Feasibility of Content Dissemination Between Devices in Moving Vehicles

Thomas Zahn, Greg O'Shea, Antony Rowstron
Microsoft Research, Cambridge, U.K.
gregos@microsoft.com, antr@microsoft.com

ABSTRACT

We investigate the feasibility of content distribution between devices mounted in moving vehicles using commodity WiFi. We assume that each device stores content in a set of files, and that each file has a version number. When two devices come into wireless range, they attempt to synchronize the latest versions of any files they have in common. This is challenging because connections are often short-lived and have variable link quality. Prior work demonstrates that current protocols perform badly under these conditions. To motivate this work, we use the example of Personal Navigation Devices (PNDs), or SatNavs, where the content to be exchanged includes maps and points-of-interest files.

We describe a protocol enabling devices in vehicles to identify and exchange content of shared interest. We evaluate the protocol using a small vehicular testbed in two urban locations and on a highway with a closing speed of 140MPH. We investigate the effects of using 802.11a versus 802.11g, placing the antenna inside or outside the vehicle, and varying the packet size. We transfer up to 70MB in the urban settings and 7MB on the highway.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication; C.2.2 [**Network Protocols**]: Applications

General Terms

Algorithms, Design

Keywords

Vehicular, Mobility, Wireless

1. INTRODUCTION

Vehicle mounted devices, such as Portable Navigation Devices (PNDs), Satellite Navigation Devices or GPS devices, which are mounted on a vehicle's windshield, are becoming increasingly popular. In Western Europe already 25% of vehicle owners have a PND. In this paper, we describe Broadside, a system which demonstrates the feasibility of performing direct device-to-device content distribution between these devices, while mobile, using commodity WiFi. This is needed because the content on the PNDs evolves over time, e.g. maps, road construction locations, speed camera locations, points of interest (POI), etc. Also, new content is introduced by content providers over time, such as tourist guides, voice packs and rich POI files.

Current PND models have several gigabytes of storage, and this is increasing each year. The maps are usually around a gigabyte in size, and the remaining space is used to store other content, which is creating a trend towards larger richer content. Limited satellite imagery is now available on some PNDs. Point-of-interest (POI) files are evolving from a simple longitude latitude coordinate and short textual description, and are beginning to include images, longer descriptions, reviews, etc. For example, a POI file representing the UK historic monuments and buildings run by a preservation charity would contain approximately 800 entries, and as a text POI would require less than 10KB. However, including opening times, general description and prices, and a single *small* image, requires approximately 35MB.

Manufacturers already use a number of techniques to update the content on deployed PNDs. Early models required the PND to synchronize through an Internet connected PC. However, owners infrequently synchronized so many current high-end models include some form of wireless connectivity, either built-in or available as an optional extra, allowing the content on the PND to be updated during use. For example, some TomTom models use Bluetooth to connect the PND to a mobile phone, and others use dedicated dongles connected to the PND. These allow the PND to use the cellular network to retrieve content updates. Others use FM-based wireless broadcast channels.

We describe and evaluate Broadside, a system that uses commodity WiFi to enable vehicle mounted PNDs to exchange files with other PNDs that they encounter. To ensure that Broadside is efficient, it is designed to exploit even short connectivity periods. Broadside uses a novel version-aware content discovery protocol that enables two PNDs to determine which files they need to exchange, and a novel and efficient data transfer protocol using a bit-vector acknowledgements. The content discovery protocol is designed to *incrementally* discover files with low overhead rather than discovering *all* files to transfer at once. The transfer protocol is optimized by using a vector-based acknowledgement scheme which reduces the number of acknowledgement pack-

CoNEXT'09, December 1–4, 2009, Rome, Italy.

ets from $O(P)$ to $O(logP)$, where P is the number of packets being transmitted.

We present performance measurements from three different environments; two urban and one high speed motorway. We demonstrate the feasibility of Broadside and also quantify the impact of using 802.11a versus 802.11g, using IP versus no IP, the impact of using externally and internally mounted antennae, and the impact of varying the packet size. The results show Broadside supports efficient file transfer.

Section 2 surveys how content is currently disseminated to PNDs. Section 3 describes Broadside, Section 5 presents the Broadside evaluation. Section 6 places the work in context to related work and, finally, Section 7 concludes.

2. BACKGROUND

For many PNDs, updating the content requires a user to attach the PND, by USB cable, to an Internet connected PC. However, users do this infrequently, and manufacturers have been exploring alternative content distribution mechanisms: predominantly using the cellular infrastructure or broadcast channels (either FM-radio or satellite based).

Exploiting cellular infrastructure for content distribution appears attractive but has several cost disadvantages. First, the increase in cost of manufacture for adding a cellular chipset is currently approximately 5-10 times higher than adding a Bluetooth or WiFi chipset. PNDs are commodity hardware devices, and a rough rule of thumb in the industry is that a $1 increase in manufacturing costs translates into a $3 increase at retail. Most PNDs retail for under $200, and the cost of adding a cellular chipset increases this by nearly 50%. Even if the consumer covers the extra hardware costs, the PND manufacturer (or content provider) needs to cover the cost of transferring the data over the cellular network. Users or content providers may be willing to pay for the small amount of bandwidth required for real-time search results, but it is not cost effective to distribute content of 10-100's MBs. Current rates per MB are significant enough that distributing even relatively small files (on the order of a few MB) can mean the bandwidth costs are higher than the value of the content. Secondly, per-country organizations control access to the cellular infrastructure, requiring manufacturers to negotiate per-country for access and potentially requiring specific hardware for each region. Furthermore, if the device is used in another country or with another operator, the content provider can end up paying effectively roaming data transfer rates. Thus, manufacturers tend to sell dedicated dongles only for certain countries, where the number of units sold is expected to be large, for example TomTom HD Traffic which is available in larger European markets.

These issues have caused some manufacturers to use Bluetooth to pair the PND to a user's mobile phone and use it as modem. By exploiting the user's existing hardware and relationship with a cellular provider, this overcomes many of the issues from a manufacturer's perspective. However, this transfers the cost issue to the user. Each user's charging plan differs with provider and country, and naive users can end up with large bills simply because they did not understand the size of the content being retrieved. Furthermore, there are often restrictions, for example, US AT&T customers using an unlimited data plan with their iPhone cannot use it as a modem. Indeed, some PNDs highlight these issues in their manuals, and allow users control over what is sent over the cellular network and limit the content transfer to small items.

The challenge of using cellular has meant other wireless technologies have been deployed, in particular broadcast based distribution. This usually uses FM radio channels, and has been widely deployed for real-time low bit-rate traffic and other information in many countries, e.g. RDS-TMC (worldwide) and MSN Direct (USA). MSN Direct provides both real-time, such as weather and traffic, and longer term content, such as gas prices and weekly movie times. In general, broadcast based solutions must frequently retransmit information to ensure that all PNDs eventually receive the content. There is no feedback, either explicit or implicit, about when a particular file has reached all PNDs. FM-based solutions can only provide low bit rates and are limited to areas where licenses can be obtained and infrastructure installed. In order to address the coverage and bandwidth issues, satellite distribution is beginning to be deployed. For example in the USA, there is Sirius Traffic and XM NavTraffic. Again, these are continent or region specific, and there is a significant cost associated with both the chipsets and deploying the infrastructure, that needs to be recouped.

Given the issues with the currently used technologies, distributing content using commodity WiFi, which uses unlicensed spectrum in most of the world, is attractive. WiFi-based content distribution could be achieved using a set of road-side hotspots, or by exploiting unsecured hotspots [6]. But deploying infrastructure is expensive, the opportunistic use of unsecured hotspots is illegal in many regions, and in non-urban areas, e.g. freeways, there may be no WiFi hotspots at all. In many countries, ISPs now provide WiFi access points for their subscribers with security enabled to avoid installation errors and to prevent freeloading.

We believe a more feasible approach could be to use device-to-device content distribution. While the focus of the paper is on the challenges of performing the device-to-device transfer, it is important to understand how to seed content into the system, and how fast the information will propagate.

Seeding The seeding process can take place in many ways. PNDs could be configured to connect to the user's home wireless network, whenever in range, and pull content from a specific web site. Therefore, when some users get into their vehicles, they will be in range of the home AP and can pull content [6]. Many users will either not configure their devices to work with their home networks, or park their vehicles in a location where it can never connect to the home network. These users will rely on the content propagating through the device-to-device network. An alternative approach would be to provide a small number of fixed or mobile access points, or infostations, that can provide content to the network.

3. BROADSIDE SYSTEM OVERVIEW

We assume that each PND is equipped with 802.11 WiFi, and when two devices come into communication range, they synchronize using Broadside. During the synchronization, if the two PNDs have common files, but one has a newer version of a file, then the file is transferred between the devices. At the end of a full synchronization, we would like, with high probability, each shared file to be of the same (latest) version on both devices.

In general, we assume that there is set of files, F, that

represents the universe of all files. Each file $f_i \in F$ has an associated signed *file certificate*, c_i, which contains a 160-bit SHA1 hash of the file, a 16-bit version number and a unique 160-bit *file identifier* (fileId). We use a Public Key Infrastructure (PKI) with an *offline* Certification Authority (CA). The CA generates signed public/private key pairs for content providers, and every PND has the public keys for the CA and all the content providers. Public keys for content providers can be distributed as files using Broadside. Each file f_i is associated with a single content provider, or owner, and only the owner can generate new file versions. Updated files require a new file certificate to be generated containing the new file hash and incremented version number. Given two file certificates containing the same fileId, the one containing the highest version number is considered the latest version. The fileId is generated by taking the SHA1 hash of the content provider's public key concatenated with a textual name for the file generated by the content provider. For each file, this is also an associated *instanceId*, which is a unique 160-bit number generated by taking the SHA1 hash of the file certificate, c_i. The file certificate includes the fileId, which is independent of the version and content of the file. In contrast, the instanceId *is* dependent on the version and content.

Each PND, A, is assumed to contain a set of files, d_A, where $d_A \subset F$. The membership of d_A can be changed dynamically, for example by applications running on A. It should be noted that applications can distribute file and configuration information in files distributed by Broadside.

4. BROADSIDE

Our prototype Broadside implementation uses a standard Atheros 802.11a/b/g chipset and firmware. We used the Atheros reference driver, modified to send a network join probe every 100ms, instead of once per second, and to overcome two issues: first to prevent premature network disconnect on a busy link when beacons were lost but unicast packets were being successfully received; second to reset the card on network disconnection to clear an occasional but persistent fault where the card would transmit for only the first 60ms of every 100ms interval. We set the 802.11 configuration parameters to use ad-hoc mode, a static channel, SSID and BSSID, to disable power management features and to set the disconnection timeout to two seconds [19, 9, 14].

Prior work has highlighted overheads when using IP: *(i)* IP address assignment and *(ii)* IP duplicate address detection and *(iii)* IP address resolution and *(iv)* protocol timeouts in the order of seconds. This motivated us to compare two implementations of Broadside. Both versions use the same content discovery algorithms, but *Broadside-IP* uses IP packets and relies upon TCP retransmissions, where *Broadside-Raw* uses raw Ethernet packets and the data transfer protocol described below.

Broadside-IP uses the standard unmodified Windows XP SP2 TCP/IP stack with a statically assigned IP address and an empty ARP cache. It runs the content discovery algorithm over TCP, and the files are also transferred using TCP. There is a 1500 byte MTU imposed by the IP stack. In Windows XP, a failed TCP connection attempt is retried after 3 and 9 seconds, and abandoned after 21 seconds. During early testing, we observed that this meant that sometimes during a run no data was transferred as the TCP connection could not be established within the connection window. Therefore, in Broadside-IP, we explicitly restart TCP connections after just one second in order to circumvent these timeouts.

Broadside-Raw exchanges 2304-byte packets directly with the 802.11 driver, using the wireless MAC address as the address of the device. Each packet has a standard Ethernet header with a special Ethernet packet type. The 802.11 and Ethernet headers are fixed size, so the overhead per data bit drops as the packet size increases, but transmit time increases with packet size which can lead to higher packet loss. Note that a device running Broadside-Raw can also contain a TCP/IP stack.

4.1 Content Discovery

Once link-level connectivity has been established between two PNDs they need to identify one or more files to transfer. To enable this, each PND periodically broadcasts an application-level beacon including the number of files it currently has and a hash of the content. The content hash is generated by hashing the list of instanceIds for each file stored on the PND, ordered by fileId. Whenever a PND synchronizes with another PND, it caches the associated content hash, and when it receives a beacon from any device, it checks the cache to see if it has already synchronized with a device advertising the same content hash. If not, the two PNDs synchronize.

Synchronization requires the PNDs to determine the set of files that are in common, but with different version numbers. Intuitively the fileId, which is version independent, is used to discover common files, and instanceId, which is version dependent, is used to discover different versions. Connection windows are often short, so the aim is to determine these files incrementally with few messages: as soon as a file is identified it can be transferred. The total number of messages required, as well as the expected number of messages before a common file with different version is found, is a function of the number of files stored on the device. Therefore, the PND with the lowest number of files initiates the synchronization.

The synchronization is achieved by using Bloom filters [5]. Previous work focuses on using Bloom filters for simply determining whether files are in common, and in contrast we also need to take the version number into account. Bloom filters encode set membership efficiently in a $k-$bit array where, for each *item* in a set, the $hash_j(item)$ *mod* k bit is set, for all hash functions in a given set of j hash functions. To check if an item is present in the set, each $hash_j(item)$ *mod* k bit is checked, and if they are all set, the item is considered a member of the originally encoded set. Bloom filters cannot yield false negatives, but they do yield false positives.

The initiating PND partitions its set of files into groups of g files randomly. It then selects one group at random and encodes all the fileIds in a Bloom filter, called the *file filter*, and all the instanceIds in second Bloom filter called the *version filter*. The length of each bloom filter is 50% of the available packet payload, and when using the full 802.11 frame size the length of each bloom filter is 9144 bits. There is a trade-off between the number of items that can be encoded in a Bloom filter and the probability of false positives. Based on heuristics [21], we use $g = 500$ and 12 hash functions, yielding a false positive rate of approximately 0.0154%. These bloom filters are then sent to the responding PND.

When a set of bloom filters is received, the device checks if each fileId it has locally is encoded in the file filter. If the fileId is a detected in the file filter, then it checks if the associated local instanceId is encoded in the instance filter. If it is in both then, with high probability, the file is common to both PNDs but the version is the same. Otherwise, if the fileId is detected but the instanceId is not, then with high probability, the file is common but they have different versions, and this file is a candidate file for transfer. After all fileIds on the device have been checked, the responding PND will contain a list of candidate files for transfer.

The next stage is that the responding PND generates a single packet-length bloom filter, called the *match filter*, encoding the fileIds of the candidate files for transfer. The match filter is then returned to the initiating PND, which checks which of the fileIds from the g files it encoded in the file filter are present in the match filter. For each one found, the fileId and version information, encoded in 22 bytes and aggregated into the smallest number of packets, is then sent back. Finally, responding PND uses this explicit file and version information can be used to determine whether to push or pull files to or from the initiator. Once all file exchanges have been completed, the initiating PND sends the file and version filter for the next group. The process is repeated until all the groups on the initiator have been processed.

False positives for the Bloom filters can impact the synchronization. False positives on the file and match filters simply can lead to more data being passed between the devices than is strictly necessary. In particular, false positives on the file filter leads to redundant fileIds being encoded in the match filter, and false positives on the match filter can lead to redundant file records being transferred. The false positive rates are configured to be low, so the overhead this introduces is minimal. However, false positives on the version filter can lead to a PND believing that both PNDs have the same version of a file when they do not. Therefore, at the end of a full synchronization, two PNDs may not have detected all files to transfer. The probability of this occurring is approximately 0.000154 for each file, which is low. Furthermore, randomly selecting the g files in each group means, when another device is found that contains the same missed file, it will be successfully discovered and transferred.

4.2 Bulk transfer protocol

TCP's performance is suboptimal for bulk data transfer in short-lived lossy connections in vehicular networks [13]. In Broadside-IP, we transfer files using TCP, but for Broadside-Raw we use a more efficient bulk transfer protocol with reduced overhead. There are two properties that we exploit in the transfer protocol: the source has access to the entire file before transfer begins, and when receiving the file, the order in which the file content is received is not important.

We use a bit-vector acknowledgment-based scheme. Each file is split into p packets, with each packet containing a 2-byte block identifier to identify the offset within the file. The source maintains a p-bit *send vector*, where each bit represents a packet and is initialized to false. The source continuously iterates through the packets and, if the associated bit in the send vector is false, the packets it transmitted. The destination maintains a p-bit *receive vector*. Whenever a packet is received the associated bit in the receive vector is set. The destination sends the receive vector to the source, which then uses the received vector as its send vector.

The destination chooses when to send the receive vector to the sender based on the number of bits set in it. The destination maintains a count of the number of set bits, b_{set}, and the number of bits set the last time the receive vector was sent to the source, b_{sent}. Whenever $b_{set} - b_{sent} \leq (p - b_{sent})/2$ the receive vector is sent. If a duplicate packet is received this also triggers the sending of the receive vector. Intuitively, this increases the number of ack packets as more of the file packets are transferred. Hence, in the common case this will generate $O(log\ p)$ acknowledgement packets. If an ack packet is lost, the next ack packet sent also includes all the information. So, for a 1 MB file with a 30% loss rate, the expected the number of acks sent is less than 1% of the total packets sent. Furthermore, if a file transfer is interrupted and needs to be resumed with either the same or a different device, the receiver can provide the receive vector and the transfer process can resume without duplicate packets being sent.

The limitation is that the bit vector needs to be transferred between the destination and source. However, a packet size of 1500 bytes supports a bit vector that is able to transfer a file of over 17MB. Given the transfer rates achieved by Broadside the practical limit on each file size is a few MB. Larger files can be decomposed into multiple smaller files, and transferred independently. We also considered using coding schemes, e.g. digital fountain [8]. These have a low computational overhead but incur a higher transmission overhead. More efficient coding schemes, e.g. networking coding [1, 12], are too computationally expensive [17] to be used in practical systems for contemporary PNDs.

5. EVALUATION

We examine the base performance of Broadside over two different transport protocols: Broadside-Raw using raw packets, and Broadside-IP using IP packets. We also examine the performance impact of using 802.11a versus 802.11g, the impact of locating the antenna inside or outside the vehicle, and the impact of varying the packet size.

The experiments use PCs running Windows XP/SP2, each equipped with a commercial 802.11 card based on the Atheros 802.11a/b/g chipset and a GPS receiver. The experiments were performed in three locations in Cambridge, UK, chosen to represent different environments. In two locations, we used one moving vehicle and one stationary vehicle and in the third we used two moving vehicles.

Figure 1 shows aerial views of the locations, indicating the position of the stationary vehicles and the route of the moving vehicles.

The *terraced* location is in the center of town, in an area densely populated with two-storey brick terraced housing adjoining the sidewalk. Vehicles are parked along one side of the street, and a one-way traffic system is enforced. The stationary vehicle was parked at the intersection of an adjoining side-street. The moving vehicle achieved a speed of approximately 15 miles per hour at closest approach.

The *residential* location is in the same town and is in a suburban area consisting of lower density two-storey brick housing. The houses have front gardens that separate the houses from the street. Vehicles are intermittently parked on both sides of the street. The stationary vehicle was parked on the side of the street. The moving vehicle achieved a speed of approximately 25 miles per hour.

The *motorway* location was a stretch of highway just out-

(a) Terraced

(b) Residential

(c) Motorway

Figure 1: Locations of our experiments in Cambridge, UK (not to scale).

side the town. We used two moving vehicles travelling at 70 miles per hour, passing each other from opposite directions, yielding a closing speed of 140 miles per hour. The two cars were both driving in the lanes closest to the central reservation, which separates the two sides of the motorway.

To minimize the impact of variance in weather and changes in the environment at each location, all experiments at a particular location were completed on the same day. Each experiment was run five times. A few runs were obstructed by other vehicles or street conditions and we excluded those runs. These runs performed better due to increased connection durations and lower vehicular speed, so the results we present are therefore conservative. Unless otherwise stated, all results shown are the mean of the 5 runs, with error bars showing the maximum and minimum values.

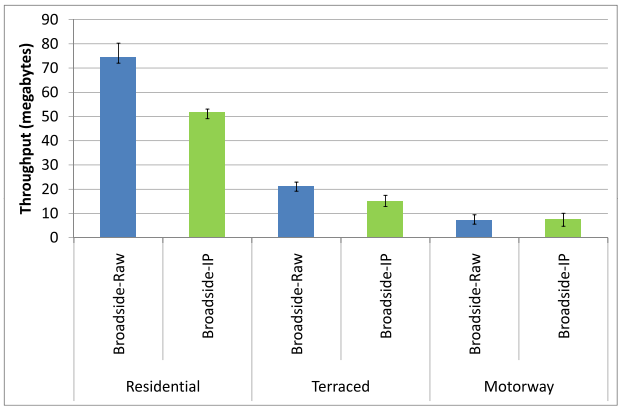

Figure 2: Throughput (base configuration).

The *base configuration* used in our experiments was a Netgear WAG311 PCI card configured to use 802.11g (channel 1) and an external omni-directional antenna with 5dB gain. The antenna was mounted on the dashboard in a similar location to where a PND device would be mounted. When running experiments in the terraced and residential locations, we recorded the number of access points using the 2.4GHz band: we observed approximately 90 and 7, respectively, split roughly equally across 3 channels (1, 6 and 12).

In the experiments each laptop was configured with 50 common 10MB files in its file set, but with different versions. In order to aid in repeatability of the experiments, one laptop acted as the data source throughout, starting with a later version of all files. In the two locations with a stationary vehicle, the moving vehicle contained the data source.

The evaluation uses three base metrics: *connection duration*, *throughput* and *startup delay*. When two devices come into wireless range, they establish a link, and when they go out of range, the link breaks. We define *connection duration* as the time for which both nodes consider an active link to exist between them. We define *throughput* as the amount of file content transferred during a single connection. Finally, we define the *startup delay* as the time between when the link is established and the transmission of the first data packet belonging to a file.

5.1 Transport protocols

To understand the impact of the transport protocol, we compare the performance of Broadside-IP and Broadside-Raw. Figure 2 shows the mean throughput achieved with the base configuration by location, with max-min error bars, for both Broadside-Raw and Broadside-IP. Broadside-Raw yields higher throughput than Broadside-IP at the residential and terraced locations, achieving 44% higher throughput on average in the residential and 40% higher on average in the terraced. The most challenging scenario is the motorway where the throughput is approximately the same for both configurations.

In order to understand the differences between the three locations further, we show the mean connection durations in Figure 3 for each location. As would be expected, the connection durations are independent of whether Broadside-IP or Broadside-Raw is used, as the physical layer connectivity

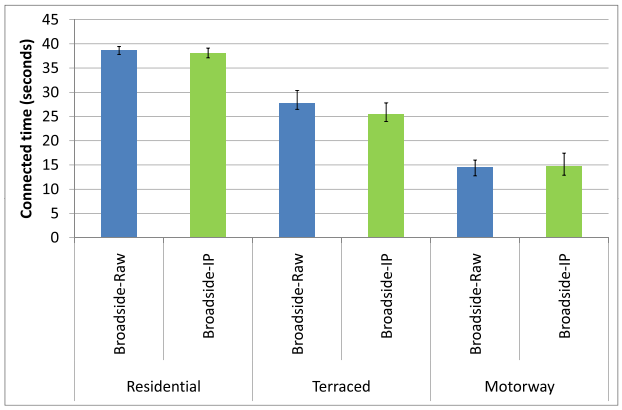

Figure 3: Connection duration (base configuration).

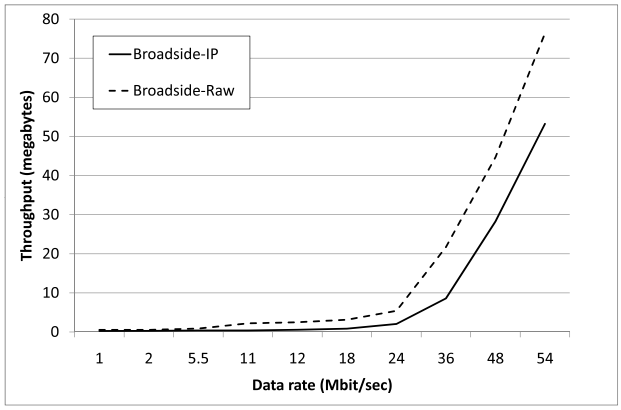

Figure 4: CDF of throughput by data rate (residential).

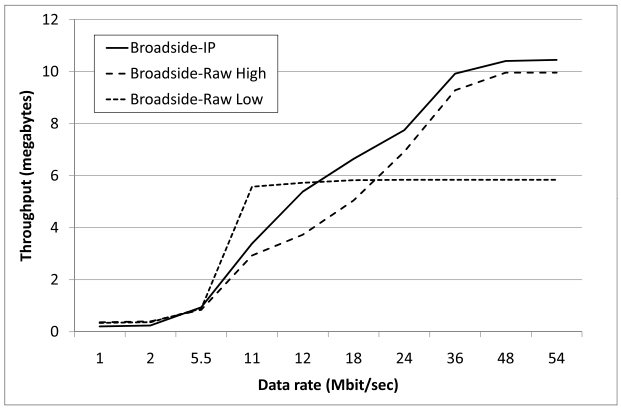

Figure 5: CDF of throughput by data rate (motorway).

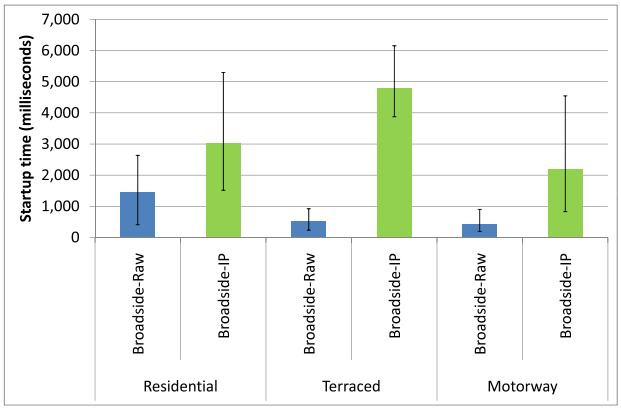

Figure 6: Startup delay with base configuration.

does not depend on the higher layer protocol. The maximum and minimum error bars are tight around the mean, indicating that the connection duration was of similar length per run. We also confirmed that the vehicle speed was consistent across runs. The highest variance is observed for the motorway, where repeatability was the hardest to achieve.

In Figure 2, we can see that there is approximately an order of magnitude difference in the throughput achieved when comparing residential to motorway for Broadside-Raw and approximately a factor of 6 for Broadside-IP. From Figure 3, we can see that only part of this difference is explained by the shorter connection times for the motorway, with only approximately a factor of three difference in mean connection duration between the residential and motorway locations.

To understand the difference in throughput, we need to examine the throughput versus data rate for each location. Figure 4 shows the CDF of throughput versus 802.11g data rate for the mean run for both Broadside-IP and Broadside-Raw for the residential location. The two versions of Broadside transfer the majority of the data at the highest two data rates 802.11g supports. In contrast, Figure 5 shows the CDF of throughput versus 802.11g data rate for a number of runs on the motorway. Examining the motorway runs in detail shows that Broadside-Raw delivers on average 7.1MB, but two runs performed noticeably worse. Figure 5 shows the mean run for Broadside-IP and two runs for Broadside-Raw. Broadside-Raw Low is one of the two runs which performed noticeably worse and Broadside-Raw High is a run which achieved better throughput. It is clear that the Broadside-Raw High and Broadside-IP runs achieve better throughput by utilizing the higher data rates with approximately 40% of the data being delivered at data rates above 11 Mbps.

We conclude that there are two main differences between the locations. The first is the connection duration, and the second is the amount of data transferred at higher data rates. In particular on the motorway, higher packet loss rates caused lower data rates to dominate compared to the other locations.

So far, we have examined the difference in performance between the locations. Figure 2 also shows, in some locations, a significant difference in the performance between the two Broadside versions. Recall that the two use different packet sizes, with Broadside-Raw using 2304 bytes and Broadside-IP using 1500 bytes and the TCP/IP stack to transfer files while Broadside-Raw uses raw packets and the bit-vector based acknowledgement protocol described.

We now examine the effects of each of these differences between the two Broadside versions. First, we consider the startup delay caused by using IP. Figure 6 shows the startup delay, which is the time from when the link-level connection is established to when the first file data packet is transmitted. It shows that, across all three locations, in the majority of cases Broadside-Raw delivers the first packet in under one second. The maximum startup delays for Broadside-Raw are

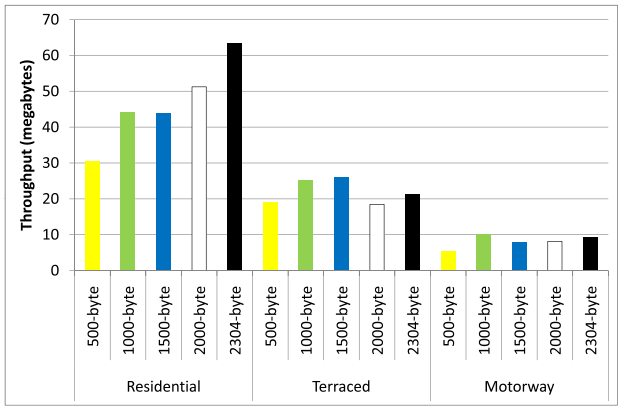

Figure 7: Impact of packet size on throughput.

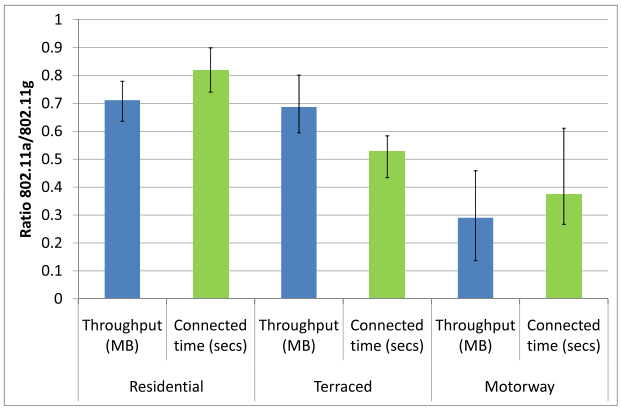

Figure 8: Throughput and connection duration when using 802.11a normalized by the results for 802.11g.

observed at the residential location, where two runs experienced a particular set of multiple packet losses during their content discovery phase that required them to restart the content discovery protocol, after a one second delay.

Figure 6 shows that, across all locations, the ratio between startup time and connection duration is at least a factor of two greater for Broadside-IP than for Broadside-Raw. In general, the Broadside-IP startup delay is dependent on location while for Broadside-Raw it is independent. This is because Broadside-IP with its dependence on the TCP/IP stack is more impacted than Broadside-Raw by a series of short-lived intermittent connections at the start [19, 2]. Interestingly, these are most frequent at the terraced location and rare on the motorway. The effect is compounded as the two nodes can *independently* disconnect and reconnect. This means that one node can disconnect and then reconnect transparently to the second node.

There are a number of overheads that contribute the higher startup delay for Broadside-IP. First, when the stack is initialized, it performs a gratuitous ARP for the local IP address for duplicate address detection. We ran some experiments and determined that the IP stack waits about 100ms for a response to this gratuitous ARP, during which time it transmits no packets. In order to communicate with the other node, the IP stack needs to perform an ARP request. Windows XP does *not* retransmit failed ARP requests, and instead keeps invalid (unresolved) ARP entries in the ARP cache for approximately 3.5 seconds. A new ARP request is only sent for IP addresses for which *no entry* is present in the ARP cache. Therefore, if the ARP request or ARP response is lost, then this can result in a significant delay. Optimizing the TCP connection to restart after one second ensures that soon after an invalid ARP entry is timed out, another ARP will be generated. However, this or the response to it can also be lost, which can yield even longer startup delays.

While the startup delay clearly impacts the throughput, it does not fully explain the difference between throughput achieved at each location for the two versions of Broadside.

5.2 Impact of packet size and ACK overhead

In order to understand the impact of varying the packet size, we ran a set of dedicated experiments. The intuition behind increasing the packet size is that the 802.11 and Ethernet headers are of fixed size, and hence increasing the packet size reduces the per payload bit overhead, thereby increasing throughput. However, if there is a constant bit error rate, then increasing the packet size will increase the packet loss rate. This can be detrimental, not just because larger packets take longer to transmit, but also because an increase in loss rate may cause the data rate selection algorithm to lower the data rate. The experiment removed as many overheads as possible; one node transmitted packets of a specific size at line rate, saturating the link, whenever the link was connected. Using the base hardware configuration, we measured the throughput obtained at the three locations with packet sizes of 500, 1000, 1500, 2000 and 2304 bytes. We ran each experiment four times, and Figure 7 shows the throughput for the median run for each packet size at the three locations.

From Figure 7, we observe that no single packet size is ideally suited to all locations, and indeed the highest throughput is achieved by a different packet size at each location. However, at all locations, the largest packet size achieves reasonable performance. So, the larger packet size accounts for a significant proportion of the gain that we see in the residential setting when comparing Broadside-IP to Broadside-Raw in Figure 2.

The final difference between the two versions of Broadside is the use of TCP connections with acks versus the bit-vector ack protocol for file transfer. The mean number of acknowledgement bytes sent by Broadside-Raw was only 0.087 MB compared to 1.66 MB for Broadside-IP, which is 0.1% and 3.2% of their throughput.

5.3 Impact of changing frequency band

In the next set of experiments, we examine the impact of changing from 802.11g (2.4GHz) to 802.11a (5.4GHz) on the performance of Broadside-Raw. We do this as a new 802.11p MAC has been proposed for inter-vehicular networking which is based on the 802.11a MAC at 5.9 GHz. As we were unable to obtain any 802.11p chipsets, comparing 802.11a to 802.11g is the closest we can get to understanding the impact of using 802.11p. For these experiments, we used a Netgear WAG511 PCMCIA card with a PCB mounted antenna configured to use 802.11a on channel 40. In all the three locations, no access points operating in the 802.11a 5.4GHz band were observed. Figure 8 shows both the throughput and the connected duration when us-

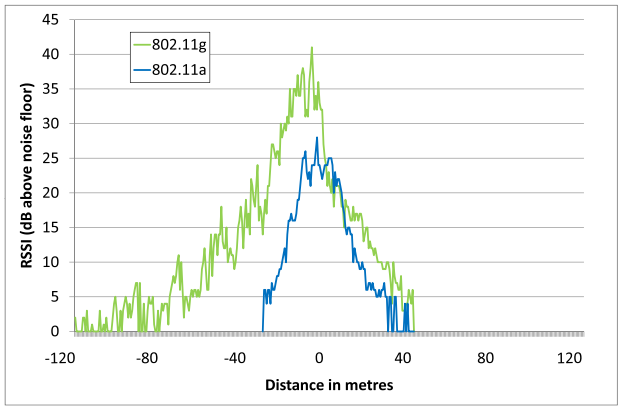

Figure 9: RSSI against distance for 802.11a and 802.11g (terraced).

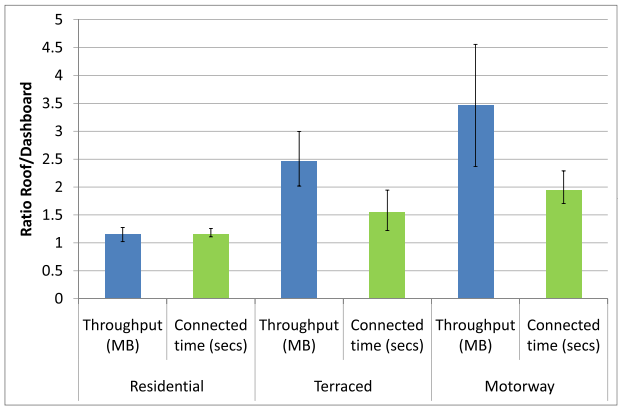

Figure 10: Effect of antenna placement.

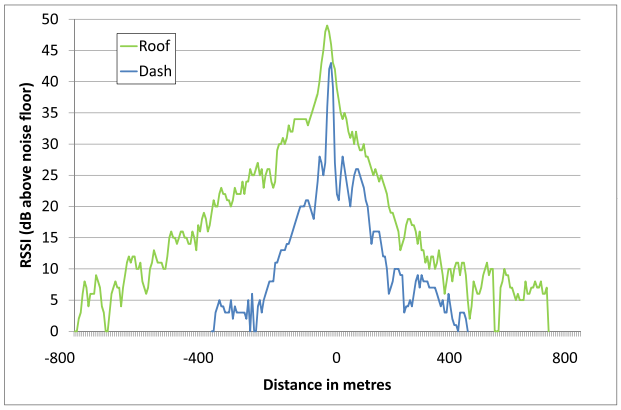

Figure 11: Effect of antenna placement on RSSI (motorway).

ing 802.11a, normalized to 802.11g using the base configuration, at all three locations. For all the locations, the 802.11a throughput is lower than the 802.11g throughput. Intuitively, the signal propagation should be lower, which results in shorter connection durations for 802.11a. To demonstrate the impact this has, Figure 9 shows the RSSI versus distance for both 802.11a and 802.11g for the mean runs at the terraced location. The connection duration is shorter, with 802.11g and 802.11a forming a connection at 121 and 35 meters, respectively. Also, 802.11g has a significantly higher RSSI during the connection period. The increased RSSI has two effects that increase throughput. First, it results in longer connection durations. Second, the rate adaptation algorithm is able to exploit higher data rates for longer, and therefore to obtain higher throughput.

The choice of antenna and wireless card had a major impact on performance. We used two cards: the Netgear WAG311 PCI card with an external omni-directional antenna tuned to the appropriate frequency band, and the Netgear WAG511 PCMCIA card with integrated PCB antennae. The PCMCIA card on 802.11g delivered an order of magnitude lower throughput at the residential location, and collapsed altogether for Broadside-IP on the motorway, compared to the base configuration. For 802.11a, the PCMCIA card outperformed the PCI card by a factor of two. It is clear that selecting the correct antenna and location is important.

5.4 Antenna location impact

The final set of experiments examined the impact of mounting the antenna internally versus externally to the vehicle, using the 802.11g base configuration. Most prior work has used a roof mounted antenna. Recall that, in the base configuration, we use an antenna mounted on top of the dashboard, at a similar location to where a PND mounted antenna would be located. For the externally mounted antenna, we placed it on the roof of the vehicle.

Figure 10 shows both the throughput and connection time of the roof mounted antenna, normalized to that of the internally mounted antenna. Interestingly, the effect of mounting the antenna externally differs between the residential and the other two locations. In general, mounting the antenna externally increases the RSSI. Figure 11 shows the RSSI versus distance between the two vehicles for the closest to mean runs for the motorway. As described in Section 5.3, the increase in RSSI leads to an increased throughput. These effects are clear for the terraced and motorway locations, where there is a significant increase in throughput partly accounted for by the increase in duration time and also by the amount of data transferred at higher data rates.

From Figure 10, the residential location does not show a significant difference between the internal and external antenna. We examined this in detail. First, at this location, building occlusions at the limits of connectivity result in only a small increase in connection time. Further, with the internal antenna we observe that the highest data rates are already used for a significant proportion of the connection duration, as was shown in Figure 4. Hence, the benefit of the higher RSSI does not enable the use of higher data rates in this location, as they are already used. The combination of these two factors means there is little impact on the throughput.

5.5 Content discovery evaluation

We compare the performance of the proposed discovery algorithm against the set reconciliation algorithm in [20], which has been proven to provide near optimal communication overhead. Set reconciliation algorithms ensure that two devices, after synchronization, have the *same set* of files. In contrast, in Broadside each device has an arbitrary set of files, and the content discovery algorithm identifies the com-

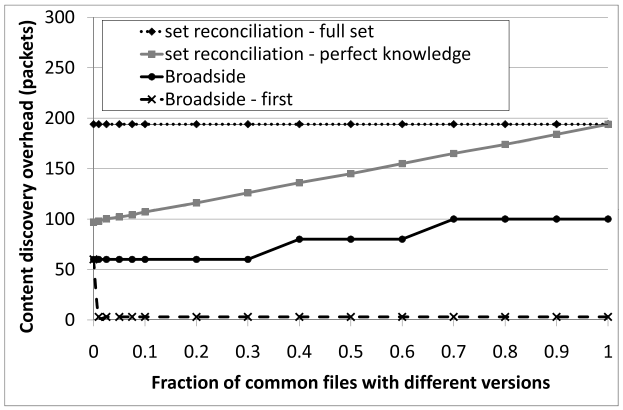

Figure 12: Content discovery overhead in terms of transmitted packets.

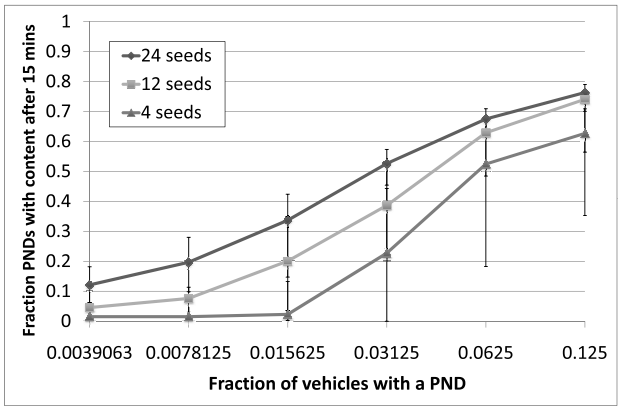

Figure 13: Fraction of PNDs with content after 15 minutes versus fraction of vehicles with PND.

mon files with different versions. Further, the files are found incrementally, and with high probability the first is identified after only a few round trips. This is a more complex problem than simple set reconciliation.

We provide a brief overview of the set reconciliation algorithm from [20], and in particular describe how we adapt it to work with version numbers. Conceptually, nodes represent their local sets as *characteristic polynomials* whose zeroes represent the elements in the local set. The ratio between two characteristic polynomials then cancels out identical entries. Two nodes exchange polynomial evaluations and locally reconstruct the polynomials through interpolation and factoring. To support versions, we concatenate fileIds with version numbers, which allows nodes to locally determine common files with differing versions. In [20], it is shown that a minimum number distinct evaluations need to be exchanged. We compare against two variants: *full set* and *perfect knowledge*. Full set represents the upper bound of one evaluation per file. Perfect knowledge represents the theoretical overhead if two devices knew the exact number beforehand.

We consider a universe of 20,000 files with each device randomly selecting 10,000 files to store. Thus, when PND A and B synchronize, on average they have 5,000 common files. We vary the fraction of common files with different version numbers from 0 to 5,000. Figure 12 shows the packet overhead for A and B to determine the set of files to transfer. Broadside generates approximately half the overhead of *perfect knowledge* because *perfect knowledge* needs to send an evaluation for *each* file difference, including those not in common. For Broadside, the overhead depends on the number of files present on A and the fraction of common files with different versions. With $g = 500$, 20 packets cover all 10,000 files, each with a match filter response. Broadside also requires a message per round containing the possible matching file records. The visible steps in Broadside occur where the number of matching file records overflows the capacity of a packet. As the number of files on A and B is constant, the overhead for *full set* is also constant, whereas for *perfect knowledge*, the packet overhead increases as a function of the number of differing entries.

Figure 12 also shows *Broadside first*, which represents the mean number of packets expected to be sent to find a file for transfer. When 500 of the 5,000 common files have different local versions, on average 25 files for transfer will be found with only 3 packets. Provided 20 or more common files with different versions exist, on average, a file to transfer will be discovered after 3 packets. In the worst-case, when all common files have the same version and so no files need transferring, 60 packets are required.

Finally, the set reconciliation algorithms evaluated will detect all differing files. Broadside, on the other hand, can fail to detect a differing file, with probability 0.000154 per file. Randomly permuting the group membership means that any missed files should be detected subsequently with high probability.

5.6 Epidemic distribution evaluation

Properties of epidemic distribution in vehicular networks have been studied elsewhere [18]. This paper focuses on the systems aspects of the device-to-device protocols used to enable content transfer between devices, and how these perform in the real world. However, for completeness, we also provide some brief results looking at the impact of PND density on the epidemic distribution of content. We ran an experiment using a mobility trace generated by Los Alamos National Laboratories (LANL) for the city of Portland, Oregon. It was generated using TRANSIMS [24, 4] and is believed to be realistic of the traffic flow in downtown Portland. The traces incorporate per-vehicle activity flows, generated from census and other information to increase the accuracy of the trace. The trace covers 15 minutes from 8:00AM on a weekday and covers a 3km by 7km area of downtown Portland straddling the Willamette River, covering both urban streets and several freeways. The trace provides position information for 16,529 vehicles in total, e.g. cars, trucks and buses, and is updated each second. The average velocity for moving vehicles is 54km/h.

We vary the fraction of vehicles carrying a PND and randomly select a further set of vehicles to act as mobile seeds carrying a PND with the seeded content. Each seed is preloaded with a file required by all devices. To transfer the content between two PNDs, the two devices must be within 150 meters for 20 seconds. We assume a per-second transfer rate, meaning if a device is in contact with another PND for k seconds then $k/20$th of the content is transferred. A PND can store partially transferred content, and can resume transfer when another PND with the content is encountered.

Conservatively, the PNDs *do not* become content sources until they have the full content. A contact duration of 20 seconds at less than a 150 meters, which is consistent with the results shown earlier, should enable us to transfer approximately 20MB. We ran the 15 minute trace and then determined the fraction of PNDs with the full content, excluding the original seeds. The vehicles carrying the seed PNDs are selected randomly, and are not guaranteed to be present in the trace for the full 15 minutes. In fact, the majority of vehicles are present for less than the full 15 minutes.

Figure 13[1] shows the fraction of PNDs with the content versus the fraction of enabled vehicles carrying a PND (excluding seed PNDs). We varied the penetration rate from 0.39% to 12.5% based on the observation that in Western Europe 1 in 4 vehicle owners currently have a PND device, and one manufacturer has approximately 50% market share. We ran the experiment with 4, 12 and 24 seeds, which would be conservative if the seed vehicles were buses or taxis. Each experiment was run 20 times and the median, with 5th and 95th percentile error bars, is shown. The results demonstrate, even over just 15 minutes, at low penetration rates and with a small number of seed devices, that a significant fraction of the PNDs can receive the content. This makes us believe, having demonstrated the feasibility of device-to-device transfer over WiFi, that it could be used as the basis for a next generation delay tolerant content distribution system.

6. RELATED WORK

A number of studies have looked at the performance and characterization of wireless links in vehicular networks, including [10, 13, 19, 11, 23, 22]. Several of these studies have identified the need to design protocols optimized for vehicular networks. A number of small-scale vehicle-to-infrastructure testbeds [10, 19, 6, 26, 3, 2] have been used to investigate the feasibility of exploiting fixed access points (APs) to provide access to the Internet etc to mobile devices. Broadside is designed to support efficient device-to-device content transfer, either between a mobile device and a static access point or between moving devices.

Cabernet [10] is the most related work done concurrently with this work. It aims to exploit short-lived IP connections from moving vehicles to Internet hosts via open access points. Cabernet tunes IP and 802.11 timeouts to support IP over 802.11 infrastructure networks, and proposes the CTP transport protocol which hides network and address changes from applications. Cabernet uses a fixed data rate of 11Mbps and claims that there is no advantage from using higher data rates. The Broadside results show that data rates above 11Mbps, up to and including 54Mbps on 802.11a/g, do yield significant benefit when channel conditions permit. Broadside does not use IP and is not designed to support general Internet access via an AP: instead Broadside provides an efficient content discovery and file transfer protocol.

Prior work has characterized the connection durations as being short, with an entry, production and exit phase [13, 23]. We also see theses in the reported results, with higher loss rates in the entry and exit phases. Broadside attempts to minimize the time that it takes to discover the content to be transferred, allowing it to exploit the production phase to achieve maximum throughput. Unlike much prior work, Broadside has been especially designed to handle the link characteristics in vehicular networks.

There are proposals for 802.11p, based on 802.11a and 802.11e, aimed specifically at supporting vehicle networks in the licensed 5.9GHz band. The 802.11p MAC aims to reduce startup delays [9] and reduce the time taken to establish a link [14]. As 802.11p chipsets are not yet available, we used 802.11a to understand how Broadside works at frequencies similar to those used in 802.11p.

In overlays, algorithms have been proposed to identifying the set of blocks belonging to a file that two nodes have in common [7]. Our use of Bloom filters was inspired by this work [7] as has much of the work on using Bloom filters in wireless and distributed systems. However, Broadside addresses a more complex problem by incorporating version numbers for the file; determining which blocks two devices have in common does not use version numbers. If devices have a large number of files in common, but all with the same version number, simply detecting common files will create significant overhead.

Set reconciliation algorithms, e.g. [20], ensure that two devices, after synchronization, have the *same set* of files. In contrast, in Broadside each device has an arbitrary set of files, and the content discovery algorithm identifies the common files with different versions. Further, the files are found incrementally, and with high probability the first is identified after only a few round trips.

The Bayou [25] system provided a weak consistency model between replicated objects. Write operations could be performed opportunistically at each replica, and the reconciliation is performed to merge the updates. The approach is based on maintaining logs of write updates and then merging these logs. The Broadside content discovery efficiently identifies which file (object) is the most recent based on global version numbers, rather than attempting to perform reconciliation across diverged replicas.

Finally, there have been proposals for doing epidemic-based information distribution in vehicular networks, especially for small size content to support intelligent transport systems. Examining content distribution has predominantly been evaluated using simulation [15, 18], and the performance of these systems often differs dramatically between simulation and the real world. The most related work is CarTorrent [16], which is based on BitTorrent-like file dissemination. It is evaluated on a real testbed and uses multi-hop routing (AODV) with TCP to fetch file blocks from other vehicles with the content. In low density environments multi-hop routing will be difficult. Our work is complementary, and indeed, a CarTorrent-like application could be easily built on top of Broadside and would be able to exploit the performance Broadside achieves. CodeTorrent [18] investigates file sharing over a VANET using network coded content and promiscuous caching of overheard content. CodeTorrent could benefit from using Broadside to transfer coded blocks between devices, and CodeTorrent has only been evaluated in simulations.

7. CONCLUSION

We have described and evaluated Broadside, a practical device-to-device content transfer system. Broadside allows two devices, when they come into communication range, to

[1]Thanks to Giovanni Pau (UCLA) who provided the data for this chart.

identify common files with differing versions and then to efficiently transfer the latest version between the devices. Broadside is generic, and we have motivated and evaluated Broadside in the context of PNDs. The results show that Broadside is effective at exploiting short connection periods to transfer significant amounts of data. This is achieved through optimizing the link-level discovery, providing algorithms for efficiently identifying the content to be transferred and through the use of our own data transfer protocol that removes the need to use TCP/IP. The next generation of PND devices could use Broadside to enable an efficient delay-tolerant content distribution network.

Acknowledgements

We thank Giovanni Pau (UCLA) for providing the data in Figure 13. We would also like to thank the anonymous reviewers and our shepherd, Boon Thau Loo, for their feedback. We also thank Dinan Gunawardena, Ilias Leontiadis, Gustavo Marfia, Simon Schubert, Peter Wheeler and Georg Wittenburg for helping run the experiments. We would also like to thank Jim Turner.

8. REFERENCES

[1] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4), 2000.

[2] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. N. Levine, and J. Zahorjan. Interactive WiFi Connectivity For Moving Vehicles. In *Sigcomm'08*, Sept. 2008.

[3] N. Banerjee, M. D. Corner, D. Towsley, and B. N. Levine. Relays, Meshes, Base Stations: Enhancing Mobile Networks with Infrastructure. In *ACM MobiCom'08*.

[4] C. Barrett, R. Beckman, K. Berkbigler, K. Bisset, B. Bush, K. Campbell, S. Eubank, K. Henson, J. Hurford, D. Kubicek, et al. Transims: Transportation analysis simulation system. *Los Alamos National Laboratory*, 2002.

[5] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[6] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden. A measurement study of vehicular internet access using in situ Wi-Fi networks. In *ACM MobiCom'06*.

[7] J. W. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed content delivery across adaptive overlay networks. In *ACM Sigcomm*, 2002.

[8] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *ACM SIGCOMM*, 1998.

[9] S. Eichler. Performance Evaluation of the IEEE 802.11p WAVE Communication Standard. In *IEEE Vehicular Technology Conference*, 2007.

[10] J. Eriksson, H. Balakrishnan, and S. Madden. Cabernet: Vehicular Content Delivery Using WiFi. In *ACM MobiCom'08*.

[11] R. Gass, J. Scott, and C. Diot. Measurements of in-motion 802.11 networking. In *WMCSA*, 2006.

[12] C. Gkantsidis and P. R. Rodriguez. Network coding for large scale content distribution. In *IEEE Infocom*, 2005.

[13] D. Hadaller, S. Keshav, T. Brecht, and S. Agarwal. Vehicular opportunistic communication under the microscope. In *ACM MobiSys*, 2007.

[14] D. Jiang and L. Delgrossi. IEEE 802.11p: Towards an international standard for wireless access in vehicular environments. In *VTC*, Spring 2008.

[15] A. Klemm, C. Lindemann, and O. Waldhorst. A special-purpose peer-to-peer file sharing system for mobile ad hoc networks. In *VTC*, Fall 2003.

[16] K. Lee, S.-H. Lee, R. Cheung, U. Lee, and M. Gerla. First experience with cartorrent in a real vehicular ad hoc network testbed. In *MOVE*, 2007.

[17] S.-H. Lee, U. Lee, K.-W. Lee, and M. Gerla. Content distribution in vanets using network coding: The effect of disk I/O and processing O/H. 2008.

[18] U. Lee, J.-S. Park, J. Yeh, G. Pau, and M. Gerla. Code torrent: content distribution using network coding in vanet. In *MobiShare '06: Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*, pages 1–5, New York, NY, USA, 2006. ACM.

[19] R. Mahajan, J. Zahorjan, and B. Zill. Understanding WiFi-based connectivity from moving vehicle. In *IMC'07*.

[20] Y. Minsky, A. Trachtenberg, and R. Zippel. Set reconciliation with nearly optimal communication complexity. *IEEE Transactions on Information Theory*, 49(9), 2003.

[21] M. Mitzenmacher. Compressed Bloom filters. *IEEE/ACM Transactions on Networking*, 10(5), 2002.

[22] J. Ott and D. Kutscher. A disconnection-tolerant transport for drive-thru internet environments. In *Infocom 2005*.

[23] J. Ott and D. Kutscher. Drive-thru Internet: IEEE 802.11b for "automobile" users. In *Infocom 2004*.

[24] L. Smith, R. Beckman, and K. Baggerly. Transims: Transportation analysis and simulation system. Technical report, LA-UR–95-1641, Los Alamos National Lab., 1995.

[25] D. B. Terry, M. M. Theimer, K. Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing update conflicts in Bayou, a weakly connected replicated storage system. In *ACM SOSP*, 1995.

[26] X. Zhang, J. Kurose, B. N. Levine, D. Towsley, and H. Zhang. Study of a bus-based disruption tolerant network: Mobility modeling and impact on routing. In *ACM Mobicom'07*.