WASP: A Software-Defined Communication Layer for Hybrid Wireless Networks

Murad Kaplan, Chenyu Zheng, Matthew Monaco, Eric Keller, and Douglas Sicker
Dept. of Computer Science, University of Colorado
Boulder, Colorado, USA
murad.kaplan@colorado.edu, chenyu.zheng@colorado.edu,
matthew.monaco@colorado.edu, eric.keller@colorado.edu,
douglas.sicker@colorado.edu

ABSTRACT

In this paper we introduce WASP, a general communication layer for hybrid wireless networks where multiple networks are used to complement each other. In our system, we capitalize on an infrastructure with a ubiquitous, wide-area network to help enable the creation of a local mobile ad-hoc network in an efficient, scalable, evolvable, and manageable way. In particular, in an architecture inspired by software-defined networking, we decouple the control plane and data plane in the mobile devices and shift the control plane to a centralized controller. The controller, reachable via the wide-area network, manages a collection of mobile devices by informing each device how to handle traffic based on neighbor information provided by the mobile devices. With this, a mobile ad-hoc network can help reduce the data burden on the ubiquitous network, and the ubiquitous network can help reduce the burden on the mobile devices. WASP can be used in different networks with different applications such as cellular and military networks. In this paper, we based our implementation on Android and tested on a collection of Google Nexus-4 devices to measure metrics such as battery consumption. We evaluate on an extended ns-3 simulation platform which we added the ability to run unmodified Android applications on the nodes within ns-3. Our experiments show that WASP scales better than traditional ad-hoc networks with only a minimal trade off of energy. Additionally, we show that a content distribution scheme using WASP on smartphones with cellular data plans significantly offloads bandwidth from the cellular infrastructure, and in turn reduces expensive data usage and energy usage.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: subject—*centralized networks, wireless networking*

ANCS'14, October 20–21, 2014, Los Angeles, CA, USA.

http://dx.doi.org/10.1145/2658260.2658263.

Keywords

ad hoc networking; wireless networking; software-defined networking

1. INTRODUCTION

Mobile ad-hoc networks (MANETs) are traditionally designed for environments with no pre-existing infrastructure. The need to self-organize leads to the need to exchange and process a high volume of control messages to manage its communication network and recover from changes in the network, such as through mobility. This leads to overwhelming the wireless channels which, in turn, causes packet loss. It also puts an extra overhead on mobile devices to process and forward these control packets. With these challenges, MANETs become unscalable and suffer high packet loss and error rates as the number of nodes increases.

In this paper, we explore a scenario where ubiquitous network connection exists but is expensive to use. We note that many of today's mobile devices routinely equipped with multiple interfaces. Examples of such hybrid networks can be seen in cellular infrastructures where smartphones have multiple interfaces (*e.g.,* Wi-Fi, Bluetooth, 3G/4G), and military setting where satellite or a unmanned aerial vehicle(UAV) provides ubiquitous network connection to soldiers' mobile devices that also are equipped with UHF/VHF radios. We take advantage of the existence of a long-range, ubiquitous wireless interface to offload network control and management from the wireless nodes to a centralized, general purpose server.

With this, a mobile ad-hoc network can help reduce the burden on the ubiquitous network (*e.g.,* by reducing the data downloaded through it), and the ubiquitous network can help enable the creation of a local mobile ad-hoc network (*e.g.,* by removing the burden of running complex routing protocols from the mobile devices).

We present the design, implementation, and evaluation of WASP [1], a communication layer within a mobile device for hybrid networks. In contrast to mobile ad-hoc networks, with WASP, the devices themselves do not run any routing protocols. Instead, we are guided by software-defined networking (SDN) [24, 18]. By capitalizing on a 'one-hop' connection to a centralized controller, we transfer the duty of network management calculations from battery constrained

[1]WASP is named for the technologies our prototype encompasses (Wi-Fi, Ad-Hoc, Software-Defined Networking, and Personal-Mobile)

devices to a powerful server (or elastic collection of servers). When topology changes occur, we do not need to flood the network with information and wait for convergence as in a distributed protocol (such as OLSR [12]). At the same time, we do not have to revert to such approaches as being exclusively reactive (as in AODV [34]) where, to be more scalable, paths are calculated when two nodes attempt to communicate. Importantly, as we will show, this shift does not introduce a lot of control overhead on the expensive wide-area radio interface – our solution uses an on-demand network control protocol which is very efficient.

It is important to highlight that WASP is a general purpose communication layer for hybrid networks, not a special purpose architecture. In this paper, we evaluate with a hybrid cellular/Wi-Fi network, but can envision this easily extending to a military setting where satellite or a unmanned aerial vehicle (UAV) provides the ubiquitous, expensive coverage and another radio technology provides the local connectivity. Further, we are not limited in scope of application. To demonstrate this, layered on top of the communication layer, we provides services for aiding applications with different content distribution requirements. For example, the *web content service* uses peer-to-peer content serving of cached content similar to Firecoral [38], but as a general service not tied to the browser. In this case, the controller also acts as a look up mechanism to track which mobile devices are caching content, make suggestions as to which device to request the content from, and enable direct communication to fetch that content.

In this paper, we make the following contributions:

- *Design* of the WASP communication layer which provides a lightweight control protocol between a central server and mobile devices, efficiently forwards traffic, and reacts quickly to failure or mobility.
- *Implementation* of WASP and associated layered services on Android, and tested in a small collection of five Google Nexus S phones.
- *Evaluation* of WASP where we demonstrate that (i) the amount of control traffic per node remains relatively small and constant with an increasing number of nodes in the network, whereas ad-hoc protocols grow exponentially and quickly become unusable, (ii) in streaming data among 100 nodes, WASP delivers up to 95% of the packets while OLSR and AODV barely deliver 10%, and (iii) a content distribution scheme using WASP significantly reduces load on the cellular infrastructure.
- *Extension to ns-3* which provides the ability to efficiently run Android application unmodified within ns-3. This capability allows us to test WASP within the context of large networks while still being able to measure the impact on a real Android device, and use the same code base for each.

2. RELATED WORK

Industrial solutions to the cellular load problems have revolved around heterogeneous networks (HetNets), a combination of cellular base stations with large and small cells, and Wi-Fi access points. While this can provide relief, it is an expensive solution and leads to significant management complexity. Others have shown the benefits of offloading from an 'expensive' (in some definition) ubiquitous / wide-area network to a local network which is less 'expensive'. For example, using a cellular radio for transmission not only has the high cost of bandwidth, but is also extremely inefficient in terms of battery usage – *e.g.,* for normal use, LTE is 23 times less power efficient when compared to Wi-Fi, and 3G is 14.6 less power efficient [27]. This has led to a number of solutions that: (i) use Wi-Fi access points instead of cellular when available [11], (ii) multi-cast video over Wi-Fi between a small collection of smartphones [31], or even (iii) tether to a friend's phone to share bandwidth caps [6]. Generalizing, these 'mobile ad-hoc networks' don't scale as the number of mobile devices is limited (*e.g.,* [31] can only handle 7-10 mobile devices). Further, to date the resulting systems have placed a great deal of responsibility on the mobile devices themselves and in many cases, limit the generality of the local network. With WASP, we extend the heterogeneous network to any wireless devices and provide a centralized network-wide management. That is, we do not believe that additional access points are a bad thing, we simply do not treat these as special boxes, but as nodes in the system that we can leverage when they are available, each with a different set of costs.

At the other end of the spectrum, Mobile Ad-hoc Networks (MANETs) or mesh networks use routing protocols (e.g., AODV [34], ROMA [22], Serval [10], and Roofnet [16]) among nodes to determine connectivity. With WASP, we look to gain benefits of MANETs for data traffic, but do so targeting a different environment and with different goals. Rather than designed for environments with no pre-existing infrastructure, in which case the network is to be created ad-hoc, WASP is designed for an environment where there is overall connectivity that we can leverage, but wish to limit its use.

We do this with a centralized management scheme. Algorithms have been developed to centrally determine scheduling transmission slots for each radio [21], and channel assignment in wireless mesh networks [36]. With WASP, we designed, built, and evaluated a real hybrid wireless communication system and added a service layer on top of the communication layer. Further, systems such as Meraki's [3], enable the central management of Wireless access points over the web, but are targeted at fixed nodes. Whereas with WASP, we go further and organize mobile nodes (in addition to fixed nodes, which can also run WASP).

Additionally, with WASP we capitalize on different interfaces, one better suited for management traffic (*e.g.,* 3G/4G cellular) and one better suited for data traffic (*e.g.,* Wi-Fi). This is similar in spirit to the separation of the VoIP wakeup notification over the cellular connection, and the VoIP data transmission over Wi-Fi [14].

Finally, we apply techniques from software-defined networking [24, 33] with WASP. The idea of using SDN to offload network control functions in heterogeneous networks which also takes advantage of device-to-device communication is fairly new. There have been some papers which propose similar ideas of leveraging SDN for heterogeneous networks [15, 32], but none had a concrete architecture, implementation, or evaluation as we have with WASP. Further, while there has been a port of OpenVSwitch [35] (which is widely used in software-defined networking) to Android [40], it was only used to enable an application to switch between,

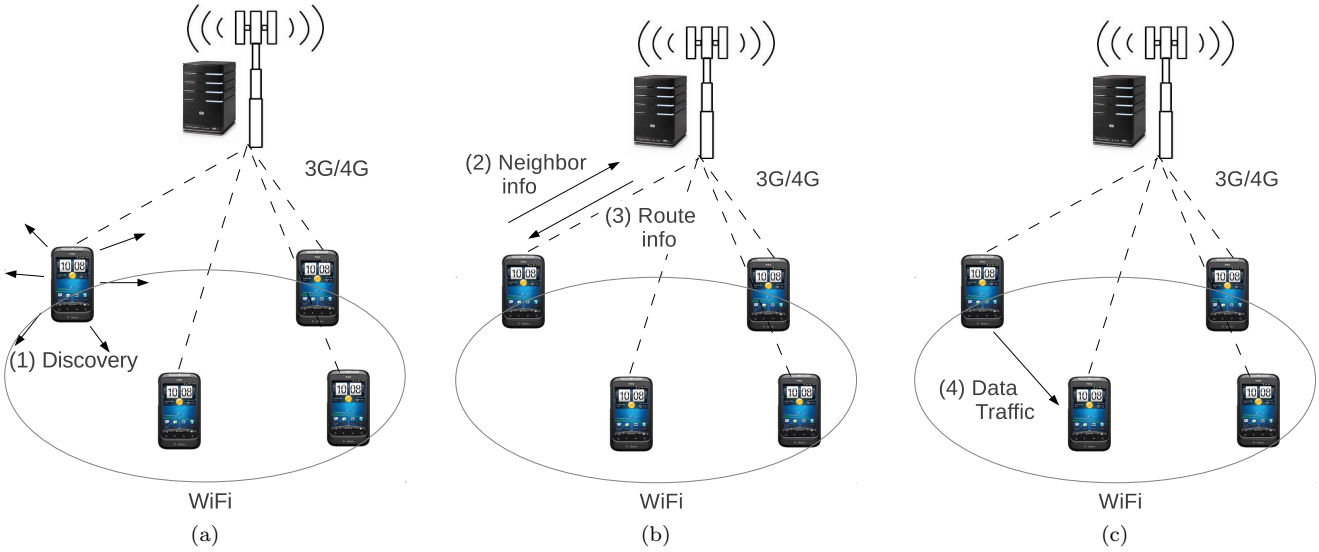


Figure 1: WASP network architecture, showing the controller and mobile devices, and the communication for (a) discovery, (b) control traffic between mobile device and controller, and (c) data traffic between mobile devices.

or use in parallel to maximize bandwidth, multiple interfaces (3G, Wi-Fi). This is more akin to MultiNet [20] or FatVAP [29] than WASP, as there is no network wide control.

3. ARCHITECTURE

Applying software-defined networking (SDN) [24, 33, 4] concepts to a network of mobile devices allows us to leverage ubiquitous wide-area network connectivity for low-rate management traffic, while leveraging ad-hoc connectivity for data traffic where possible. In doing so, it relieves the mobile devices from having to run the typically expensive routing protocols and enables more complex management toward network-wide optimization. As illustrated in Figure 1, WASP consists of a central controller, mobile devices, and a communication protocol among them.

Our architecture might look similar to existing SDN protocol architectures (*e.g.,* OpenFlow [33]), however these protocols were built to primarily target wired networks such as datacenters. As presented in this paper, OpenFlow would probably be sufficient, but we believe going forward, our design will begin to deviate from OpenFlow (*e.g.,* adding support for channels).

3.1 Controller

A distinguishing aspect of WASP is the use of a centralized controller. Rather than intelligence in the nodes themselves, we push functionality into a centralized server as much as possible. The controller can technically reside and be accessed by any means. For example, with the current architecture, WASP relies on a direct connection (via 3G/4G) between a node and the controller. This matches our targeted operation environment where an existing cellular infrastructure has good coverage, but usage needs to be reduced.

Figure 2 shows the main components of WASP controller. The essential goal of the controller is to have a network wide view. This is accomplished by communicating and collecting

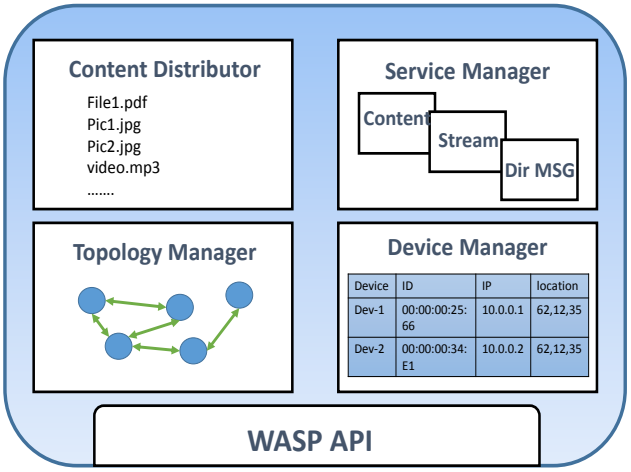


Figure 2: Overview of WASP controller's components.

information from each node through **WASP-API** . This information is managed by **Device Manager** and includes a node's ID, neighbors, data to share, available Internet connections, and any other information we can easily add in the future (*e.g.,* location, battery). The controller uses this information to calculate the routes among nodes, through **Topology Manager** , and sends the routing tables back to the nodes to indicate what the next hop is for each destination. Importantly, with this architecture, the controller can evolve to calculate routes differently without requiring software updates to each mobile device. We can continuously add more complexity into the algorithms within the controller, and if, for example, we find a more optimal assignment of paths, we can notify the mobile devices of the change. The controller can also be used as a **Content Distributor** between the Internet and the mobile devices. As we will see in the evaluation in Section 6, if one of the nodes

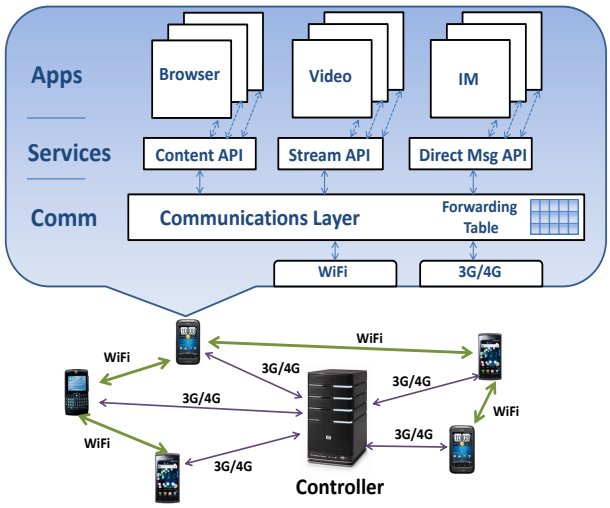


Figure 3: Overview of WASP device architecture.

in WASP's network requests data (*e.g.,* picture, video) and this data is not cached by other nodes in the network, the WASP controller can provide the requested data directly to the node. Another feature of WASP is that controller can manage and add more services, through **Service Manager** , and easily push them to all nodes in the network. We discuss WASP's service in more details in Section 4.

We assume that all nodes in the WASP network are registered in the controller with some unique identity. In our current implementation we use nodes' MAC addresses and can easily change it in the future for any better identification method. This helps the controller to assign unique IP addresses to the nodes and manage sub-nets to create node clusters in the network if needed. The controller keeps track of the nodes through the neighbor list messages and keep-alive messages sent from each node. If a node does not contact the controller in a specified (configurable) amount of time, and it is not reported to be a neighbor of another node, the controller will assume that this node is unreachable and delete it from network, recalculate the route, and send the updates to the other reachable nodes.

3.2 Nodes

In WASP, we take advantage of nodes equipped with multiple interfaces. We use the interface connected to the ubiquitous network to send information to the controller such as the node's ID, neighbor updates, and keep-alive messages. In return, the same interface is used to receive information from the controller, such as IP assigned and routing tables. The other interface is used for local communications among the nodes to send and receive periodic Hello messages. It is also used to send, receive, and forward data packets.

As we can see in Figure 3, at the lowest layer, the WASP communication layer provides low-level functions such as sending a message to a given node, receiving a message from another node, sending a message or query to the controller, and receiving a message from the controller. The WASP communication layer then uses the underlying socket interface to communicate with controller and each next-hop neighbor.

Table 1: WASP Control Messages

Type (from-to)	Name	Description
Node-Controller	Node-ID	Node sends its ID to request an IP address in first time registration or reconnecting
Node-Controller	Neighbor-Update	Sent when node's neighbor table is updated
Node-Controller	Find-Data	Sent when node requests data and needs to know if any other node in the network has it (*e.g.,* file)
Node-Controller	Keep-Alive	Sent periodically (default 5sec) unless node is sending any of the previous messages
Controller-Node	Assigned-IP	Reply with IP address to Node-ID message
Controller-Node	Routing-Table	Sent to nodes that are affected by any change in network topology
Controller-Node	Data-Node	Node-ID that host the requested data in Find-Data message, or Data itself if no node hosts the data requested
Node-Node	Hello	Sent periodically (default 1sec) unless node is sending or forwarding data packets

The WASP services are additional packages that use interfaces provided by the communication layer. Each WASP service provides an inter-process communication API, which allows bound mobile device applications to utilize its functionality. APIs are specific to services, such as: *get_content(URI)* for the web content service. Multiple applications can bind to one or more WASP service simultaneously. One example service is the web content distribution service which allows web interfaces (*e.g.,* a browser) to exchange content within the WASP mobile-to-mobile network. We discuss WASP services in more details in Section 4.

Each node in the WASP network maintains two tables. The first one is the neighbor list table. The table contains neighbor's ID and IP(assigned by the controller) and its time stamp. Every time the node receives a hello message from another node, it updates the table by either adding the neighbor, if it not existed already, or refreshing the neighbor's time-stamp. If a timer of one of the neighbors timeouts, the node deletes the neighbor from its list. In case of adding or deleting neighbor, the node updates the controller with such information. The second table is the forwarding table. It contains next-hop(neighbor ID) and destination ID fields. The forwarding table is updated whenever the controller sends route update to the node. Once the node receives a data packet to be forwarded, it looks at the packet's header and match the destination ID with the corresponding next-hop in the routing table. The node then uses the neighbor list to lookup the next-hop IP and forward the packet to it through the local communication channel.

3.3 Communication

As we mentioned in the previous two subsections, WASP manages the network through two types of control messages. One is between controller and nodes (controller-messages) and the other type is among nodes (hello-messages). Table 1 shows WASP control messages with brief description.

To reduce the number of messages sent from nodes, the node does not need to send keep-alive messages if it has re-

cently sent the controller a neighbor update messages. Once the controller receives any message from the node, it updates its timer and considers the node reachable. At the same time, after the controller calculates the routes, rather than sending the entire routing table of the network to all nodes, it only sends routing table updates (diffs) to the nodes affected by the topology change. This approach considerably reduces the number and the size of messages sent by the controller.

4. LAYERED SERVICES

The ability to communicate within a network of mobile devices only has benefit if applications can and do use it. As such, with WASP we layer services on top of the communication layer as a mechanism to optimize the specific communication pattern.

In general, we have completed the direct communication and web services, and are in the process of adding a streaming service, and can and will expand as future work. In each case, there are alternate, probably better, implementations, but we do not go into an in-depth exploration here. These are merely meant to demonstrate the possibilities.

4.1 Direct Communication

More and more, mobile devices are already directly communicating between mobile devices. Of course, that was the original functionality of mobile devices before data plans became popular – to make calls to other mobile devices, and then later on to send text messages to other mobile devices. With data traffic, this direct communication has evolved to -include voice, video, or text chat (instant messaging). Further, services such as Samsung AllShare Play [9], are pushing applications in the direction of mobile device-to-mobile device communication.

Extending an application to use the WASP communication layer is trivial as the communication is direct already. However, we need the application to know whether the other mobile device is within local (multi-hop) range or not, transparent to the user. To do this, a lookup (much like DNS) to the controller, will indicate whether a given user ID is currently in the same network as the inquiring mobile device, and if so, what the node identifier is.

4.2 Web Content

Web content has been shown to follow a Zipf distribution where a relatively few pieces of content are accessed most of the time [17]. Even more, studies have shown that the popular content is getting even more popular [28]. This has provided the basis for in-network content caching [37] and peer-to-peer content distribution networks [23]. We extend this concept into a mobile device-to-mobile device content distribution network with the WASP Web content service.

With the WASP Web content service, each mobile device maintains a cache of content recently accessed and can act as a server to serve up that content – capitalizing on the caching of popular content to alleviate cellular use and to help with flash crowds (the Slashdot effect). This has been explored with FireCoral [38], but with WASP the cache is not tied to the web browser. In fact, we would not be surprised if the locality of access in a network of mobile devices is greater than found in a p2p network of computers connected over the Internet. Off course, this needs to be further studied, but our belief is based on the fact that the mobile device network has physical locality, which may introduce a bias of like individuals (*e.g.,* college students on a campus). The logically central controller (or for future exploration, other tracking mechanism) maintains a tracker to track which mobile device has cached which content. This means that the mobile devices should (and eventually must) notify the central controller of any changes to the cache.

The general flow of an access is an application uses the WASP content service (running on the same mobile device) to request a URI. The WASP content service contacts the central tracking server to find which mobile devices have cached that URI, the server responds with a list of mobile devices that are within some hop-count radius, the requesting mobile device's WASP content service then requests the URI from one of the mobile devices in the list, and if that mobile device still has the content (*i.e.,* the tracker was up-to-date), the mobile device will return the content (and if it doesn't have the content, the requesting mobile device will try the next mobile device, and eventually can fall back on the Internet over the cellular connection). At this point, the requesting mobile device caches the content, possibly evicts some other cached item, and notifies the central tracking server.

We acknowledge that there may be some privacy issues in this scheme such as other users knowing what sites a given user is visiting. However, this is not the main focus of this paper and used as an example to illustrate how an underlying mobile-to-mobile network can benefit content distribution.

4.3 Streaming Service

While the WASP streaming service is still in development phase and we leave its evaluation as future work, we discuss it here not to indicate that it is in itself a contribution, but to provide further understanding of the potential of WASP.

The streaming service provides an efficient distribution means for an emerging communication pattern. As smart mobile devices become a more integral part of our lives and as connectivity and processing power of the devices increases, so to will our consumption of streaming media such as live sports. In many cases, multiple mobile devices will be receiving the same stream. With WASP, rather than each mobile device getting the stream over the cellular connection, a single stream can be transferred over cellular and then the mobile devices collectively distribute the stream over the Wi-Fi direct mobile device-to-mobile device network. As an aside, a special case of this is where one of the mobile devices is generating the stream, in which case we can limit the cellular traffic to zero.

An effective mechanism is a multicast tree to distribute the stream. The challenge in this case is fairness – *e.g.,* the node that downloads the stream over the cellular connection is at a disadvantage from a cost and power consumption standpoint. Here, we can leverage the approach taken with SplitStream [19] which divides the stream into multiple substreams and distribute each with a different multicast tree. In fact, there has been further research which extends this (each of which could be incorporated into WASP as future work) – SV-BCMCS [26] leverages proximity to the basestation as an additional consideration, and Microcast [31] capitalizes on the broadcast nature of Wi-Fi.

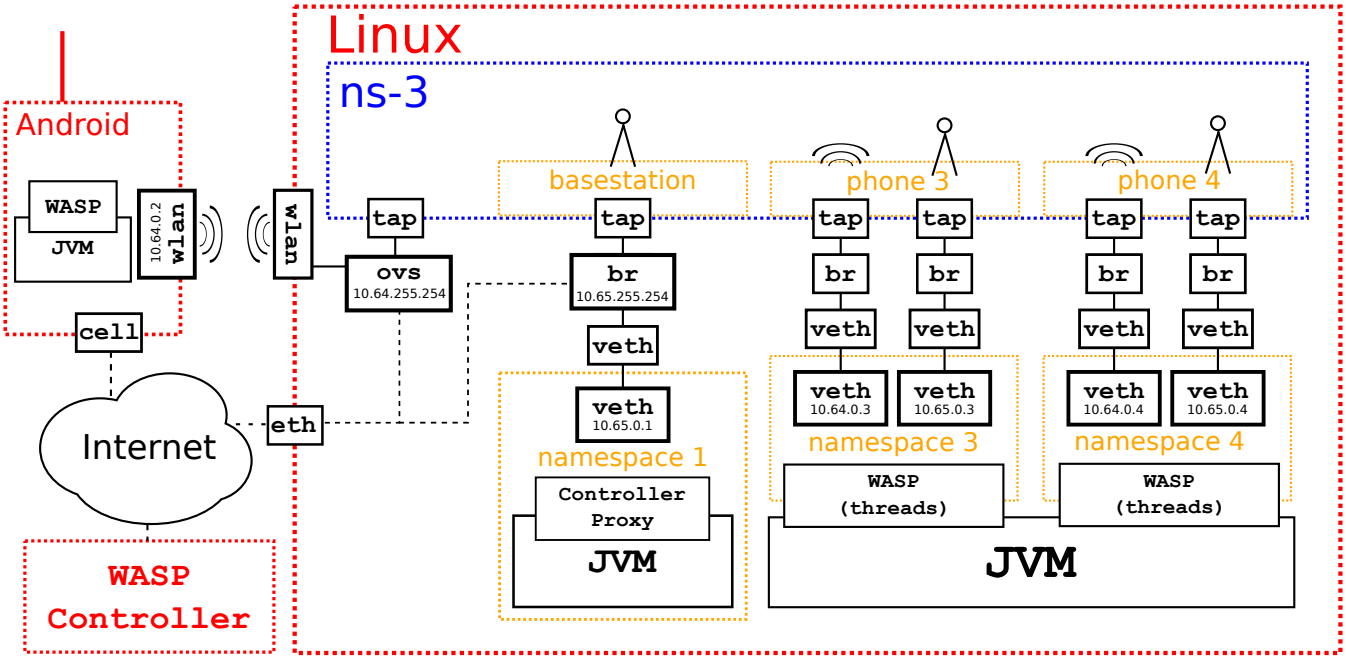


Figure 4: WASP simulation with real Android application and phone-in-the-loop.

5. IMPLEMENTATION

In this paper, we focused on the scenario of hybrid cellular networks to implement WASP. We believe WASP can be implemented on any type of hybrid networks as long as mobile devices equipped with multiple interfaces.

5.1 Controller Implementation

While WASP's architecture is flexible, for the case of a cellular network, we envision the logically centralized controller being run as a service by the cellular company in order to optimize the network (likely coupled with a hot-swappable backup server). However, a 3rd party could also assume this role since we can communicate with any server on the Internet over the cellular connection. The controller resides in the Department of Computer Science at the University of Colorado main campus with a public IP address. The controller is a server machine running Ubuntu OS with Intel-Core-i7X2GHz CPU and 8GB of RAM. We wrote 3000 lines of Java code to implement WASP's Controller prototype. Currently, the controller implementation calculates the all-pairs shortest path – a solution which does not scale well due to the algorithm complexity. We believe better approaches can be used such as clustering the network to smaller groups and manage them individually, or incremental shortest path algorithms.

5.2 Node Implementation

We built an initial prototype of WASP on top of the Android platform. WASP is implemented as a service within Android such that it runs in the background and has higher priority. For communication with the controller, each phone opens a TCP connection (through the 3G/4G interface) with the controller. The connection is used to send and receive messages and can stay in idle mode without disconnecting for up to 30 minutes depending on the cellular company [39]. Neighbor discovery and data packet forwarding are performed with the Wi-Fi interface and controlled by the WASP communication layer as well. We verified the operation of WASP and associated test applications on a small collection of five Google Nexus 4 devices.

5.3 Extended ns-3 Testbed

To perform evaluations beyond a small collection of phones we made use of the ns-3 simulation environment [5]. We implemented an extension to ns-3 which provides the ability to efficiently run Android application unmodified within ns-3. Doing so allows us to test and evaluate WASP on real devices as well as within the context of large networks, while being able to use a consistent code base. This includes the WASP communication layer, WASP service layer, and an application which uses a WASP service. For the application, we are only referring to the logic of the application. The user interface is not intended to run in both ns-3 and on the phone as we do not intend to support graphical interfaces. We implemented the same extension to AODV and OLSR to compare their performance to WASP. We refrained from using the ns-3 implementation of AODV and OLSR for two reasons. First, we wanted the code of WASP in ns-3 to be the same as the Android application, and for the most direct comparison with WASP we leveraged an Android implementation of AODV and OLSR. Second, ns-3's current implementation of AODV has an implementation issue that limits the protocol's scalability and performance (as discussed in various support forums, such as [13]). With our AODV extension to ns-3, AODV performance is greatly increased. We used the existing open source Java implementation of AODV and OLSR in our testbed and comparison to WASP [1, 8].

Each simulation includes four types of nodes which are shown in Figure 4.

The **basestation** is the LTE access point. It mediates the LTE network and is the Internet gateway. The basestation is not part of the Wi-Fi network.

The **controller** is an application running at an arbitrary location on the Internet. For our testbed this location is a nearby virtual machine, however deployments can place this anywhere from a datacenter to the basestation.

The bulk of the testbed consists of a configurable number of **emulated nodes** with both LTE and Wi-Fi interfaces. To achieve the ability to run unmodified Android applications, each node combines an ns-3 `Node` class with the WASP client running in a Linux network namespace. The network namespaces isolate each network stack. Within each network namespace are the WASP threads representing the different parts of WASP. For efficiency, we run a single JVM for all WASP nodes, but are able to assign a collection of threads representing a single WASP node into its own network namespace. To provide the connectivity between ns-3 and the WASP client code, each interface is implemented as a chain including an ns-3 `NetDevice` class, layer 2 tap device, Linux bridge, and a virtual Ethernet pair.

In extending ns-3, we opened up a new possibility that we are currently working on – **phone-in-the-loop** (shown in the left side of Figure 4). Hardware-in-the-loop has been used extensively in testing of embedded systems. We are replicating this with the ns-3 environment with a *phone-in-the-loop* extension to verify that WASP works on a phone at larger network scales. The phone uses Wi-Fi to connect to the testbed server, but is additionally on the ns-3 emulated Wi-Fi network. The phone-in-the-loop uses our carrier's actual LTE basestation and connects to the controller over the Internet. (Note that in Figure 4 we use Open vSwitch [7] for bridging because many wireless chipsets are not supported on Linux bridging). With this, the logic for a given node (or set of nodes) within ns-3 is being run across a real wireless channel on a real phone. As it is still early stage, we leave the evaluation and discussion of phone-in-the-loop for future work, but mention it here to provide a view into the possibility of the extended ns-3 platform.

The testbed is configured by two programs. There is an ns-3 `C++` program for creating the Phy and MAC layers of the Wi-Fi and LTE networks. Additionally a shell script creates all of the necessary devices and assigns IP addresses within each namespace (in real life, the controller will be the one assigning IP addresses to the nodes). The script also sets up Internet access by assigning IP addresses to the appropriate bridges and establishes NAT rules in `iptables`.

6. EVALUATION

In this section we evaluate the performance of WASP implementation, the benefit of our centralized approach, and compare WASP to AODV and OLSR protocols under different network settings. We start by presenting experiment setup in ns-3, evaluate packet delivery ratio and number of control packets of WASP, AODV, and OLSR. We then show the amount of traffic WASP's used through LTE. Further, we show the benefit of using WASP in interesting applications such as content distribution and flash crowd event. Finally, we describe and evaluate power consumption in WASP, AODV, and OLSR.

6.1 WASP's Performance vs. AODV and OLSR

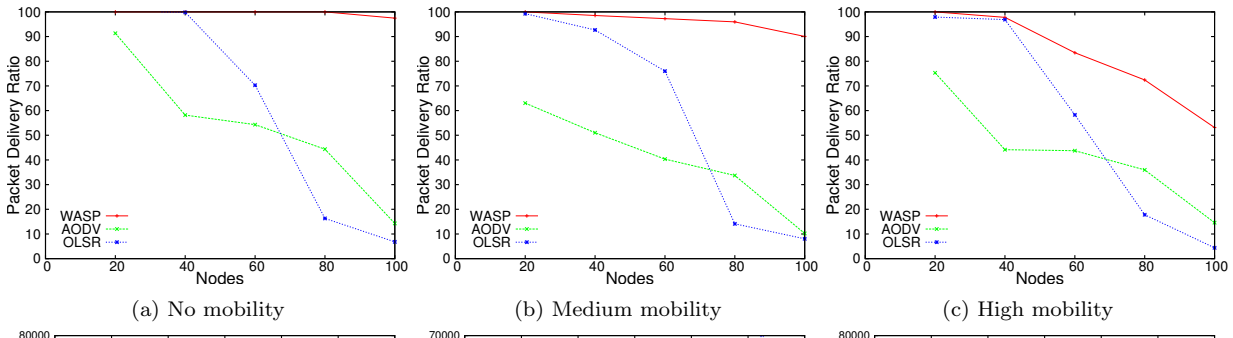
In the following experiments we measure packet delivery ratio and total number of control packets from all nodes generated in each protocol. We designed two sets of network setups – one for fixed density and another one for fixed radius. In fixed density experiments, the density of the nodes in the simulation grid is fixed by increasing the radius as the number of nodes increases – allowing us to understand the effect the number of nodes has on performance. In fixed radius experiments, the number of nodes increases while the grid area is constant – allowing us to understand the effect increased density has on performance. With this, we can understand the performance of WASP in various scenarios. Table 2 shows number of nodes and corresponding grid radius and mobility of nodes. For each experiment we pick 10% of the nodes to be senders and 10% to be receivers. The rest of nodes simply serve to forward packets (if the controller selects a given node as part of a path). The senders simultaneously send 300 packets at the rate of 1 packet every 100ms. We ran each experiment 5 times and took the average.

Table 2: WASP vs. AODV and OLSR Experiment Parameters

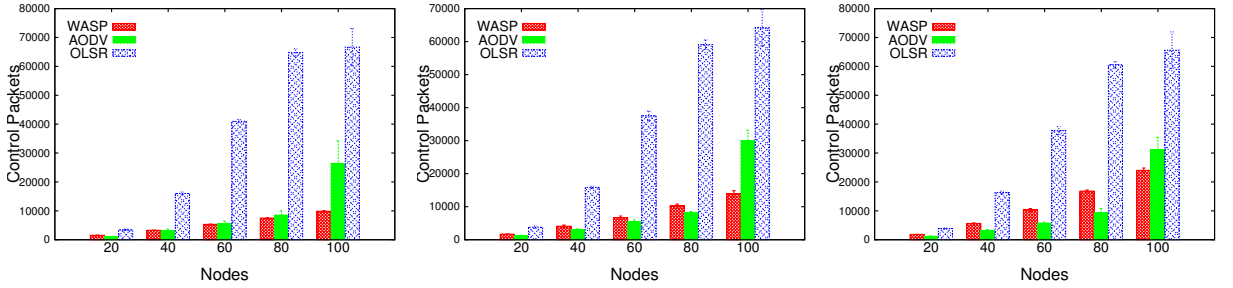
	Fixed Radius	Fixed Density
Radius (meter)	250	250, 356, 437, 504, 564
Number of nodes	20, 40, 60, 80, 100	20, 40, 60, 80, 100
Mobility (m/s)	medium(1m/s)	station(0m/s), medium(1m/s), high(5m/s)
Broadcast Interval (s)	1 (WASP, AODV, OLSR)	1 (WASP, AODV, OLSR)
Neighbor Timeout (s)	3 (WASP, AODV, OLSR)	3 (WASP, AODV, OLSR)

As we see in Figures 5a, 5b, and 5c, WASP achieves higher packet delivery rate than OLSR and AODV under the same network topology, mobility and load setup. We believe this is because scalability of MANETs is limited by the fact that high volume of control messages must be exchanged and processed (e.g, in the case of OLSR) and the high packet processing overhead which causes the node to drop packets (e.g, in the case of AODV). A distinguishing feature of WASP is that we apply software-defined networking (SDN) principles and shift all the control overhead to the controller. In this way, WASP's nodes avoid exchanging large number of control packets for topology changes and the number of control packets increases linearly and proportionally to the number of nodes and their mobility as seen in Figures 5d, 5e, and 5f.

In fixed radius networks, we wanted to see the impact of making the network as dense as possible. As expected, AODV and OLSR overwhelm the network with high volume of control packets as seen in Figure 6b. This causes both data packets and many control packets to be dropped since every node is waiting for the medium to be free to send its packets. With WASP, our control traffic is significantly lower. This stems from the fact that the amount of routing control messages in OLSR and AODV grow non-linearly with the network size, as compared to WASP for which there is a linear increase. This is in contrast to discovery control messages (hello messages) which grows proportional to the network size (since each node sends at a fixed rate) in each approach (WASP, AODV, OLSR). In reducing local control messages, the network is freed for more data packets to be delivered – leading us to a considerable high packet delivery ratio, as seen in Figure 6a.

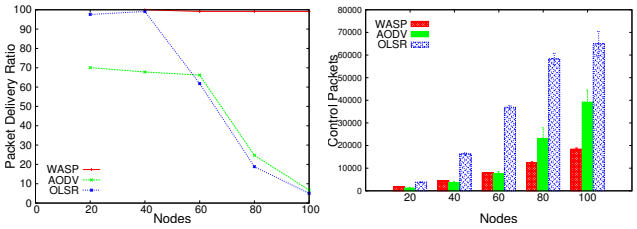


(a) No mobility (b) Medium mobility (c) High mobility



(d) No mobility (e) Medium mobility (f) High mobility

Figure 5: Performance of WASP, AODV, and OLSR under fixed density networks. (a), (b), and (c) show packet delivery ratio. (d), (e), (f) show total number of control packets generated from all nodes.



(a) Packet delivery ratio (b) Number of control packets

Figure 6: Performance of WASP, AOVD and OLSR under fixed radius networks (a) shows packet delivery ratio. (b) shows total number of control packets generated from all nodes.

In addition to delivery performance, the amount of control traffic is important in WASP. We measured the aggregate control traffic sent over LTE over time for a network with 100 nodes within a radius of 1000m. In this experiment, each node will move at high mobility. As seen in Figure 7, the aggregate control traffic over LTE for all 100 nodes is minimal.

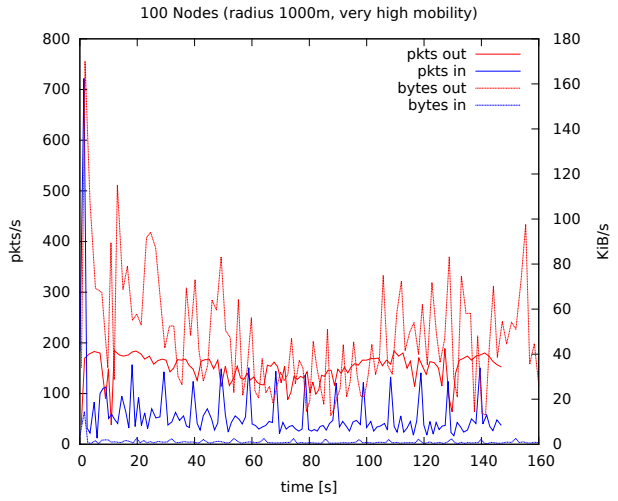


Figure 7: Control traffic over LTE over time for 100 nodes with high mobility.

6.2 Throughput and Latency

As WASP is based around the ability for participating phones to relay traffic among phones via the Wi-Fi connection, an important metric is the throughput and latency of this forwarding. To determine this, we used three phones (Google Nexus-4 labeled A, B, and C) arranged in a line (*i.e.,* phone B has a Wi-Fi link to both A and C, but A and C do not have a link between them). In this setup, B will forward traffic between A and C. We used the iperf for Android application to measure throughput [2], and our own custom application to measure latency between A and C. The measured throughput was 7.096 Mbps and the round-trip time was 11.668 ms. This compares to the direct communication between two phones measured at 12.753 Mbps throughput and 10.559 ms round-trip time. The decrease in throughput and increase in latency is to be expected with an extra hop, but the numbers are fairly reasonable given our application level forwarding in an unoptimized implementation.

6.3 Battery Consumption

While it may appear that using a cellular channel for control may lead to a design which is significantly less power efficient – *e.g.,* for normal use, LTE is 23 times less power

efficient when compared to Wi-Fi, and 3G is 14.6 less power efficient [27]. Fortunately, we show that this is not the case. A brief analysis of network and computation cost on mobile device for running WASP protocol will be given, followed by experiment setup for measuring power consumption and the experiment results on WASP, AODV and OLSR. The activities of mobile devices running WASP protocol include: (i) periodically sending and receiving "Hello" message and data packets via Wi-Fi, and (ii) sending and receiving control packets as needed to and from controller via LTE. The computation cost is small as generating the neighbor list and performing routing table lookups are $O(n)$. The major power consumption is the network load – sending and receiving control and data packets.

To learn the scalability of WASP and power consumption characteristics when the network size increases, data needs to be collected for a large network. As the major power consumption is network traffic on mobile device, we took a log file for each device sending and receiving packets via Wi-Fi and LTE in the previous ns-3 experiments, and then simulated the sending and receiving behavior on real phone (this removed any computation, which AODV and OLSR have significantly more than WASP). The simulation on the phone exactly follows the log, *i.e.,* the time interval, packet size, and network. During simulation, the screen is off and all other apps on phone are disabled.

Figure 8 shows the battery trace for a phone running WASP, AODV, OLSR, LTE (Where the phone is receiving all the data through the LTE interface), and IDLE (where the phone is not transferring any data) under fixed density and medium mobility experiment with 100 nodes. WASP is 7% less energy efficient than AODV and OLSR. This is the trade off for performance as in this size of 100 nodes and type of mobility, WASP is delivering 95% of the packets while AODV and OLSR are delivering less than 10%.

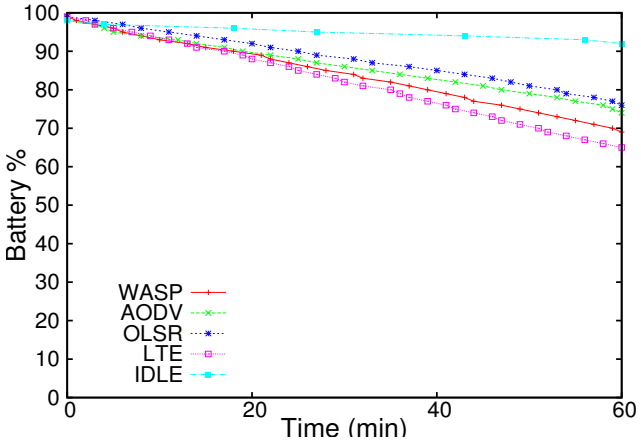


Figure 8: Battery log under fixed density and medium mobility for 60 minutes.

6.4 Content Distribution

From the applications perspective, the usefulness of WASP depends on the ability to capitalize on the phone-to-phone network for data traffic. To evaluate this, we explored the web content service to show amount of LTE bandwidth saved for different parameters – namely the network size, cache size, and acceptable number of hops.

In this experiment, each application will be continuously fetching content according to the following model (as described in [17]): 1) Content size varies according to a Weibull distribution. We use a total of 10,000 objects (effectively cutting off the long tail). Each of these objects is assigned a URI and a size based on the distribution. 2) Content popularity follows a Zipf distribution. When fetching content, each node will select a URI based on this distribution. 3) Content is continuously fetched with the interval between each request being chosen at random between 0 and 3,000 milliseconds. 4) Content is cached with a fixed size cache with a least recently used eviction scheme.

Shown in Figure 9 is the amount of savings as a function of the network size. Interestingly, the network size seems to have little effect as we can get roughly a 35% saving in the amount of cellular traffic.

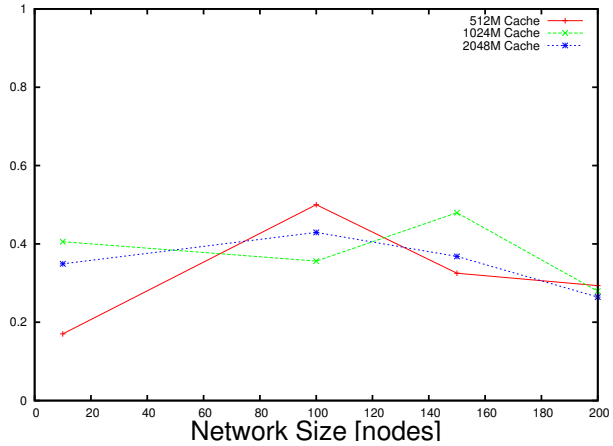


Figure 9: Fraction of traffic serviced locally as a function of network size.

In addition to providing benefits for average-case (Zipf) distribution of web requests, WASP can be a great fit in supporting a flash crowd (also known as the the Slashdot effect), where there is a single, or small set, of content that is fetched by many at roughly the same time. To evaluate this, we provided a similar setup to what was used to evaluate the Coral CDN's relief on the web server during a flash crowd [23] In our setup within ns-3 we use 200 nodes, each mobile with using `RandomWaypointMobilityModel`. Each node is configured to access the same 3 objects starting at a randomly distributed start time between 0 and 180 seconds. We limit the number of hops away that a node can request the content from to be 3 hops. Shown in Figure 10, we can see that an initial spike of requests over LTE, as more phones access the content, the traffic over LTE quickly drops.

7. DISCUSSION

In this section we discuss WASP's possible limitations and how to overcome them in our future work.

7.1 Incentives to use WASP

One challenge WASP faces is the incentive for users to participate in the network as it requires users to forward traffic (consuming battery) on behalf of others. While a setting such as the military can force the use as the overall benefit to the collective network is greater with WASP than

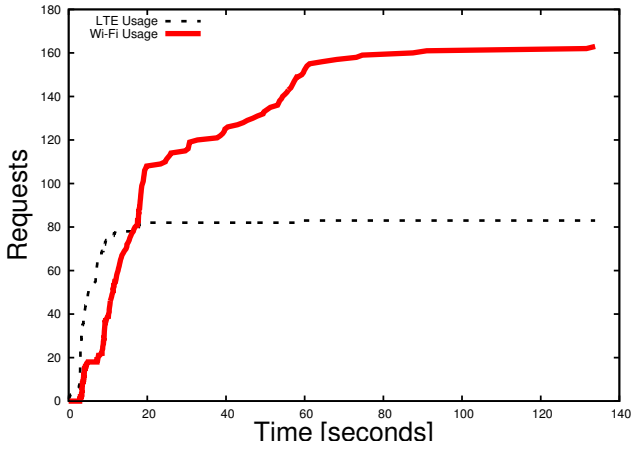


Figure 10: WASP handling a flash crowd.

without, we understand this might be a barrier for commercial networks such as cellular networks where each user is independent. We believe that overtime, everyone benefits. Even so, a barrier still remains as it is human nature to want to see evidence they are benefiting. To overcome this challenge, we envision a credit based system applied to WASP and managed by the controller. The credits can be monetary or better service for nodes participating. The controller will add/subtract credits based on node's role in the network (*e.g.,* nodes requesting data will use some credits while nodes sending or forwarding traffic earn credits). This type of scheme may fit well within the new pricing structures being explored [25].

7.2 Wi-Fi Access Points

A push by industry is to offload data from the cellular infrastructure to Wi-Fi access points, when they are available. Unfortunately, finding Wi-Fi access points can be challenging (Even in highly dense environments such as stadiums [30]) and accessing shared Wi-Fi access points often results in poor performance (due to many users accessing it). In an attempt to overcome these issues, node-to-node tethering apps are appearing to allow sharing of cellular data plans [6], leading to different management for when Wi-Fi access points are available and for when they are not. Since WASP is generally targeting any hybrid network, we don't only focus on mobile devices to build our network. We view Wi-Fi access points as simply another node in the network that also has two interfaces – wired Ethernet for the 'one hop' connection (via the Internet) to the controller (optimized for long distance), and the Wi-Fi interface for communicating locally (optimized for local area). Each link simply has a cost associated with it (not just in terms of dollars) – *e.g.,* LTE might be higher cost than using Ethernet. The cost, by design, is flexible and can dynamically change. In current open Wi-Fi access points, the link from the access point to the controller might be considered free, but our model is more flexible and supports future business models and can incorporate additional costs such as where the access point is overloaded.

7.3 Energy Consumption

Although WASP considerably outperforms AODV and OLSR, it is (slightly) less energy efficient than AODV and OLSR. We believe this is because WASP uses the ubiquitous expensive channel to communicate with the controller. We believe we can reduce WASP's energy consumption by reducing the number of packets sent and received between the node and the controller. From the node's prospective, this can be done by allowing the node to wait for a little amount of time before updating the controller of any neighbor missing. With the nature of wireless networks, hello packets might be dropped or not reached easily.

7.4 Deployment on Android Today

One of the challenges we had in order to deploy WASP on Android phones was the Wi-Fi Direct interface. Wi-Fi ad-hoc mode is not accessible to unrooted phones, and Wi-Fi Direct requires user interaction to establish connections between phones which we find unpractical to our system. (WASP, using Wi-Fi ad-hoc, could authenticate connections automatically, eliminating the need for the interaction). In either case, a simple update to Android would make WASP instantly deployable.

8. CONCLUSIONS AND FUTURE WORK

In this paper we presented the design, implementation, and evaluation of WASP. WASP is built on the idea that by leveraging different interfaces which are optimized for different needs, we can provide a more efficient and scalable network for mobile devices. We show through our extended ns-3 simulation environment that WASP is more scalable than existing ad-hoc protocols, at only a small energy penalty (due to control messages going over LTE), Further, through the network WASP manages, we are able to save about 35% of traffic that would otherwise be served over cellular.

As future work we are looking to roll WASP out to real users once we overcome the Wi-Fi direct limitation. Beyond live user testing, WASP opens up much future work in evolving the control of the network and services run on the phones. Optimizing the services for the various trade-offs involved will require an in depth study of each. Further, we experimented with certain design decisions but the design space has much more to explore.

9. REFERENCES

[1] Adhoc-on-android. https://code.google.com/p/adhoc-on-android/.
[2] iperf for android. https://play.google.com/store/apps/details?id=com.magicandroidapps.iperf.
[3] Meraki. http://www.meraki.com/.
[4] Nicira Networks. http://nicira.com/.
[5] ns-3: A discrete-event network simulator for internet systems. http://www.nsnam.org.
[6] Open garden. http://opengarden.com/.
[7] Open vswitch: Production quality, multilayer open virtual switch. http://www.openvswitch.org.
[8] Running olsr on android phones. http://www.olsr.org/?q=olsr_on_android.
[9] Samsung AllShare Play. http://www.samsung.com/us/2012-allshare-play/.
[10] Serval: communicate anywhere, anytime. http://www.servalproject.org/.

[11] Serval demo: Client migration. http://www.serval-arch.org/demos/client-migration/.

[12] rfc 3626: Optimized link state routing protocol (olsr). http://www.ietf.org/rfc/rfc3626.txt, 2003.

[13] ns-3-users Google Groups: routing performance problem in large scale MANET. https://groups.google.com/d/msg/ns-3-users/1tq8kFuoy1Y/cjnlh9LDdJwJ, Nov. 2012.

[14] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta. Wireless wakeups revisited: energy management for VoIP over Wi-Fi smartphones. In *Proc. conference on Mobile systems, applications and services (MobiSys)*, 2007.

[15] H. Ali-Ahmad, C. Cicconetti, A. de la Oliva, M. Draxler, R. Gupta, V. Mancuso, L. Roullet, and V. Sciancalepore. Crowd: An sdn approach for densenets. *2013 Second European Workshop on Software Defined Networks*, 0:25–31, 2013.

[16] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *Proceedings of the 11th annual international conference on Mobile computing and networking*, MobiCom '05, pages 31–42, New York, NY, USA, 2005. ACM.

[17] L. Breslau, P. Cao, L. Fan, G. Phillips, , and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *Proc. IEEE INFOCOM*, Mar. 1999.

[18] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. Gude, N. McKeown, and S. Shenker. Rethinking enterprise network control. *IEEE/ACM Transactions on Networking*, 17(4), Aug. 2009.

[19] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: high-bandwidth multicast in cooperative environments. In *SOSP*, 2003.

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

[21] S.-M. Cheng, P. Lin, D.-W. Huang, and S.-R. Yang. A study on distributed/centralized scheduling for wireless mesh network. In *Proc. conference on Wireless communications and mobile computing (IWCMC)*, 2006.

[22] A. Dhananjay, H. Zhang, J. Li, and L. Subramanian. Practical, distributed channel assignment and routing in dual-radio mesh networks. In *Proc. SIGCOMM*, 2009.

[23] M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with coral. In *Proc. Symposium on Networked Systems Design and Implementation (NSDI)*, Mar. 2004.

[24] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A clean slate 4d approach to network control and management. *SIGCOMM Comput. Commun. Rev.*, 35(5):41–54, Oct. 2005.

[25] S. Ha, S. Sen, C. Joe-Wong, Y. Im, and M. Chiang. TUBE: Time-dependent Pricing for Mobile Data. In *Proc. SIGCOMM*, 2012.

[26] S. Hua, Y. Guo, Y. Liu, H. Liu, and S. S. Panwar. Scalable video multicast in hybrid 3g/ad-hoc networks. *Trans. Multi.*, 13(2):402–413, Apr. 2011.

[27] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. A close examination of performance and power characteristics of 4g lte networks. In *Proc. conference on Mobile systems, applications, and services (MobiSys)*, 2012.

[28] S. Ihm and V. S. Pai. Towards understanding modern web traffic. In *Proc. conference on Internet measurement conference (IMC)*, IMC '11, 2011.

[29] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi. FatVAP: aggregating AP backhaul capacity to maximize throughput. In *Proc. Symposium on Networked Systems Design and Implementation (NSDI)*, 2008.

[30] P. Kapustka. NL West Leads MLB Stadium Wi-Fi Scorecard, with 4 out of 5 Teams Offering Network Service to Fans. http://www.mobilesportsreport.com/2013/03/nl-west-leads-mlb-stadium-wi-fi-scorecard-with-4-out-of-5-teams-offering-network-service-to-fans/.

[31] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, and A. Markopoulou. MicroCast: cooperative video streaming on smartphones. In *Proc. conference on Mobile systems, applications, and services (MobiSys)*, 2012.

[32] K. O. Marc Mendonça, Bruno Astuto A. Nunes and T. Turletti. Software defined networking for heterogeneous networks. *IEEE COMSOC MMTC E-Letter*, 8:36–39, May 2013.

[33] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, Mar. 2008.

[34] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. IETF RFC 3561, July 2003.

[35] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, and S. Shenker. Extending networking into the virtualization layer. In *Workshop on Hot Topics in Networks (HotNets)*, Oct. 2009.

[36] A. Raniwala, K. Gopalan, and T.-c. Chiueh. Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 8(2):50–65, Apr 2004.

[37] E. J. Rosensweig and J. Kurose. Breadcrumbs: Efficient, best-effort content location in cache networks. In *IEEE INFOCOM*, 2009.

[38] J. Terrace, H. Laidlaw, H. E. Liu, S. Stern, and M. J. Freedman. Bringing P2P to the Web: Security and Privacy in the Firecoral Network. In *IPTPS*, 2009.

[39] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang. An untold story of middleboxes in cellular networks. In *Proc. SIGCOMM*, 2011.

[40] K.-K. Yap, T.-Y. Huang, M. Kobayashi, Y. Yiakoumis, N. McKeown, S. Katti, and G. Parulkar. Making use of all the networks around us: a case study in android. In *Proc. workshop on Cellular networks (CellNet)*, 2012.