BOLT: Realizing High Throughput Power Line Communication Networks

Ahmed Osama Fathy Atya*, Karthikeyan Sundaresan†, Srikanth V. Krishnamurthy*, Mohammad A. Khojastepour† and Sampath Rangarajan†

*University of California, Riverside, †NEC Labs, USA

{afath001, krish}@cs.ucr.edu, {karthiks, sampath, amir}@nec-labs.com

ABSTRACT

Power line communications (PLC) offer an immediate means of providing high bandwidth connectivity in settings where there is no in-built network infrastructure. While there is recent work on understanding physical and MAC layer artifacts of PLC, its applicability and performance in multi-flow settings is not well understood. We first undertake an extensive measurement study that sheds light on the properties of PLC that significantly affect performance in multi-flow settings. Using the understanding gained, we design *BOLT* , a framework that adopts a learning-based approach to effectively manage and orchestrate flows in a PLC network. *BOLT* is flexible and is agnostic to standards; it can be used to implement scheduling algorithms that target different performance goals. We implement *BOLT* on three different testbeds using off-the-shelf PLC adapters and showcase its ability to effectively manage flows, delivering several folds throughput improvement over state-of-the-art solutions.

CCS Concepts

•**Networks** → **Network dynamics;** *Network experimentation; Physical topologies;*

Keywords

Powerline Communication; Experiments; Testbeds; BOLT

1. INTRODUCTION

Power line communications are attractive for providing backhaul Internet connectivity in settings without an in-built network infrastructure, especially in third world countries. Several retail segments such as healthcare, industrial automation and warehousing, are increasingly relying on Internet connectivity [1], [2] and in many such segments, deploying an Ethernet backhaul from scratch may not be cost effective. Furthermore, in many such applications, there is a large amount of local, peer to peer data transfers that are required (e.g., health records between medical devices and IT servers *and* remote patient monitoring feeds in hospitals, video surveillance data between cameras and storage servers in retail warehouses) for which, PLC can be attractive.

PLC research is yet to mature: While PLC commercial adapters are now available for home applications [3], they primarily target low throughput unsaturated traffic flows. The viability of using PLC as an alternative for delivering Ethernet like throughputs in multi-flow settings has received little attention. Recently, there has been some work done on understanding the efficacy of the 1901 MAC (Medium Access Control) protocol, which is the basis for access control in commercial PLC adapters [4, 5, 6]. These efforts showcase the short comings of 1901 and suggest the tuning of a few protocol parameters towards improving its efficiency. However, tuning of such parameters on commercial adapters is not viable today. Moreover, 1901 has other performance issues that we showcase later. Thus, we ask the question "How can we achieve high stable throughputs with PLC in multi-flow settings in a completely standards agnostic manner?"

Challenges: There are three main challenges that we will need to overcome to answer the above question. **First,** the topology of a PLC network is often unknown since it is hidden behind walls and the connectivity is typically established without communications in mind; in fact, nodes that are geographically close are not necessarily direct neighbors. The network structure dictates which transmissions interfere with each other. While the 1901 MAC resolves this issue to some extent, it can lead to poor throughput as well as unfairness in many cases. To drastically reduce the ill effects of interference, an understanding of the network topology needs to be derived. **Second,** the quality of the PLC channel is time varying. The impedance loads on the PLC lines vary as electrical devices that are plugged in, and are turned ON or OFF. This causes the throughputs on certain links to either degrade

DOI: 10.1145/1235

or improve, thereby causing dynamics in the network topology. The time scales of these dynamics will influence the solutions that one can deploy for effective management of flows. As shown later, the 1901 MAC does not account for these variations and this contributes to its inefficiency. The **third** related challenge is to resolve the first two challenges in a standards-agnostic way. This requires lightweight solutions above the MAC layer that are sufficiently adaptive so as to cope with the PLC channel dynamics.

Contributions: (1) Towards addressing the above challenges, we undertake an extensive application (flow) level measurement study. While our study leads to many interesting observations, two are especially noteworthy. First, the study shows that flows do indeed interact in a PLC network in unpredictable ways. Some flows can co-exist simultaneously, and their joint activation can yield significant throughput gains compared to when they are activated in isolation. However, in other cases, joint activation of flows can hurt the throughput compared to when they are sequentially activated. Second, the study suggests that the PLC channel is *quasi-stationary*. This is an artifact of these variations arising from electrical devices being turned on/off, which does not happen at very fine time scales and all that often.

(2) Based on the understanding gained from the above study, we design *BOLT* , a flexible framework that appropriately configures the PLC network to derive high throughputs while adhering to whatever fairness constraints are desired. Towards this, *BOLT* does the following:

- Determining flow interactions via calibration measurements can be expensive; if done naively it can result in an exponential number of measurements. *BOLT* imposes a logical structure on the PLC network using which, it is able to drastically reduce the number of such measurements needed for flow management (now polynomial with respect to the number of nodes in the network).
- *BOLT* leverages the quasi-stationarity of the PLC channel to intelligently apply machine learning (ML) classifiers to determine flows that can be simultaneously active, and their potential throughputs. The approach (i) has high accuracy with small amounts of training data and (ii) is resistant to noise from the plugging in of electrical devices.
- Finally, based on the above estimation, *BOLT* aggressively reuses the available channel capacity by scheduling as many simultaneous flows as possible (at the granularity of time epochs) while adhering to desired fairness constraints. This scheduling is done at the application layer and the goal is to limit the level of contention that has to be handled by the 1901 MAC. This will reduce collisions due to hidden terminals or unnecessary backoffs, both of which if not dealt with, lead to loss in throughput and unfairness.

(3) We extensively evaluate *BOLT* on three different testbeds to show that it provides several fold throughput improvements (1.5x - 8.5x) over the state of the art solutions (including those in current day adapters), while remaining standards-agnostic. Further, we show that the algorithms within *BOLT* strike a good balance between performance and complexity.

2. BACKGROUND AND RELATED WORK

In this section, we first provide relevant background on PLC. Subsequently we discuss related studies.

PLC Channel: PLC operates in the 1.6 and 86 MHz bands. Its channel [7] is similar to the wireless channel: (a) the signal is attenuated due to cable losses and, (b) multipath propagation occurs due to cable branching and unmatched line ends. The multipath reflections depend on the electrical devices hooked on to the network and their impedance. Studies such as [8] and [9] have built channel models for indoor PLC. However, they do not measure or characterize the properties of interest such as (i) the interactions between flows in terms of simultaneous activation and (ii) the time scales of throughput variations due to channel dynamics.

PLC Adapters: Today's popular PLC adapters are Homeplug AV1, Homeplug AV2 and HomePlug GreenPHY [3]. Using 1155 OFDM subcarriers and turbo codes, AV1 operates nearly at the theoretical maximum rate (ranges from 14 to 200 Mbps). AV2 uses an additional bandwidth from 30-86 MHz and a MIMO like PHY to achieve data rates of up to 1.5 Gbps. The MAC protocols used in these adapters, are variants of 1901 which in turn is based on CSMA/CA [10]. Typically, each PLC network has a central controller that helps impose a time-slot structure to facilitate the use of 1901.

Related Work: The work in [10] models CSMA/CA for PLC. In [11], the authors perform limited experiments to gather some insights on the throughput of UDP and TCP over a PLC network. The authors in [12], try to characterize the end to end throughput over PLC. These limited efforts however, have not explored multi-flow environments or mechanisms to improve the PLC system performance.

There is recent work on analyzing the throughput performance of the PLC MAC protocol (based on IEEE 1901) [6, 5, 4]. The authors show that collisions and unnecessary backoffs are worse with 1901 as compared to 802.11 [4]; this is because (i) of the large slot duration and (ii) nodes increase their backoff times when they sense the channel to be busy (unlike in 802.11), with 1901. The authors argue that by tuning certain back-off parameters, the throughput of 1901 can be maximized. However, setting these parameters in commercial devices with no access to the firmware is difficult. Furthermore, the analysis does not take into account the loss in performance due to dynamics of plugged in electrical devices. Our goal is to design and develop an adaptive framework that allows flexibility in such scenarios, and reduces MAC contention, by imposing a schedule at a higher layer.

Note that while techniques from wireless (PHY, MAC) measurement studies (eg. [13]) can be borrowed and applied in the PLC context, this would (i) require support from the PLC adapters (not available today) and (ii) more importantly, result in changes to the access protocol (the 1901 standard)

	Mircowave OFF		Mircowave ON	
	Isolation	Contention	Isolation	Contention
τ_{l1}	95	85	29	29
τ_{l2}	96	75	92	63

Table 1: Variation in throughput due to connecting and disconnecting devices at outlet U.
to address the previously mentioned issues.

3. UNDERSTANDING PLC

To understand the factors that influence PLC throughputs, we undertake an extensive measurement study with commercial PLC adapters in a multi-flow environment. Our goal is not to characterize the PLC channel from the physical layer perspective as in prior efforts, but to understand how channel dynamics and flow interactions influence throughputs from an application layer perspective. Our measurements are at the granularity of flows; flows are from a source to a destination and could potentially encounter multiple power outlets en route. Flow contention is handled by the MAC protocol (1901 based) in all of the experiments reported here.

We perform experiments on three different testbeds. The first is at an enterprise office setting (ENT), the second is at a university lab (UNI testbed), and the third is in a residence (house). Details of the testbed setups are in Section 6. We employ PLC adapters from multiple vendors for diversity.

How do electrical appliances affect the throughput of a PLC network?: Switching on electric apparatus (plugged into power outlets in the PLC network) injects noise onto the channel [14], which degrades the throughput. Towards quantifying the performance degradation due to plugged in electrical devices, we first perform a controlled experiment where we vary the number of electrical devices that interfere with PLC transmissions. In Fig. 2, we plot the throughput of a flow in the presence (or absence) of different apparatuses. The distance between the source and destination was ≈ 2 m. The electric apparatus (fluorescent lamps, Dell laptops, small microwave ovens, printer) are connected to a power outlet that is 80 cm from the destination power outlet. Using *iperf* [15], a UDP flow was generated between the source and the destination. It is hard to a priori know the capacity of links in a PLC network; wire material and age will dictate the maximum throughput of a line. Thus, we start with a relatively high UDP flow rate (500 Mbps) and gradually reduce the rate until we observe no losses (this is the estimated line capacity). The results are averaged over 20 runs, each lasting 60 seconds. As seen from the figure, lightweight devices such as fluorescent lamps or laptops do not inject much noise and thus, barely hurt the throughput. However, microwave ovens or printers (HP LaserJet 4200 in our study) are heavyweight in terms of the noise they inject and significantly hurt the throughput. It is interesting that as the number of devices of a certain type increase, the throughput degradation appears to be linear; thus, after an initial calibration phase, the impact of additional devices (of the same type) can be empirically estimated (although we do not use this property explicitly in *BOLT*). The key takeaway here is, the fact that different brands/types of electric apparatuses project different levels of noise (based on their electric load), makes it challenging to predict the throughput of a flow.

Microscopic study: Towards understanding the impact of electrical appliances on 1901, we construct a controlled topology with five power outlets as shown in Fig. 3 in a residency; sources A and C, and destinations B and D, are connected via AV2 adapters to the outlets as shown. Power outlet U is unused in one set of experiments (setup 1); in a second set of experiments (setup 2), an active microwave oven (power 650 watts) is plugged into U (similar effects were seen with our HP LaserJet 4200 printer but the results are not shown because of space constraints). The results from the experiments are tabulated in Table 1. The average throughputs of the flows from A to B and from C to D in *isolation* are 95 Mbps and 96 Mbps, respectively in setup 1. In setup 2, they are 30 Mbps and 92 Mbps respectively. It is evident that the flow from A to B is significantly affected by the microwave oven, but not the one from C to D. When the two flows are activated *simultaneously* in setup 1, the throughputs achieved by the flow from A to B and that from C to D, are 85 Mbps and 78 Mbps, respectively; notice that the 1901 MAC decreases the throughputs of the individual flows, but provides some semblance of fairness. The overall throughput is higher and thus, this may be desirable. However, with setup 2, while the throughput of the flow from A to B remains almost unchanged compared to isolated operations (29 Mbps), the flow from C to D takes a significant hit (throughput drops to 63 Mbps) during concurrent operation. Thus, even though the microwave by itself does not influence the latter flow, the interactions with the flow that is affected degrades its performance when 1901 is used (the two flows share the capacity and the flow affected by noise eats into the capacity of the flow that is relatively unaffected). Specifically, if the poor quality flow accesses the channel, it can cause the good quality flow to repeatedly back off. The overall throughput is still higher than if the flows were sequentially activated but this may not always be the case (as seen in later experiments in uncontrolled settings). Thus, there is a need to account for these interactions, and depending on whether the throughput increases or decreases, switch from concurrent activity to sequential activity or vice versa when electrical devices are turned ON/OFF.

How can one expect the throughput on a PLC channel to change with time?: Figs. 4-6 presents the dynamics in the throughput measured over different time scales in various settings. The data is from 150 to 250 arbitrarily chosen flows in three different environments (other flows exhibited similar behaviors): ENT offices, ENT cafeteria and a UNI laboratory. In Fig 4, we show the average throughput of the considered flows over hours. The noise due to devices in ENT offices is extremely low compared to that in the other cases; this is because the electrical equipment here mostly consists of lamps and laptops. In the cafeteria, the

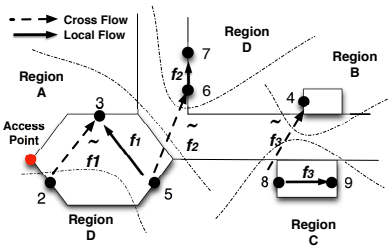

Figure 1: Illustration of a PLC topology.

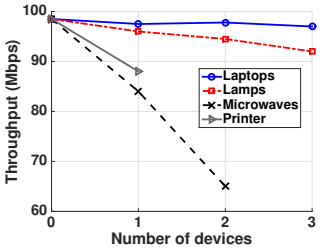

Figure 2: Impact of electric apparatuses.

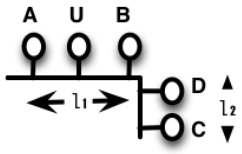

Figure 3: Controlled Topology

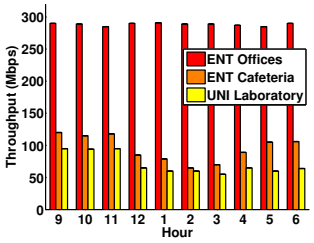

Figure 4: Dynamics over hours.

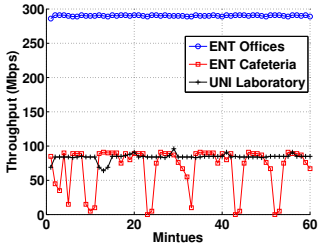

Figure 5: Dynamics over minutes.

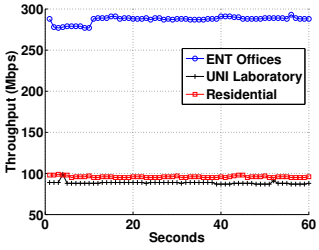

Figure 6: Dynamics over seconds.

throughput is generally lower due to more heavyweight appliances (e.g., refrigerator) but is generally steady. However, a throughput degradation is noticed (Fig. 4) due to the usage of devices such as the microwave oven, coffee maker, etc. during the lunch period. The throughput in the UNI lab is again lower compared to the ENT offices due to the presence of devices such as printers and heavy duty servers which are typically on. After about 11:00 am, when students start using their research equipment (e.g., switch on their desktop machines or other servers) a throughput degradation is seen.

A minute-by-minute depiction of channel fluctuations during the lunch period at the ENT cafeteria (12:00 - 1:00 pm) is shown in Fig. 5. We see that while the average throughput is similar to that in a relatively static setting (UNI lab), operation of electric apparatus can cause significant variations in noise and therefore the achieved throughput, at these short time scales (order of minutes). Our experiments did not reveal any variations at even finer time scales (seconds or milliseconds). We conclude that in general, the PLC channel experiences only slow fading and is *quasi-stationary* in nature unlike wireless (with fast fading); however, it can exhibit large variations in short time scales (order of minutes) in dynamic environments during certain times of the day.

Does proximity between the transceiver pair imply high throughput?: Next, we measure the throughputs between all possible pairs from our 16 node UNI testbed. We measure the geographical distance between a pair of outlets, and the throughput between the outlets in isolation. We classify the throughputs into two categories: HIGH if it is $> 80Mbps$, and LOW otherwise (the line capacity was $\approx$ 130 Mbps). From Fig. 7 we see that proximity does not always translate to a high throughput. More importantly even distant nodes could enjoy high throughputs (e.g., when they are more than

Individual Throughput		
f_1	f_2	f_3
291	95.6	236
$\tilde{f}_1$	$\tilde{f}_2$	$\tilde{f}_3$
42.9	11.8	11.3

LZF (pairwise)			
	f_1	f_2	f_3
f_1	NA	379	331
f_2	379	NA	221
f_3	331	221	NA

CZF (pairwise)			
	f_1	f_2	f_3
$\tilde{f}_1$	NA	30.9	18.39
$\tilde{f}_2$	30.9	NA	11.6
$\tilde{f}_3$	18.39	11.6	NA

Table 2: Throughputs (Mbps) for Fig. 1.

6 or even 12 meters apart). This demonstrates that one cannot determine which transceiver pairs are likely to be adjacent in the network topology just based on geographical proximity (e.g., as will be the case in wireless). Note that wiring diagrams of buildings do not reflect the actual electric connectivity because of: (i) connecting different apparatuses changes the actual electric load (ii) repairs and updates of the electrical wiring constantly change the wiring diagram.

Do flow interactions increase or decrease throughputs with 1901 ? We consider our ENT setup with interacting regions, shown in Fig. 1. It is obvious that flows that are logically separated in the PLC network (don't interact) can be simultaneously active (in fact, 1901 does take care of this). The scenarios of interest to us are smaller constrained areas, where flows interact at the MAC layer. We examine if 1901 effectively handles contention in such cases.

We concurrently establish (i) two (ii) three and (iii) four randomly chosen sets (S) of flows (200 sets in each case) and examine the aggregate throughputs that are achieved. For the concurrently chosen flows, the transmitters and the receivers are distinct. We only establish flows between transceiver pairs that can directly receive signals from each other. The aggregate throughput of each *set* of flows is averaged over 10 runs, each lasting for 30 seconds. All packet sizes are equal to 1480 bytes. Fig. 8 shows the CDF of the aggregate throughput gain of operating the set S of flows concurrently relative to the sum of their individual average through-

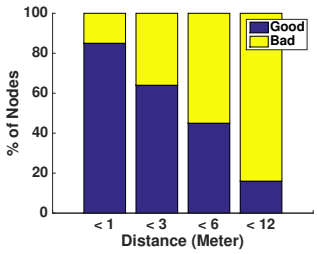

Figure 7: Proximity versus throughput

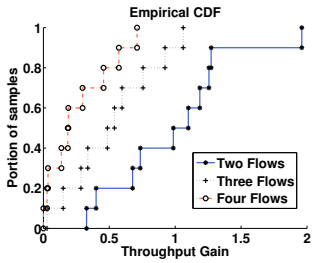

Figure 8: Concurrent invocation of flows.

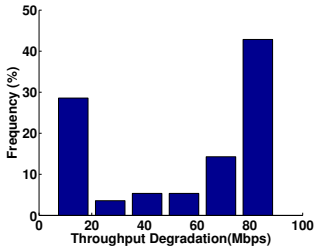

Figure 9: Contention hurts throughput.

puts i.e., $\frac{\tau_S}{(\frac{\sum_{i \in S} \tau_i}{|S|})}$, where τ_i is the throughput of flow i when operated in isolation and τ_S is the aggregate throughput when the flows in set S operate concurrently. We make many interesting observations: (i) Operating three flows provides a marginal throughput gain of about 8% in only 10% of the scenarios. More importantly, invoking three or more flows concurrently almost always leads to a loss in throughput (gain < 1) compared to invoking the flows in isolation. Note that this will be the case with current PLC solutions, where there is no flow regulation at the application/network layer and 1901 arbitrates channel access. (ii) Operating a pair of flows leads to a throughput gain in 60% of the cases, with 40% of the cases yielding over 25% gain and 10% of the cases even yielding over 50% gain. (iii) At the same time, even with pairs of flows, there are several instances (40% of the cases), where operating the flows in isolation is better. These trends are consistent across different PLC networks that we consider (UNI and ENT) and also across different adapters (results omitted due to space constraints).

Achieving spatial reuse while avoiding pitfalls of contention: The higher aggregate throughput from operating *pairs* of flows concurrently indicates that there is room for spatial reuse in PLC contention domains (similar to wireless). However, with an increased number of flows (≥ 3), this opportunity disappears. Note that the 1901 MAC uses a 2-level backoff scheme and larger slot times, and these contribute to inefficiency in its contention process; the impact is exacerbated with an increasing number of flows. This was analyzed in depth by [4]. Another potential cause for the throughput degradation with increasing number of simultaneous flows is hidden terminals; here, two transmitters who cannot hear each other, transmit simultaneously to cause collisions at a receiver. Unlike in a wireless setting (e.g., WLAN) where hidden terminals are often in the proximity of each other, they could be at arbitrary unknown locations in the PLC topology. To illustrate the ill effects of hidden terminals, we form a line topology of four nodes. We choose a fixed flow (from node 1 to 2), and initiate a second flow (from node 3 to 4) concurrently; the second flow may contend with the first flow, and depending on the relative positions of the transceivers may project a hidden terminal. From Fig. 9, we see that in ≈ 42 % of the cases, the throughput degrades significantly (by 80 Mbps) and largely results from hidden terminals (owing to the topology). In summary, opportunities exist for spatial reuse but concurrent flows must be carefully orchestrated to avoid throughput degradations.

4. ZONING THE PLC NETWORK

For ease of management, we subdivide the PLC network into what we call zones. A zone is similar to a contention domain in a wireless network, but as viewed at a higher layer; transceiver pairs with common nodes, that can sustain high throughputs belong to the same zone. If flows in two zones do not interfere with each other, then the zones can simultaneously carry flows with 1901. However, if they interfere with each other, then the arbitration of channel access across the two zones is currently handled by 1901. In the following we consider such *coupled zones* and show that spatial reuse could be possible in such zones as well.

Zones: We define a flow, f_{ij} as a stream of application packets that originates from a sender, i, towards a destination, j. A zone, $\mathcal{Z}$, is defined as a subset of nodes, $n \subseteq N$, such that the average throughput, τ, between its members is $\geq \alpha$. Therefore, a node, $i \in \mathcal{Z}$ iff $\tau_{ij} \geq \alpha, \forall j \in \mathcal{Z}$ $(j \neq i)$. We say that two zones $(\mathcal{Z}_1, \mathcal{Z}_2)$ are *loosely coupled (LC)* if there are no common nodes between them and the throughput between any node in the first zone and any node in the second zone is less than a threshold, η, i.e. $\tau_{ij} \leq \eta$, $\forall i \in \mathcal{Z}_1, j \in \mathcal{Z}_2$ $(0 < \eta << \alpha)$. Loose coupling implies that flows that are fully contained within the two LC zones, do not strongly influence each other and can probably be active simultaneously to provide spatial reuse. In contrast, if $\tau_{ij} > \eta$, we say that zones i and j are strongly coupled (SC) or just coupled.

Zones can capture the throughput influence flows have on each other based on their mutual contention/ interaction on the physical medium. While we discuss how to create and maintain zones (Section 5) and choose α (Section 6) later, we reiterate here that a simple clustering of nodes based on proximity to form zones, is inappropriate as shown earlier.

Classifying flows: Given the zones, the PLC flows can be classified into two categories: local zone (LZF) and cross zone flows (CZF). A flow is local if both the source and sink nodes belong to the same zone, while a flow is considered a CZF, if the source and sink nodes belong to different zones. In Fig. 1 (our ENT testbed) each region (see dashed lines)

corresponds to a zone. The throughputs obtained by various flows within and across zones is presented in Table 2. we see that LZFs tend to exhibit higher throughputs than CZFs. When two flows minimally impact each other, we call them *disjoint flows*. Disjoint flows can be invoked concurrently (reuse) to yield higher aggregate throughput than the average of the individual flows (as seen in Table 2). For example, if f_1 and f_3 are concurrently active they yield an average throughput of 331 Mbps as opposed to 263.5 Mbps when they were sequentially invoked. However, this is not the case for non-disjoint (coupled) flows; for example, concurrent invocation of $\tilde{f}_1$ and $\tilde{f}_3$ (Table 2) yields an average throughput of 18.39 Mbps which is lower than the 27.1 Mbps that is achieved if they are activated sequentially. Next, we point out here that the automated formation of zones and identifying disjoint flows is non-trivial.

5. *BOLT* : SYSTEM DESIGN

In this section, we describe *BOLT* and its component functions. *BOLT* orchestrates an appropriately chosen set of concurrent flows within a single PLC network (at the granularity of time epochs) to enable high throughput. It overcomes the negative effects that arise with 1901 based MAC protocols during periods of high contention (as discussed earlier), by controlling the flows that are simultaneously active. In brief, *BOLT* leverages the concept of zones, and tries to aggressively activate disjoint flows together, while separating strongly coupled flows in time. It is flexible and can incorporate any desired fairness requirement. It is also adaptive to the restructuring of the PLC topology due to electrical devices being turned ON/OFF. It sits at the application layer and is standards agnostic i.e., works with any off-the-shelf PLC adapters; thus, it is readily deployable today.

Challenges: Building *BOLT* in practice has a number of associated challenges. Specifically: **(i)** How do we form zones? Ideally, the zones should include all the nodes that satisfy the requirements defined earlier (throughput between each pair within a zone should be $\geq \alpha$). However, in practice, forming ideal zones may be hard and the overhead for doing so may be prohibitive. **(ii)** How do we determine which flows can be activated simultaneously, and which ones should be isolated from each other ? We need to do this with a small set of training measurements. In addition, the measurements could be noisy due to switching on/off of electrical apparatuses interim. **(iii)** Given the flows that can concurrently be activated, how do we optimally schedule them to maximize spatial reuse ? **(iv)** How do we handle the dynamics of the PLC channel?

5.1 Overview

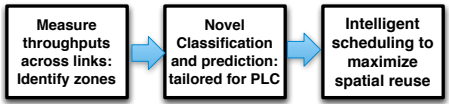

Figure 10: High Level Operational View of *BOLT*

BOLT is built with the above challenges in mind. The quasi-stationary nature of the PLC channel allows *BOLT* to adopt a learning approach to accurately predict the throughput due to interactions between pairs[1] of contending flows with a small set of training measurements. It then efficiently schedules multiple application layer flows in real-time towards achieving very high spatial reuse. *BOLT* is primarily implemented at a central coordinator (CC), which is essentially one of the PLC nodes. *BOLT* 's operations consist of the following (continuously executing) phases (see Fig.10):

(1) Training phase: The CC collects *individual* flow measurements and uses them to construct zones efficiently. This allows the CC to later determine a set of disjoint flows that can be active simultaneously across loosely coupled zones. Subsequently, an additional small set of real-time throughput measurements are performed on *pairs of flows* between nodes chosen from pairs of strongly coupled zones. These measurements are then input to the next phase to determine which flows in SC zones can also be concurrently activated.

(2) Contention inference phase: Using the measurements from the training phase, features that capture flow interactions are intelligently identified. Subsequently an appropriate set of machine learning classifiers are used to accurately (i) classify flow pairs into those that yield a higher aggregate throughput during concurrent operation (termed *reuse-friendly*) and those that do not, and (ii) predict the resulting aggregate throughput for flow pairs that are reuse-friendly.

(3) Scheduling phase: Next, *BOLT* invokes an efficient, yet flexible, scheduling algorithm at the CC to determine the set of application flows that should operate concurrently at the granularity of time epochs. The goal of the scheduler is to maximize reuse in the PLC network, while at the same time, ensuring fairness between flows (across epochs). The schedule is disseminated to the relevant PLC nodes using the same PLC channel or a side-channel.

Handling Dynamics: The addition or removal of new users or electrical appliances or changes in the flow throughput, impact zone construction. This information is fed back to the CC in *BOLT* every epoch (T_e); in response, CC implements scheduling changes. However, the flow configurations (classification and prediction) themselves are updated at coarse time scales of tens of seconds or minutes (T_c); this is sufficient to handle the dynamics in the PLC channel.

5.2 Measurements to train *BOLT*

The first phase of *BOLT* involves performing a small set of measurements. The measurements serve two purposes: (a) identifying zones and (b) capturing flow interactions.

Identifying Zones in the PLC Network: Using the notion of zones reduces overhead by restricting flow interaction measurements (needed for training). It also helps read-

[1]Based on our measurement study, we do not consider combinations beyond flow pairs in SC zones (as they degrade performance).

ily identify opportunities for reuse (disjoint flows). Recall that zones consist of a group of nodes such that the throughputs achieved between any pair within the group is higher than a threshold. This threshold, α, implicitly dictates how large the zones will be. Ideally, given a value of α, the zones should correspond to the largest cliques such that the throughput between the members of each clique is $> \alpha$. This reduces the number of zones formed and the consequent flow interaction measurements. Towards determining the zones, based on the measurements, the central controller forms a graph $G = (V, E)$ with all the nodes in the vertex set V; an edge exists between two nodes if the flow between them yields a throughput $\geq \alpha$. Now, the goal is to find the minimum set of cliques that will cover all nodes; this will ensure that the largest cliques are classified as zones. Unfortunately, this maps to the NP-hard, *vertex clique cover* problem, whose goal is to find the minimum set of cliques that cover all the vertices in a graph. Thus, we leverage a well-known lemma [16] to design a simple, greedy algorithm for zone construction that yields good performance in practice.

LEMMA 5.1. *The chromatic number of a graph is equal to the minimum number of co-cliques (cliques in the complement graph) needed to cover the vertices of the graph.*

The chromatic number refers to the minimum number of colors needed to color a graph. Using the above lemma, we first construct the complement of graph G, namely G'. Then we color the vertices of G'. Now, vertices (nodes) belonging to the same color in G' form a single zone (clique) in the original graph G. Thus, we will have as many zones in G as the number of colors needed to color G'. However, graph coloring is itself a hard problem. Hence, we employ the popular greedy algorithm (Welsh-Powell algorithm) for coloring, which at each iteration, picks the vertex (among un-colored vertices) with the highest degree and assigns it the smallest color (number) that is not used by any of its neighbors. This algorithm is known to use at most $\max_i\{\min\{\delta_i+1, i\}\}$ colors, where δ_i is the degree of node i in the graph (nodes are ordered based on their degree) [17]. Thus, the algorithm uses at most $\Delta + 1$ colors (Δ being the maximum degree in G') and hence constructs at most $\Delta + 1$ zones. Unless the concerned graph exhibits certain structure (e.g., chordal graphs), it is hard to provide good worst case guarantees for coloring and hence clique cover problems. However, we show the effectiveness of our algorithm in practice, in Section 6.

All the nodes measure the link throughputs (every T_C seconds unless significant flow througphut changes are perceived) to their neighbors (the nodes with which they can communicate) and send this information to the CC. Given this set of individual flow measurements, the CC simply executes Algorithm 1 for determining the zones.

Measurement of Flow Interactions: In a very naive case, if one were to consider all combinations of flows to determine whether or not they yield a throughput gain when jointly activated, one would end up with an exponential number of possibilities. By only considering pairs of flows (as guided by our measurement study) we drastically reduce this requirement. In this case, if there are N nodes, $\binom{N}{4} \cdot 3 = O(N^4)$ measurements are needed with a brute force approach; this is because one can select four nodes to establish a pair of flows (3 distinct flow pairs for every 4 nodes). The approach would still incur significant overhead as these measurements might need to be performed every T_c, and N can be large.

Algorithm 1: Zone Construction

Input: $G = (V, N)$

Output: $\vec{\mathcal{Z}}$ set of zones;
Initialization: $G' = Complement(G)$;
$\mathcal{U} = V'$ uncolored vertices. $\mathcal{C} = \phi$ Colored Vertices ;
$C = \phi$ set of colors;
while $\mathcal{U} \neq \phi$ **do**
  $v = HighestValence(\mathcal{U})$;
  Color(v);
  $C \leftarrow$ UpdateColorSet(C,v) ;
  $\mathcal{U} \leftarrow \mathcal{U} - v$;
  $\mathcal{C} \leftarrow \mathcal{C} \cup v$;
for $\forall c \in C$ **do**
  $\mathcal{Z}_c = \phi$;
  for $\forall v \in \mathcal{C}\ c = Color(v)$ **do**
    $\mathcal{Z}_c \leftarrow \mathcal{Z}_c \cup v$;

In *BOLT* , interactions between flows are measured only across *pairs* of proactively formed zones. This results in a total of $\binom{Z}{2} \cdot \binom{\frac{2N}{Z}}{4} \cdot 3$, which in turn translates to $O(\frac{N^4}{Z^2})$ measurements; here, Z is the number of zones and nodes are assumed to be uniformly distributed across the zones for simplicity, i.e., $\frac{N}{Z}$ per zone). The reduction comes from restricting flow pair measurements to only *pairs* of zones (flow pairs, whose four end points are such that they belong to more than two, i.e., 3 or 4 zones, are not measured). Further, even for each pair of zones, not all flow pairs (involving both local and cross flows) in the two zones are measured. Only a fraction x of the net flow pairs, i.e. $\binom{\frac{2N}{Z}}{4} \cdot 3x$ in the two zones are picked randomly and measured; this is then used to construct the flow interaction models (explained in Section 5.3). The latter is in turn used to predict interactions between other flow pairs (spanning more than 2 zones) that were not measured. Our experiments reveal that the appropriate construction of zones (with larger size), allows x to be as low as 0.1 (10%), while still yielding good prediction accuracy.

5.3 Flow Contention Inference

At the end of the measurement phase, the CC can determine the LC and SC zones. Conceivably, links in the LC zones can be concurrently activated to increase spatial reuse. The challenge is to determine which links can be concurrently activated within and across SC zones to further increase spatial reuse. We discuss our approach for addressing this challenge in this section (our approach identifies spatial reuse opportunities across both LC and SC zones).

The quasi-stationary property of the PLC channel motivates the use of machine learning (ML) to learn and predict the impact of flow interactions. Our approach consists of a classification and, a prediction stage. The classification stage determines whether the aggregate throughput from concurrently running a pair of flows exceeds the average throughput over running the flows in isolation. The prediction stage estimates the aggregate throughput of a pair of flows activated concurrently, using training data. The two stages are based on classifier and prediction models used in ML; *BOLT* includes a feedback mechanism that recalibrates these models every T_c to adapt to channel variations.

Classification: Towards identifying flows that can be concurrently activated, we first identify a set of features that capture interactions between the transceiver pairs of the flows. Then, we intelligently use a set of classification models that take these features as inputs and determine if the flow pairs are amenable to concurrent activation.

What features to use?: Whether or not two transceiver pairs (flows) can simultaneously communicate depends on the interactions between them (carrier sensing, interference, noise etc. affect these interactions). We consider interactions (i.e., throughput) between all possible combinations (i.e., every pair from the four nodes making up the two flows) as the basis feature set. Next, we reduce this basis feature set towards making the process faster and more accurate. This reduction is based on well-known ML techniques (dimensionality reduction). We consistently see that two of these features get eliminated in our experiments. Specifically, this reduction is a direct artifact of zoning; since the flow pairs we consider are between pairs of zones, the interactions between a transmitter and an unintended receiver can be derived based on the interactions between the two transmitters, the two receivers, and between the legitimate transmitter-receiver pairs. We do not elaborate on this here due to space constraints, but one can easily construct toy examples to see why this is the case.

The classification algorithm takes the reduced set of feature values and maps them onto two clusters (one for concurrent, one for isolated). Different classification algorithms have their pros and cons (based on the ways in which the clusters are formed). Although the PLC environment is relatively stable, there could be noise in the training set from electrical devices being turned ON/OFF. Thus, the classifier needs to be noise resistant. As a design goal, we seek to make sure that the training data for classification is not large (and yet achieve good accuracy). Finally, we want the approaches to be simple and efficient (fast).

Given the above objectives, we intelligently combine three classifiers to achieve very high accuracy. Our approach, shown in Fig. 11, first combines the outputs of two well known, simple classification techniques viz., Classification Trees (CT) and TreeBagger (TB).If the outputs of CT and TB differ (i.e., one suggests concurrent operations while the other contradicts), we choose a third approach namely, the nearest neighbor classifier (NN) to resolve the conflict. A description of these classification techniques can be found in [18].

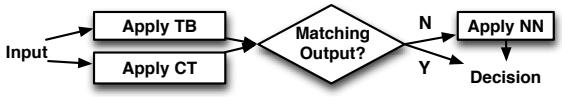

Figure 11: Classification Process Overview

The rationale for our approach is as follows. CT does not require a large training set but is sensitive to noise. The quasi-stationarity ensures low noise most of the time; however, plugging in devices causes noise which may cause CT to underperform. Thus, while CT is generally usable, it needs to be supplemented to deal with the noisy cases. TB uses innovative ordering to reduce impact of noise; however, the ordering inherently introduces a sampling bias (meaning some clusters may have a better chance of being picked). To reduce bias a larger training set may be needed. Thus, while it eliminates the issue with CT, it brings up a new problem. Thus we combine CT and TB and if they yield the same output decision, we can be relatively confident that the decision is the right one. If however, their outputs contradict, we go with a third classier for resolution. Here we choose a very popular generic classifier, viz., NN. NN offers lower accuracy than CT and TB (hence it was not chosen in the beginning) with small amounts of training data, but is not sensitive to noise and does not introduce sampling biases. We could have used multiple classifiers and used a majority vote at this stage, but prefer the simpler approach of just using NN since it offers high enough accuracy with low complexity.

Prediction of aggregate throughput : Finally, *BOLT* seeks to estimate the potential aggregate throughput for a given pair of flows that are activated concurrently (along with the individual contribution of the flows in the pair); this prediction is later used when scheduling flows. We use boosting tree (BT) [19] as a regression approach for prediction. Our choice is motivated by the fact that BT is an ensemble method that fits complex non-linear variables (i.e., features) to predict outcomes. In addition to being fast, BT is resistant to missing data (low amount of training data) and eliminates outliers (due to noise). BT achieves a high prediction accuracy by adaptively combining multiple binary trees [19].

5.4 Flow Scheduling

At this stage, the CC knows which flows are reuse-friendly and the throughput gains from concurrent invocation of such flows. The scheduler then chooses the active flows at each epoch to maximize the aggregate throughput while meeting desired fairness constraints. Such scheduling problems are typically cast as utility maximization problems, as in

$$\text{Maximize} \sum_{f \in \mathcal{F}} U(\bar{\tau}_f) \tag{1}$$

where, $\bar{\tau}_f$ is the average throughput received by flow f in T_c, and $\mathcal{F}$ represents the set of flows. The choice of the utility function, $U()$, determines the fairness policy. We impose

proportional fairness ($U(\bar{\tau}_f) = \beta_f \log(\bar{\tau}_f)$) [20], but ***BOLT*** can support other fairness policies governed by concave utility functions (e.g., max-min, min potential delay); β_f is the priority weight for a flow (e.g., latency-sensitive flows could have higher priority). The approach accounts for flow diversity and allocates resources to provide average throughputs ($\bar{\tau}_f$ over long-term, T_c) proportional to the flows' priorities and transmission rates. Solving the above problem ensures that the aggregate throughput of flows is maximized in a proportionally fair manner over time scales of T_c.

Per-epoch Scheduling: It has been shown that finding the optimum solution to the above problem is equivalent to solving the following per-epoch (T_e) scheduling problem viz., maximizing the aggregate *marginal* utility in *every epoch*, i.e., Maximize $\sum_{f \in \mathcal{F}} \Delta U(\bar{\tau}_f)$ [20, 21]. ΔU_f is the marginal utility received by flow f in the epoch and is given by $\Delta U_f = \frac{dU_f}{dT_e} = \frac{dU_f}{d\bar{\tau}_f} \cdot \frac{d\bar{\tau}_f}{dT_e} = \frac{\beta_f \tau_f}{\bar{\tau}_f}$ for proportional fairness [21], where τ_f is the predicted throughput (transmission rate) for flow f in the current epoch and $\bar{\tau}_f$ is the *long term* average throughput received by the flow f so far. At the end of every epoch t, the average throughput received by a flow f is updated as,

$$\bar{\tau}_f(t) \leftarrow (1 - \frac{1}{T_e})\bar{\tau}_f(t-1) + (\frac{1}{T_e})\tau_f(t) \tag{2}$$

where $\tau_f(t) = 0$ when flow f is not scheduled. Updating the average throughput through an exponentially weighted moving average allows for fast adaptation to network dynamics. By tracking the throughput received by a flow thus far with $\bar{\tau}_f$, the scheduler weights the flow (as $\frac{1}{\bar{\tau}_f}$) accordingly in the next epoch to ensure fairness at time scales of T_c.

We now focus on the per-epoch scheduling problem, i.e.

$$\text{Maximize}_S \sum_{f \in S} \frac{\beta_f \tau_{f,S}}{\bar{\tau}_f} \tag{3}$$

Here, β_f and $\bar{\tau}_f$ essentially serve as constant weights for the flow in the current epoch. Hence, the optimization is w.r.t. to the flows chosen for schedule (S) in the current epoch; here, the throughput of a flow ($\tau_{f,S}$) chosen in the schedule depends on other flows in the set S, scheduled concurrently.

Algorithm: We want to select flows and/or flow pairs for each epoch; the selection must ensure that the chosen flows do not negatively impact each other. Our scheduling problem can be cast as a maximum weight independent set (MWIS) problem as follows. We represent each individual flow (f), as well as each of the reuse-friendly flow pairs (f, g) as a separate vertex on a graph G, with the weight (w_i) of the vertex (i) being the weighted throughput achieved by the corresponding flow ($w_i = \frac{\beta_f \tau_f}{\bar{\tau}_f}$) or flow pair ($w_i = \frac{\beta_f \tau_{f,fg}}{\bar{\tau}_f} + \frac{\beta_g \tau_{g,fg}}{\bar{\tau}_g}$). Here, $\tau_{f,fg}$ ($\tau_{g,fg}$) is the throughput of flow f (g) when flows f and g operate concurrently. Any two vertices whose mutual flows are not disjoint have an edge between them in G. If two vertices represent a flow pair each, then each of the flows in one pair must be disjoint w.r.t. each of the flows in the other pair to avoid an edge in G. Now, the scheduler seeks to find the subset of flows and flow pairs that are disjoint, for which the aggregate weighted throughput is the maximum. This essentially reduces to finding an independent set of vertices (disjoint flows/flow-pairs) on G with maximum weight and automatically yields the optimum solution to our per-epoch scheduling problem.

However, finding even a maximum independent set or MIS (without weights or equivalently unit weights) on general graphs is NP-hard. Hence, we use a greedy algorithm (inspired by those for finding the MIS [16]) that yields efficient performance in practice. At each iteration, the vertex with the smallest degree is chosen and added to the independent set and its edges and neighboring vertices are removed. With vertices carrying weights in our case, we suitably adapt the algorithm to pick the vertex (i^*) that yields the smallest loss in weighted throughput in each iteration,

$$i^* = \arg\min_i \gamma_i, \text{ where } \gamma_i = \sum_{j \in N(i)} w_j - w_i \tag{4}$$

Note that, when we pick a vertex (flow or flow pair), we eliminate its neighbors from the schedule and hence their weight contributions. Thus, at each iteration, the algorithm strives to pick the vertex that not only contributes maximum weight by its addition to the schedule but also minimum loss due to removal of its neighbors. In the end, we are left with a set of flows and flow pairs that are mutually disjoint and can be scheduled concurrently in the current epoch to maximize the aggregate weighted throughput. At the end of the epoch, the average throughputs ($\bar{\tau}_f$) of all the flows are updated based on Equation 2, which in turn affects their corresponding weights $\frac{\beta_f}{\bar{\tau}_f}$ and hence controls the relative flow priorities (track fairness) for scheduling in the next epoch.

Remarks: When the flow weights are the same, the problem and our algorithm reduce to that of a MIS and yield a worst case approximation guarantee of $\frac{\Delta+2}{3}$ [22].

5.5 Discussion:

Our focus in this work is on link-level single-hop flows. The end-points of a desired flow may not be directly reachable from each other and multiple hops may be needed to establish connectivity. With ***BOLT*** , we maximize the link throughput considering each hop independently. Maximization of multi-hop path level throughput is left for future work.

If all nodes are unable to reach a single PLC coordinator, the network may be sub-divided into multiple PLC sub-networks and a coordinator chosen for each. Then, *BOLT* will operate (at the coordinator) within each smaller PLC sub-network independently to allow for scalability. We believe that in large PLC networks, with such a sub-division, a large fraction of nodes in one PLC sub-network possibly can be made disjoint from those in the other sub-networks (different parts of an enterprise building). We expect the contention between nodes at the edge of these sub-networks to be small and handled by the 1901 MAC. A careful assessment of how to form such sub-networks is left to the future.

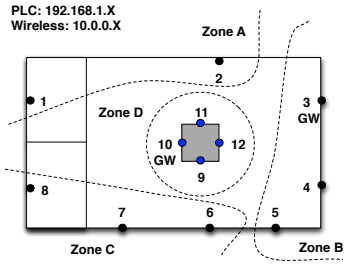

Figure 12: UNI Testbed.

Training Size	10%	20%	30%	40%	50%
BC	87(3)	89(2)	88(4)	93(3)	95(2)
TB	86(5)	90(2)	91(3)	94(3)	94(1)
NV	76(14)	74(11)	82(4)	79(7)	80(5)
NN	84(4)	87(2)	89(2)	93(3)	95(2)
SV	83(4)	85(2)	86(3)	87(3)	92(3)
CT	85(5)	87(4)	86(4)	93(3)	94(3)

Table 3: Classification accuracy (%): mean (std dev); ENT dataset

Technique	UNI	Time	ENT	Time
LL	13	7	12	7
DT	14	6	11	6
BG	16	6	14	6
BT	11	5	10	4
NE	13	20	12	19
LS	17	6	15	6
RV	15	240	11	238
GR	15	10	12	10
VH	14	15	12	14

Table 4: Prediction error (%) and training time (msec)

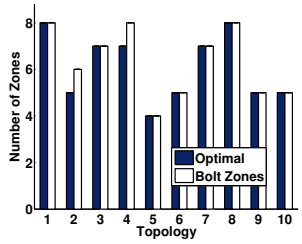

Figure 13: Sub-optimality of *BOLT* 's zone construction.

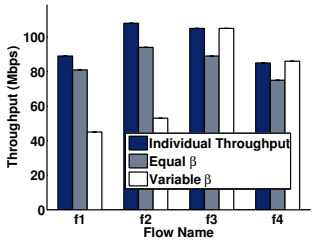

Figure 14: Evaluating fairness with *BOLT* .

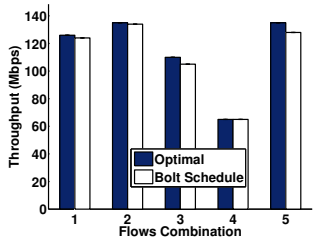

Figure 15: Sub-optimality of *BOLT* 's scheduler

6. EXPERIMENTAL EVALUATIONS

We implement *BOLT* and deploy it on three testbeds (an enterprise, a university and a residence). We evaluate each of *BOLT* 's components as well as its holistic performance.

6.1 Implementation and setup

Testbeds: Our first testbed is in an enterprise and has eight machines (nodes) running Ubuntu 9.4 with two interfaces (a WiFi interface and a PLC interface) and spanning offices, labs, cafeterias and conference rooms. We experimented with different topologies but the one shown in Fig. 1 was the default topology. UDP flows with payload sizes of 1480 bytes, were established between node pairs for 30 seconds by default. The second testbed (shown in Fig. 12) is in a large lab setting at a university and consists of 16 machines running Ubuntu 14.4. The third testbed consists of 6 nodes, that are deployed at a residence; two of these run OS X and four run Windows 8. The data collection and scheduling phases of *BOLT* are implemented in C++. The classification and prediction models are implemented with Matlab. We implement the approach described in [23] to estimate the average flow throughput (in isolation) via a probing phase. Note here that we operate the experimental networks at relatively high loads since these are the regimes where the management of multiple flows becomes important.

Experimental Setup: Each network has an assigned central controller (CC) that performs network coordination and scheduling. The PLC channel itself is used as a control channel to collect measurements and disseminate schedules. The overhead for control traffic is extremely small and does not interfere with ongoing communications; these packets essentially only contain the identifiers of the transceivers, file sizes in the case of requests, and time epochs assigned in the case of schedules and are 12 bytes long.

We consider one-hop PLC flows generated using Pareto and Uniform random variables; they are set-up between two randomly picked nodes in the test-bed. Later we consider a gateway model, where all PLC flows converge towards a common gateway(s) to access the Internet. *BOLT* schedules flows at the granularity of epochs of 100 ms. Unless otherwise specified, each experiment lasts for 5-10 minutes and results are averaged over twenty runs. We consider three levels of traffic load viz., high, medium and low that correspond to averages of (3,10, >10) flows per time epoch (100 msec). The average throughput per flow per epoch is our main metric of evaluation. To save space, we mainly present results from the UNI testbed (unless explicitly mentioned); experiments in other settings yielded similar behavioral results.

6.2 Benchmarking *BOLT* 's Components

Zone Construction: In this experiment, we compare the number of zones formed by *BOLT* 's zone construction algorithm with the *optimal* number of zones in the PLC topology. Ideally, node pairs between which the link quality is good must belong to the same zone. Determining a threshold for classifying a link as a "good quality" link is hard. We find via several experiments that if a link is able to achieve 75% of the maximum (isolated) link throughputs possible, it can be classified as a good quality link. Thus, in our UNI network, we assume a throughput threshold of $\alpha = 80$ Mbps for determining the zones (maximum link throughputs are ≈ 100 Mbps). Using a similar approach we find that nodes in different loosely coupled (LC) zones can establish flows that can only sustain $< 1\%$ of the maximum throughput possible; correspondingly we set η to be 1 Mbps on the UNI network. The optimal number of zones is found via an exhaustive search across combinations of nodes (possible due to the moderate-scale topology) and picking the combination

with the least number of zones. We consider ten different PLC topologies. As seen in Fig. 13, in 8 out of 10 of the topologies, *BOLT* constructs the optimal number of zones. Only in two of the topologies, *BOLT* employs one additional zone. Although our current testbed is relatively small, recall that zones are similar to contention domains in wireless; thus, small regions of interacting zones are of interest. Due to the inherently constrained nature of such interactions, we believe the efficiency of our zone construction will also scale to larger topologies.

Picking α: While a smaller value of α reduces the number of zones (more nodes per zone) and thus decreases the training measurements needed, it negatively affects the accuracy of predicting flow interactions. Our choice of α strikes a balance between accuracy and reducing training measurements.

Classification and Prediction: Since the accuracy of the classification and prediction models directly impact *BOLT* 's performance, we now compare our classifier/prediction approach against state-of-the-art techniques. First, we compare *BOLT* 's classifier (*BC*) against five popular classifiers (described in [18]) viz., TreeBagger (TB) , Naive Bayes Classifier (NV) , Nearest Neighbors (NN) , Support Vector Machines (SV)and Classification trees (CT). Table 3 depicts this comparison w.r.t different training set sizes for the ENT dataset (results from the UNI dataset were similar and are thus omitted due to space constraints). By intelligently combining results of CT, TB and, NN in case of ties, *BC* achieves higher average accuracy than any of the five individual classifiers in more than 90% of the training scenarios (CDF over training scenarios not presented here). Further, with just 10-20% of the training data, *BC* is able to provide about 90% accuracy, which is highly promising. The reasons for BC's superior performance are as follows. First, by combining the results of TB, CT and NN, *BC* outperforms each of these individual classifiers. For SV to be efficient, its parameters need to be calibrated for each specific topology; using SV without such customized calibration can cause high false positives and negatives. In NV, the posterior probability estimate is negatively affected if there is only a small occurrence of a certain class (e.g., an isolated node in a zone).

To evaluate the accuracy of our prediction model using Boosting trees (BT), we compare it with eight other different prediction models (described in [19]): Least squares Linear regression (LL) , Decision trees (DT) , Bagging trees (BG) , Neural networks (NE) , Least Squares Support Vector Machines (LS), Relevance Vector machine (RV) Gaussian Process Regression (GR) and, Variational Heteroscedastic Gaussian Process Regression (VH). In Table 4, we show the root mean square error (RMSE) rate (%) and training time (ms) for each technique for the two datasets. The RMSE captures the accuracy of prediction, while the training time captures the computational intensity (i.e., the time required to construct the prediction model). We see that in addition to its high accuracy, BT's running time is the fastest.

Scheduling: To evaluate the efficiency of our scheduler, we create two sets of six flows each. *First*, we set the priority coefficient of both flow sets to one ($\beta_f = 1$, *Scenario 1*). *Second*, we increase the coefficient for the latter set to two, while keeping that of the first set to one (*Scenario 2*). We plot the throughput of two of the flows from each set in Fig. 14 for the two scenarios. In *Scenario 1*, *BOLT* 's scheduler loses about 5-10 Mbps in throughput for the individual flows compared to their throughputs in isolation (total network throughput is higher). This shows that the throughput loss due to sharing of the PLC medium is minimal, indicating that the scheduler picks the right set of flows to operate concurrently (so as to not degrade their individual throughputs appreciably). Further, the flows receive their average throughput proportional to their individual epoch throughputs (proportional fairness is ensured). In *Scenario 2*, the higher priority flows have a two-fold increase in their relative throughputs, at the expense of the flows in the first set. This means that in a given duration, the first two flows are able to transfer much more data than the second two flows (which have lower priority and thus get scheduled less often). This indicates *BOLT* 's ability to differentiate between flows of different types, thereby allowing it to prioritize and provide lower latency for real-time flows.

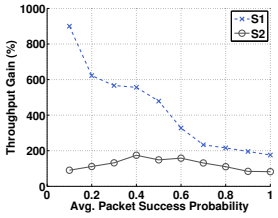

Figure 16: Throughput gain over baselines.

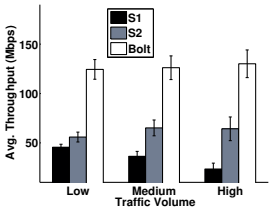

Figure 17: Impact of traffic load.

Scheduler sub-optimality: We now quantify the sub-optimality of *BOLT* 's per-epoch schedule computation (recall that finding the optimal schedule corresponds to solving the NP-hard MWIS problem). We consider a topology with a subset of six nodes and establish four flows (arbitrarily chosen) between these nodes. We choose five such combinations, C_k, $k \in \{1, 5\}$. We also exhaustively generate all possible schedules for each combination, and pick the optimum i.e., the one that yields the highest utility (weighted throughput). We compare this utility against that achieved with *BOLT* 's schedule (priority weights are set to 1). Fig. 15 compares the utilities for the five combinations in the two cases. We observe that *BOLT* 's (much simpler) scheduler yields utilities that are at most 5% lower than the optimal.

6.3 Holistic evaluation of *BOLT*

Impact of Scheduling: We consider two baselines: one where all given application layer flows are concurrent (state-of-the-art PLC operation, **S1**), and another, where such flows are queued and only one flow is active in any given epoch (**S2**). Since no classification/prediction models are required

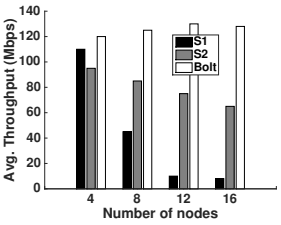

Figure 18: Impact of topology size.

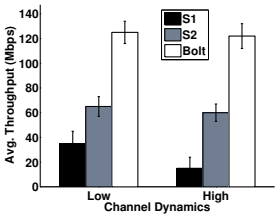

Figure 19: Effect of channel dynamics.

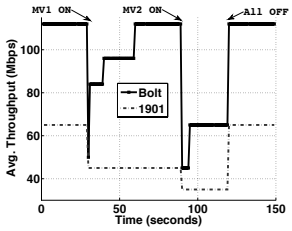

Figure 20: resiliency to channel dynamics.

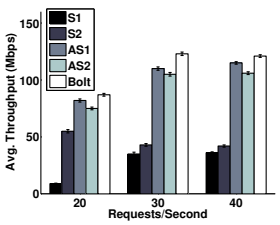

Figure 21: Holistic evaluation

for either of these baselines, they can be considered as alternate potential systems for PLC. In both cases, the MAC protocol based on 1901 in the adapters arbitrates the channel. In the first experiment, we vary the average packet delivery ratio (PDR) and measure the throughput gain of *BOLT* over the two baselines. PDR is a function of the channel dynamics (eg. electrical apparatuses plugged on/off) that needs to be controlled to allow for comparison; hence, we control PDR by manually dropping packets with a certain probability at the receiver. We keep the average traffic load to be three flows per epoch. As shown in Fig 16, we observe that the throughput gains with *BOLT* as compared to S1, can be as high as 8.5x. The gain is higher in bad channel conditions, where running all incoming flows together (in S1) leads to aggravated collisions/back-offs. By scheduling the right sets of flows together, *BOLT* alleviates this effect and achieves much higher throughputs. With S2, running individual flows sequentially avoids the collision/back-off penalty; however, it also misses out on reuse opportunities that lead to higher total throughput. The throughput gains with *BOLT* over S2 are also significant (100-200%) and remain almost constant across different channel conditions. Here, *BOLT* 's gains are mainly from its ability to leverage spatial reuse.

Impact of Traffic Load: Next, we study the impact of traffic load in Fig. 17. The average throughput of S1 decreases as traffic load increases; more flows increase contention (backoffs) and collisions from hidden terminals. The average throughput of S2 remains almost constant over different traffic loads since it schedules a single flow in every epoch and hence does not depend on the number of flows (but only their individual throughputs). Notably, *BOLT* 's throughput is at least 2.5x that of S1 and 1.5x that of S2.

Impact of variations in the topology size: Next, we examine how the performance of *BOLT* changes as the topology size is varied; the topologies we consider contain SC zones. For each topology (of a given size), we divide the nodes into two equal size sets; the first are the sources and the second set, the destinations. Thus, the load increases linearly with the topology size. For each topological size, we choose 20 randomly chosen configurations (as described above), and run experiments for 30 seconds each. From Fig. 18, we see that *BOLT* maintains a steady throughput as the topology scales. There is in fact a slight improvement in throughput due to better exploitation of spatial reuse from the increased number of flows. S1 suffers from collisions/backoffs as the topology (and thus, the corresponding load) is scaled. The throughputs with S2 remain fairly stable; however, because of a slightly higher number of poorer quality links, the throughput takes a slight dip as the topology size (and thus, the number of flows) is increased.

Impact of Channel Dynamics: Our next experiments (Fig. 19) are done during different times of day and capture the impact of channel dynamics on *BOLT* 's performance. The rate of switching (on/off) of electric apparatuses (which induces the dynamics) during peak times (arrival of people to work, lunch time, etc.) is high and at other times (early morning, at night, etc.) is low. The average throughput of *BOLT* does not vary much in either case; it schedules different subsets of flows across different epochs and thus copes with the dynamics. Similarly, S2 has a relatively steady performance as it picks different flows across epochs.

To further illustrate *BOLT* 's ability to cope with dynamics, we conduct a controlled experiment on our residential testbed. Here, we turn on two microwave ovens in an operational PLC network at different times. At each instance, as shown in Fig. 20, the throughput takes a hit. However, *BOLT* quickly isolates the effect of the poor quality links and the network is reconfigured to restore the throughput to almost the same levels prior to the oven(s) being turned on. On the other hand, without *BOLT* , the poor quality links further degrade the throughputs that were achieved with just 1901 (which were lower than with *BOLT* to begin with); the more the noise the bigger the hit with 1901. This experiment exemplifies *BOLT* 's ability to quickly cope with channel dynamics to restore throughput. Fairness is compromised a little in the short term (results not shown due to space constraints), but is restored long term as the microwaves go off.

Impact of Classification/Prediction Models: Finally, to study the impact of the models' choices on *BOLT* 's performance, we consider two alternate systems, where we retain the scheduler in *BOLT* , but change its classification/prediction models: *(a) Alternative System 1 (AS1):* The classification model is Nearest Neighbor (NN) and the prediction model is Gaussian Process Regression (GR). *(b) Alternative System 2 (AS2):* The classification model is Naive based (NV) and the prediction model is Neural networks (NE). We also consider a different scenario where all nodes communicate with two common gateway nodes connected to the Internet.

We stream multiple HTTP pages (CNN, Facebook, Twitter, Google, Yahoo) through these two gateway nodes. The average size of the HTTP web pages are 4.4, 1.3, 2.7, 5.4 and 2.4 MB, respectively for these sites. We generate requests for web pages according a Pareto random variable, and assign the requests to each node randomly. The results are averaged over 30 runs, each lasting for 1 minute. In Fig. 21, we compare the average throughput of *BOLT* with that of AS1 and AS2. We see that the bulk of *BOLT* 's gain comes from its scheduling component. However, its prediction/classification models still offer gains over AS1 and AS2 (15-30%); this shows that the selection of the right classification/learning models is useful. Specifically, the higher accuracy from these models contribute to better scheduling decisions and hence, system throughput. S1 and S2 underperform *BOLT* as well as AS1 and AS2 because of their sub-optimal scheduling decisions (as seen in prior experiments).

7. CONCLUSIONS

In this paper, we design and implement a standards agnostic framework *BOLT* , to realize the throughput potential of PLC, for it to serve as a viable backhaul for local network connectivity. *BOLT* aggressively reuses the spectrum while avoiding collisions and backoffs to drive the network throughput to near-optimal levels, while enforcing desired fairness requirements. It is lightweight and uses only a small set of online training measurements. Real-world experiments showcase *BOLT* 's ability to improve system performance significantly over state-of-the-art PLC solutions.

Acknowledgments: This work was partially supported by the NSF NeTS grant 1528095, and was initiated when the first author was at an internship at NEC labs.

8. REFERENCES

[1] M. Obitko, V. Jirkovsky, and J. Bezdicek. Big data challenges in industrial automation. In *Industrial Applications of Holonic and Multi-Agent Systems*, 2013.

[2] T. B Murdoch and A. S Detsky. The inevitable application of big data to health care. *JAMA*, 309(13):1351–1352, 2013.

[3] HomePlug Support for IEEE Standards. http://bit.ly/1AThvHO.

[4] C. Vlachou, A. Banchs, J. Herzen, and P. Thiran. Analyzing and Boosting the Performance of Power-Line Communication Networks. In *ACM CoNEXT*, 2014.

[5] C. Vlachou, A. Banchs, J. Herzen, and P. Thiran. On the MAC for Power-Line Communications: Modeling Assumptions and Performance Tradeoffs. In *IEEE ICNP*, 2014.

[6] C. Vlachou, J. Herzen, and P. Thiran. Fairness of MAC protocols: IEEE 1901 vs. 802.11. In *IEEE International Symposium on Power Line Communications and Its Applications*, 2013.

[7] M. Gotz, M. Rapp, and K. Dostert. Powerline channel characteristics and their effect on communication system design. *Communications Magazine, IEEE*, 42(4):78–86, Apr 2004.

[8] M. Zimmermann and K. Dostert. A multipath model for the powerline channel. *Communications, IEEE Transactions on*, 50(4):553–559, Apr 2002.

[9] M. Korki, N. Hosseinzadeh, H.L. Vu, T. Moazzeni, and Chuan Heng Foh. A channel model for powerline communication in the smart grid. In *IEEE Power Systems Conference and Exposition*, 2011.

[10] C. Vlachou, A. Banchs, J. Herzen, and P. Thiran. Performance analysis of MAC for power-line communications. ACM SIGMETRICS, 2014.

[11] G. Horvat, Z. Balkić, and D. Žagar. Power line communication throughput analysis for use in last mile rural broadband. In *20th Telecommunications Forum TELFOR 2012*, 2012.

[12] R. Murty, J. Padhye, R. Chandra, A. R. Chowdhury, and M. Welsh. Characterizing the end-to-end performance of indoor powerline networks. *Harvard University Microsoft Research*, 2008.

[13] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Measurement-based models of delivery and interference in static wireless networks. In *ACM SIGCOMM*, 2006.

[14] S. Sancha, F.J. Canete, L. Diez, and J.T. Entrambasaguas. A channel simulator for indoor power-line communications. In *ISPLC.*, 2007.

[15] TCP/UDP Bandwidth Measurement Tool. http://iperf.fr/.

[16] D. B West. *Introduction to Graph Theory*. Prentice-Hall, 2000.

[17] D. J. Welsh and M. B. Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. In *The Computer Journal*, 1967.

[18] I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Inc., 2005.

[19] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[20] F Kelly, A. Maulloo, and D. Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research society*, 1998.

[21] A Sang, X. Wang, M. Madihian, and R.D. Gitlin. A flexible downlink scheduling scheme in cellular packet data systems. *Wireless Communications, IEEE Transactions on*, 5(3), March 2006.

[22] M. Halldorsson and J. Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. In *Algorithmica*, 1997.

[23] R. Liyong, C. Bo, and W. Jing. A novel packet-pair-based inferring bandwidth congestion control mechanism for layered multicast. *SIGOPS*, 37(4), Oct 2003.