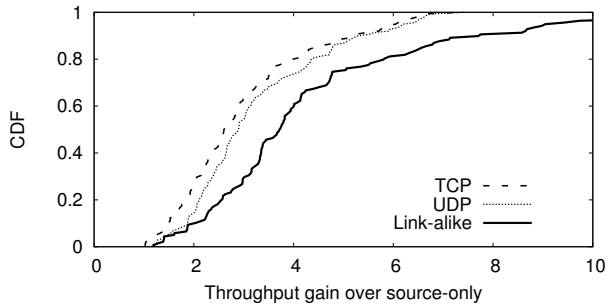

Figure 6: The CDF of the overall throughput gain of the three compared schemes against the source's wired uplink alone.

Figure 4 shows that our implementation of Link-alike closely matches the throughput estimated using our model. In fact, 95% of the two hundred runs achieve 90% or more of the theoretical throughput.

VI.D. Broadcast Gain

Next, we quantify the benefit of Link-alike in contrast to unicast. For each of the 200 network configurations, we perform a file transfer using TCP, UDP unicast and Link-alike, one immediately after another. Figure 5 compares the three schemes in terms of overall throughput. Link-alike consistently outperforms both TCP and UDP. In addition, the percentage gain from Link-alike appears independent of the actual throughput value.

In Figure 6, we show the overall throughput gain of each scheme over using just the wired uplink of the source. The median throughput improvements are 2.6x with TCP, 2.9x with UDP unicast, and 3.7x with

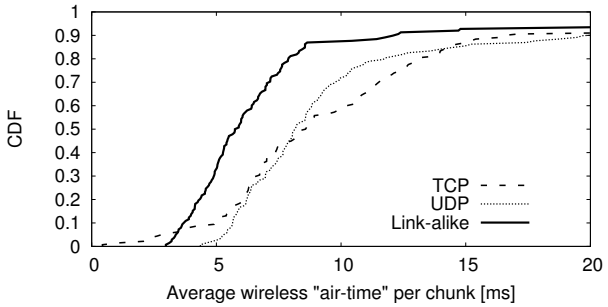

Figure 7: The CDF of the average wireless air-time per chunk delivered via neighbors.

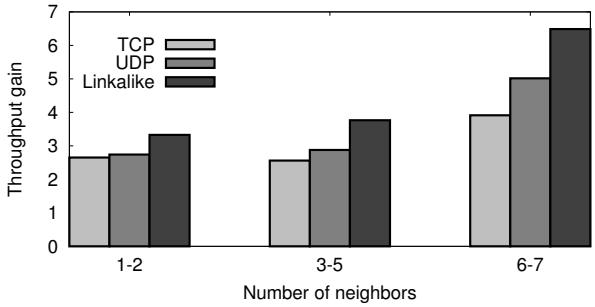

Figure 8: The plot shows the overall throughput gain of each scheme depending on the number of neighbors.

Link-alike. These results show the effectiveness of Link-alike's wireless-specific approach: through opportunistic broadcast, Link-alike provides 28% better throughput than UDP unicast, and through the combination of opportunism and avoiding poor TCP behavior in the lossy environment, it provides a 43% improvement over TCP. These improvements occur despite the broadcast approach having (at present) inferior bitrate adaptation than TCP unicast—the latter adjusts the bitrate for each packet using the state-of-the-art SampleRate algorithm, instead of using a fixed rate throughout the entire experiment.

Another important point in the comparison of Link-alike and the unicast approach is that the aforementioned performance benefits are associated with significantly reduced wireless costs. Figure 7 shows the CDF of the wireless cost computed as the ratio of total wireless air-time consumed to the number of chunks delivered via neighbors. The transmission time of a 1500 byte frame at 5.5 Mbps with no retransmissions is 3.3ms. In our experiments, the median cost of a chunk sent using TCP is 8.7ms, which is 47% higher than the 5.9ms needed by Link-alike. Link-alike substantially reduces the amount of airtime required to successfully upload a file.

So far, we have looked at the statistical gains one should expect from Link-alike across a range of scenarios. In Figure 8, we break down those gains based on the number of neighbors assisting in the file distri-

bution. As expected from analysis, using more neighbors increases the throughput gain from Link-alike versus unicast. More neighbors provide more wired capacity, which means that the system is more wireless-bottlenecked, the regime in which Link-alike excels by opportunistically using *any* neighbor that correctly receives a chunk. In contrast, given limited wireless bandwidth, unicast can only saturate the top few neighbors with the highest link quality. TCP is even more limited by its congestion control algorithm and ends up using only a subset of the available neighbors.

VI.E. Opportunistic Failure Tolerance

Opportunism in Link-alike not only leads to appreciable performance improvement and reduction in wireless cost, but has the potential to offer increased robustness in the presence of link and node failures. Each broadcast is forwarded by *any* neighbor selected by the central scheduler. Fluctuations in the number of available neighbors change the potential for throughput, but will not jeopardize the operation of Link-alike itself. We explicitly tested this scenario by randomly killing neighbor Link-alike processes and restarting them at a later time. Link-alike worked correctly and seamlessly: when new neighbors appeared, Link-alike opportunistically started using them; when neighbors disappeared, Link-alike stopped using them and rescheduled the appropriate blocks, effectively masking out any network failures.

VII. Discussion and Future Work

This section discusses several design decisions and challenges related to this work.

Incentives. Link-alike requires a number of nearby neighbors that are willing to participate in the system. However, user concerns about issues such as liability for illicit content uploaded across their access link may prevent broad adoption of systems such as Link-alike. One sign that this may not be an issue is the popularity of systems such as FON [3], in which users provide guests access to their broadband connections. Similarly, the increasing popularity of community wireless networks [2, 4] suggests that users are willing to share the wired connectivity in their neighborhood. Technical solutions that assign liability more appropriately [26] may address many user concerns.

Wireless channels. Broadcast requires that the source and the neighbors operate on the same channel. This shared channel might therefore be highly congested. Moreover, extending the design to multi-source or multi-hop scenarios means that the system will contend with itself for the wireless medium.

One simple approach to help alleviate this wireless contention is to use multiple radios. Each could operate on an independent channel, but at least one would use a common neighborhood-wide channel. As radios become cheaper, commodity access points may start shipping with multiple radios, or users may simply purchase two access points (one for use with Link-alike and one for in-house connectivity). A second radio is likely to cost less than a single month of high-speed broadband connectivity, making this option viable. BT's Home Hub addresses this issue by using a dedicated FON SSID; a similar solution could be used by Link-alike.

Security. The design presented above does not address security. We assume that the source and destination use standard end-to-end security techniques to achieve confidentiality and integrity if desired. These techniques allow for a threat model in which the neighbors and the network are untrusted. Note that the neighbors cannot perform a denial-of-service attack by dropping chunks. The transfer is opportunistic; if the neighbors drop packets, the source itself will eventually just ship all of the chunks over its own upstream link.

Sending chunks through untrusted neighbors does, however, leak some information. For example, the neighbors may discover both the destination and size of transfers. Note that recent studies [25] have shown that wireless networks leak significant amounts of similar information even when using encryption and not using Link-alike.

The system design presented above assumes that the source can broadcast chunks to neighboring access points. If these access points use wireless security mechanisms such as WEP or WPA, the source will not be able to communicate with them. A potential solution is to share a "Link-alike key" among neighbors in one of the additional WEP key slots in each access point, to concurrently permit both WEP and non-WEP access, or to use a separate access point for Link-alike.

Common neighborhood scenarios. Link-alike's ability to improve upload depends on the number of neighbors, wireless link bandwidth, wireless packet loss rates and wired network bandwidth. An obvious question is whether Link-alike performs well in the conditions observed in current neighborhoods.

Although a detailed evaluation of wireless network conditions is beyond the scope of this paper, we did perform a small scale measurement study. We asked 30 people to collect wireless connectivity measurements from inside their homes. They recorded the wire-

less neighborhood information (AP id, signal strength, noise level, RSSI, etc.), every second for ten minutes (typical NetStumbler output). The users lived in both rural and urban areas, in houses and apartments. This initial collection shows that in today's neighborhoods, a home AP may be able to reach between 0 and 31 different APs at a variety of link qualities. Only rarely was a neighbor's AP available at the highest transmission rate. The average number of APs within reach was 5-6. While limited in scope, this study confirms the existence of high-density wireless neighborhood environments that enable Link-alike to facilitate higher capacity uploads. In addition, based on the collected measurements, we expect loss rates in such environments to vary greatly, which is one of the scenarios in which Link-alike provides the most performance benefit. Finally, the wired network bandwidth numbers used in our evaluation were based on typical DSL configurations in the United States.

A related concern is that technology trends will change neighborhood connectivity such that Link-alike may no longer be effective. For example, 802.11n promises speeds up to 200 Mbps and improved range. Similarly, several companies have begun to provide fiber-optic connections to the home (e.g., Verizon's FIOS) that offer an order-of-magnitude improvement in upload speeds (e.g., 5–10 Mbps). As shown in Figure 3, the techniques presented in this paper are still useful as long as the ratio between the wired and wireless bandwidth remain high enough.

Backward compatibility. Our design for aggregating broadband connections via wireless is specific to whole-file transfer where pieces of the file can arrive out-of-order. Limiting the design to this context allows each neighbor to establish independent TCP connections to the destination machine. Thus, each neighbor traverses any NAT or firewall that it is behind, as usual, to reach the destination and deliver the data. The destination runs a user-level file transfer application to receive these TCP connections puts the file back together, but no kernel or TCP-level stream re-assembly is necessary.

Our current focus with Link-alike is understanding the potential gains from aggregation. To do so, we deliberately focused on bulk file transfer and permitted the requirement that the server participate in the protocol.

Shared bottlenecks and Acceptable Use Policies. There are two concerns about using Link-alike with commercial ISPs. First, cable modem connections (unlike DSL) may share the same trunk line to the ISP. Two neighbors might thus share the same limited trunk capacity, rendering Link-alike ineffective. In practice, however, cable modem devices rate-limit connections more aggressively than the limited trunk bandwidth. This issue of rate-limiting, however, brings up a more important point: acceptable use policies.

Many consumer ISP user agreements prohibit users from selling or even giving away their network bandwidth. We note two encouraging trends in this area: First, many smaller ISPs (e.g., Speakeasy) permit users to re-use their capacity. Second, some larger ISPs (e.g., British Telecom) have drafted policies to explicitly permit users to participate in Wi-Fi sharing services [5].

Multi-source. Link-alike was designed and evaluated for a single source. We also plan to explore the challenges that arise in multi-user environments. These scenarios are complicated both by the potential for wireless collisions and by the opportunity to use neighbors as helper nodes for more than one source. Providing fairness in these environments may be particularly difficult.

Multi-hop. While Link-alike offers significant benefits in high density scenarios, we expect that it could also lead to appreciable gains even in less dense networks. In this case, however, helper nodes will need to decide whether to forward or drop a packet, but also whether they should re-broadcast it to neighbors that are more than one hop from the source.

Downloads. Our design focused on uploading data from the neighborhood to an Internet server. Aggregating broadband connections may also improve the speed of network downloads. Opportunistic broadcast, however, is less likely to be useful in such settings since there is exactly one node that desires the downloaded data and speeding up downloads is less critical due to the highly asymmetric nature of most broadband links.

VIII. Conclusion

Link-alike is a simple and robust protocol that uses opportunistic broadcast to exploit wireless diversity in neighborhood networks and enable services that require higher upstream capacity through the aggregation of multiple broadband links. Link-alike is tailored to the specific challenges of neighborhood wireless: it takes advantage of wireless broadcast to avoid unnecessary retransmissions, it provides a novel broadcast rate adaptation algorithm to make efficient use of the wireless and avoid the problems of TCP performance on lossy links, and its design shifts small control packets away from the wireless medium to the relatively free wired downlinks.

Our experiments in a physical wireless testbed indicate that Link-alike not only outperforms the traditional unicast approach, but also comes very close to the theoretical capacity bounds provided in this paper. Importantly, Link-alike also greatly reduces the number of wireless transmissions necessary to achieve that capacity. Our results demonstrate that Link-alike delivers a 3.7x improvement in median throughput compared to a single wired uplink, over tests with an average of 4.2 neighbors. It significantly outperforms prior sharing solutions popular in both academia and commercial products, offering 28% improvement over UDP unicast, and 43% improvement over TCP striping. Trends in community wireless networking and the support by AP manufacturers and Internet Service Providers makes the problem of broadband link aggregation very timely. Link-alike offers such a solution.

References

[1] Broadband Bonding. `http://www.broadbandbonding.com`.

[2] Community Wireless Solutions. `http://www.cuwin.com/`.

[3] FON. `http://www.fon.com/en/`.

[4] FreeNetworks. `http://freenetworks.org/`.

[5] The Wi-Fi community built by you. `http://www.beta.bt.com/apps/openwifi`.

[6] WiBoost. `http://www.wiboost.com`.

[7] OpenWrt supported hardware. `http://wiki.openwrt.org/TableOfHardware`, July 2008.

[8] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In *Proc. ACM SIGCOMM*, Stanford, CA, Aug. 1996.

[9] J. Bicket. Bit-rate selection in wireless networks. Master's thesis, Massachusetts Institute of Technology, Feb. 2005.

[10] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *Proc. ACM Mobicom*, Cologne, Germany, Sept. 2005.

[11] S. Biswas and R. Morris. ExOR: opportunistic multihop routing for wireless networks. In *Proc. ACM SIGCOMM*, Philadelphia, PA, Aug. 2005.

[12] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *Proc. ACM SIGCOMM*, Kyoto, Japan, Aug. 2007.

[13] R. Chandra, P. Bahl, and P. Bahl. MultiNet: Connecting to multiple IEEE 802.11 networks using a single wireless card. In *Proc. IEEE INFOCOM*, Hong Kong, Mar. 2004.

[14] B. Cohen. Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, USA, June 2003.

[15] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *Proceedings of the 20th International Conference on Data Engineering (ICDE)*, page 449, Washington, DC, USA, 2004. IEEE Computer Society.

[16] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. ACM Mobicom*, San Diego, CA, Sept. 2003.

[17] H.-Y. Hsieh and R. Sivakumar. pTCP: An end-to-end transport layer protocol for striped connections. In *IEEE International Conference on Network Protocols (ICNP)*, Paris, France, Nov. 2002.

[18] M. Ihmig and P. Steenikiste. Distributed dynamic channel selection in chaotic wireless networks. In *13th European Wireless Conference*, Paris, France, Apr. 2007.

[19] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi. FatVAP: Aggregating AP backhaul capacity to maximize throughput. In *Proc. 5th USENIX NSDI*, San Francisco, CA, Apr. 2008.

[20] A. Kuzmanovic and E. W. Knightly. TCP-LP: A distributed algorithm for low priority data transfer. In *Proc. IEEE INFOCOM*, San Francisco, CA, Apr. 2003.

[21] P. Larsson. Selection diversity forwarding in a multihop packet radio network with fading channel and capture. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(4):47–54, 2001.

[22] Meraki Wireless Network. `http://meraki.com/`.

[23] A. K. Miu, H. Balakrishnan, and C. E. Koksal. Improving loss resilience with multi-radio diversity in wireless networks. In *Proc. ACM Mobicom*, Cologne, Germany, Sept. 2005.

[24] S. Nath, P. B. Gibbons, S. Seshan, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, MD, Nov. 2004.

[25] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall. 802.11 user fingerprinting. In *Proc. ACM Mobicom*, Montreal, Canada, Sept. 2007.

[26] N. Sastry, J. Crowcroft, and K. Sollins. Architecting citywide ubiquitous Wi-Fi access. In *Proc. 6th ACM Workshop on Hot Topics in Networks (Hotnets-VI)*, Atlanta, GA, Nov. 2007.

[27] N. Thompson, G. He, and H. Luo. Flow scheduling for end-host multihoming. In *Proc. IEEE INFOCOM*, Barcelona, Spain, Mar. 2006.

[28] A. Venkataramani, R. Kokku, and M. Dahlin. System support for background replication. In *Proc. 5th USENIX OSDI*, Boston, MA, Dec. 2002.

[29] G. R. Woo, P. Kheradpour, D. Shen, and D. Katabi. Beyond the bits: cooperative packet recovery using physical layer information. In *Proc. ACM Mobicom*, Montreal, Canada, Sept. 2007.