

# SCAP: Smart Caching in Wireless Access Points to Improve P2P Streaming

Enhua Tan<sup>1</sup>, Lei Guo<sup>1</sup>, Songqing Chen<sup>2</sup>, and Xiaodong Zhang<sup>1</sup>

<sup>1</sup>Dept. of Computer Science and Engineering  
The Ohio State University  
Columbus, OH 43210, USA  
{etan, lguo, zhang}@cse.ohio-state.edu

<sup>2</sup>Dept. of Computer Science  
George Mason University  
Fairfax, VA 22030, USA  
sqchen@cs.gmu.edu

## Abstract

*The increasing number of wireless users in Internet P2P applications causes two new performance problems due to the requirement of uploading the downloaded traffic for other peers, limited bandwidth of wireless communications, and resource competition between the access point and wireless stations. First, an active P2P wireless user can significantly reduce the downloading throughput of other wireless users in the WLAN. Second, the slowdown of a P2P wireless user communication can also delay its relay and data sharing service for other dependent wired/wireless peers.*

*In order to address these problems, in this paper, we propose an efficient caching mechanism called SCAP (Smart Caching in Access Points). Conducting intensive Internet measurements on representative P2P streaming applications, we observe a high percentage of duplicated data packets in successive downloading and uploading data streams. Through duplication detection and caching at the access point, these duplicated packets can be compressed so that the uploading traffic in the WLAN is significantly reduced. Our prototype-based experimental evaluation demonstrates that by effectively reducing the redundant P2P traffic in the WLAN, SCAP improves the throughput of the WLAN by up to 88% and reduces the response delay to other Internet users meanwhile.*

## 1 Introduction

Peer-to-peer (P2P) techniques have been successfully utilized in many daily applications on the Internet, such as large file distribution with BitTorrent [5], and VoIP telephony with Skype [16]. With the increasing demands of streaming media on the Internet [7], P2P techniques begin to be used to deliver streaming media on the Internet. The number of users of P2P-based IPTV and live media systems, such as PPLive [8], keeps increasing.

Meanwhile, wireless accesses to the Internet have become pervasive with widely deployed Wi-Fi networks on campus, in offices, at home, and public utilities, due to its low cost and ease of use. More and more people access Internet services via wireless connections, on both mobile devices such as laptops and stationary desktop computers.

However, media streaming over wireless channels is challenging due to the high bandwidth and real-time delivery requirement, and the unpredictable degradation of signal quality caused by noise, fading, attenuation, and interference in the physical communication channel. For P2P-based live streaming through wireless channels on the Internet, these challenges are even more serious for the following three reasons:

First, in a P2P-based streaming system, each peer has to upload the media content it receives to other peers as well as download media data from other peers. Due to the heterogeneity of P2P communications, some peers may work like a server and upload much more traffic to other peers than they consume. Thus, in a WLAN, the limited network bandwidth may be congested by the P2P uploading traffic. Furthermore, the slowdown of the P2P wireless peer communication can also delay its relay and data sharing service for other dependent wired/wireless peers.

Second, different from a wired network, in a wireless LAN, a station normally accesses the Internet through an access point (AP). In commonly deployed 802.11 WLANs equipped with the Distributed Coordination Function (DCF) channel access mechanism [12], all stations in the WLAN, including the AP, share the entire upstream and downstream bandwidth in the network. Since all downstream traffic must go through the AP, the downstream traffic has a lower priority for delivery due to the competition among physical stations [14]. The extra upstream traffic introduced by P2P streaming aggravates the channel competition between upstream and downstream traffic remarkably.

Third, the quality of streaming in a WLAN is very susceptible to the packet delay, loss, and retransmission caused by channel congestion [9] and signal quality degra-

dation [11]. For a station running P2P streaming, when the possibility of transmission errors exists due to channel condition problems, the extra upstream traffic can increase the number of transmission errors correspondingly, and the cost of contention window back-off caused by these errors may increase exponentially.

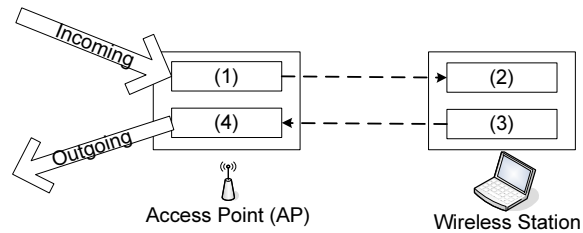
Thus, P2P-based live streaming over wireless LANs not only brings challenges on its own delivery quality, but may also degrade the performance of other network applications in the WLAN.

To address the above mentioned problems, in this paper, we propose an efficient caching mechanism to reduce the upstream traffic in P2P-based live streaming, called SCAP (Smart Caching in Access Points). We conduct Internet measurements on several representative P2P streaming applications, and find that a high temporal locality among the P2P streaming traffic exists. Our measurements show that, on average, for P2P-based streaming, there are more than 80% duplicated data packets in successive downloading and uploading data streams. Motivated by the measurement results, we propose a low overhead traffic reduction scheme based on Rabin fingerprinting [15] to quickly identify the downstream packets that will be uploaded to or relayed to other peers. Instead of uploading the entire data packet, SCAP temporarily caches the corresponding downloaded packets in the AP, and the relay peer only uploads a small identity tag to the AP, significantly reducing upstream traffic in the WLAN. Upon receiving a data tag, the AP recapsulates the uploaded packet from the associated data packet in its cache, and sends it to the remote peer on the Internet. SCAP can effectively reduce the competition for the limited bandwidth in WLANs, and improve service quality for the Internet peers. Our prototype-based experimental evaluation results demonstrate that by effectively reducing the redundant P2P traffic in WLAN, SCAP improves the throughput of the WLAN by up to 88% while also reducing the response delay to other Internet users.

The rest of the paper is organized as follows. We sketch our proposed application-independent traffic reduction scheme in section 2. Our measurements are conducted in section 3. Section 4 details our design and implementation. The prototype system is evaluated in section 5. Section 6 describes some related work and we make concluding remarks in section 7.

## 2 Basics of the Traffic Reduction Scheme

For P2P-based streaming, the received data should be forwarded to other peers, and thus, if a wireless user is involved, the same media content will be downloaded through the AP first, and then will be uploaded through the AP to other peers successively several times depending on how many peers this wireless station needs to upload to. This



**Figure 1. A workflow of the traffic reduction scheme: AP stores downstream data in buffer (1), which is the same as the content stored in buffer (2) on the wireless station. If the upstream data in buffer (3) is same or largely duplicated with that in buffer (2), the wireless station only needs to supply the difference to the AP, which assembles with data in buffer (1) to deliver.**

data communication pattern provides us with an opportunity to reduce the amount of traffic between the AP and the wireless station by exploiting temporal locality through caching. The basic idea is as follows: we reserve a buffer in the AP to store the recently transmitted downstream data. In each wireless station, a buffer is also reserved to store the recently received data from the AP. Once the wireless station needs to send data out, it compares whether the upstream data is the same as the downloaded data in its buffer. If they are the same, the wireless station only needs to send information about the destination of this upstream data chunk to the AP, and the AP can simply assemble the original packet with the data stored in its local buffer and send it out. Figure 1 shows such a workflow.

There are two technical merits in our scheme. First, it is application independent; and second, it should work well for wireless users in all P2P applications, since it is expected that a wireless user will upload the same data it just received to the next peer(s).

However, in practice, even for P2P streaming applications, upstream and downstream data may not always be the same for two reasons. First, the local client software may impose some processing on the received data. For example, the local client software may reduce the encoding rate (or resolution) of the media content before uploading. Second, the network stack may perform some processing as well. For example, several downloaded packets may be assembled and re-packetized before uploading. For the former case, the data duplication could be very small, even with a very large buffer. For the latter case, if a reasonably sized buffer is reserved, it is still possible to identify and reduce most redundant traffic, in which the buffer size should take the delay into consideration.

Thus, the effectiveness of our traffic reduction scheme is

highly dependent on the applications, although the scheme itself is application-independent. Since the amount of data duplications is critical to the success of our scheme, we will present in the next section a measurement study of data duplications in several representative P2P streaming applications.

### 3 Duplication Detection Methods

#### 3.1 Hash and Rabin Fingerprinting based Duplication Detection

To efficiently identify data duplications between upstream and downstream channels, an option is to use a hash function. However, if the application re-packetizes the data before sending it out, running the hash function with continuously varying input length starting from various offsets may lead to low efficiency.

In contrast, fingerprints may result in better performance in such situations. Similar to hash values, fingerprints are short tags for large objects. Generally, if two fingerprints are different, then the corresponding objects are different, because the probability that two different objects have the same fingerprint is very small. For our purpose, we particularly adapt Rabin Fingerprinting.

Rabin Fingerprinting [15] is a method for implementing public key fingerprint using polynomials over a finite field, which works as follows. Let  $A$  denote a string of  $A = (a_1, a_2, \dots, a_m)$ . If we associate the string  $A$  with a polynomial  $A(t)$  of degree  $m - 1$ ,

$$A(t) = a_1 t^{m-1} + a_2 t^{m-2} + \dots + a_m,$$

and if  $P(t)$  is an irreducible polynomial of degree  $k$ , then Rabin Fingerprint of  $A$  is the polynomial

$$RF(A) = A(t) \bmod P(t).$$

Rabin Fingerprinting has the property of the distributive over addition, i.e.,  $RF(A+B) = RF(A) + RF(B)$ . Based on this, Rabin Fingerprinting is particularly computationally advantageous in that if

$$\begin{aligned} RF(a_1, \dots, a_m) &= (a_1 t^{m-1} + a_2 t^{m-2} + \dots + a_m) \bmod P(t) \\ &= r_1 t^{k-1} + r_2 t^{k-2} + \dots + r_k \end{aligned}$$

then

$$\begin{aligned} RF(a_1, \dots, a_m, a_{m+1}) &= r_1 t^k + r_2 t^{k-1} + r_3 t^{k-2} + \dots + r_k t + a_{m+1} \bmod P(t) \\ &= (RF(a_1, \dots, a_m) t + a_{m+1}) \bmod P(t) \end{aligned}$$

This leads to fast fingerprint computation for a continuous data stream. Instead of generating a new fingerprint

from scratch for a string, advancing the fingerprint requires just some linear computation: an addition, a multiplication, and a mask.

This property enables Rabin Fingerprinting to compute tags for a continuous input stream easily, which would be difficult if MD5 or SHA were used. Rabin Fingerprinting has been used in a number of applications for similarity or redundant traffic finding (see e.g. [17]). In our study, we will use it and compare our algorithm with hash-based algorithms.

#### 3.2 Measurement and Analysis of Internet P2P Streaming Traffic

In this subsection, we study the traffic characteristics of several P2P streaming applications. We aim to answer the following questions: (1) how much is there duplicated traffic in practice? (2) what is the cost, in terms of computing cycles and buffer sizes, to identify such duplications?

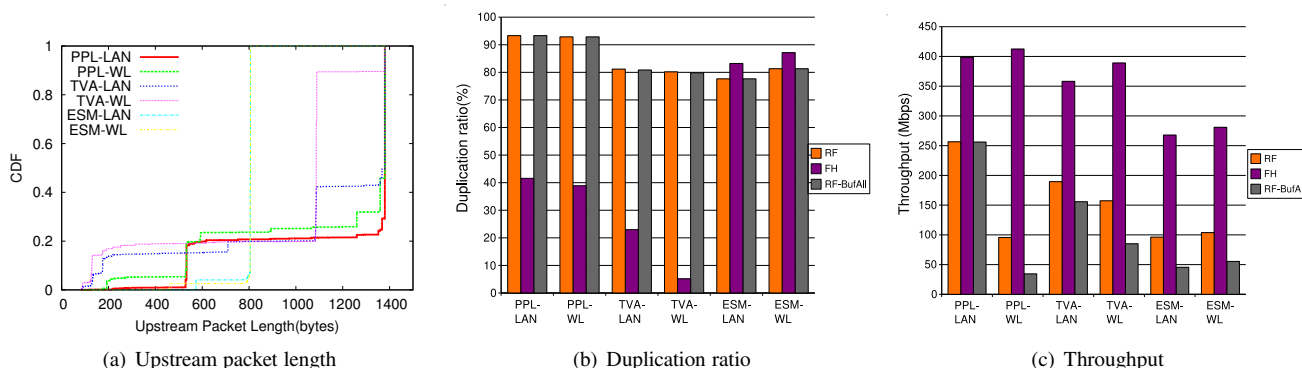
We have collected traces from several representative P2P applications, including PPLive [1], TVAnts [2], and ESM [3], in our LAN/WLAN with Ethereal by running these applications from September to November in 2006. The outgoing bandwidth for the LAN is 100 Mbps, and the effective connection speed for the wireless station is 5.5 Mbps. We collect traffic in both WLAN and LAN, in order to compare whether their characteristics are different.

Two algorithms have been used for data duplication detection in our traces. The first one is based on Rabin Fingerprinting (simplified as *RF*). This algorithm scans each downstream packet and extracts certain fingerprints (ending with  $\gamma$  zeros) for a fixed-length ( $\beta$ ) content, which are used as landmarks for later searching based on the fingerprints of an upstream packet. Another detection algorithm is based on hashing at a fixed position of a packet (simplified as *FH* for Fixed Hashing), which computes the hash value for the first  $\beta$  bytes of the downstream packet starting from the first  $\alpha$  bytes (which is possibly the application level header), and stores this value for later comparison with the hash value computed from an upstream packet. We restrict the number of buffered downstream packets to 50,000 (which needs about 75MB memory for a packet size of 1,380 bytes), and we use  $\gamma = 8$ ,  $\beta = 64$  settings. We use  $\alpha = 16$  for all applications except for ESM, since its application level header is much larger, in which case we set  $\alpha = 54$ .

Table 1 summarizes the statistics of the workloads. *PPL* represents PPLive trace, and *TVA* is short for TVAnts workload. The workload name ending with *WL* is collected in WLAN. According to the *Up/Down Ratio* column, the upstream to downstream traffic ratio is different among these workloads, affected by a number of possible factors, including the numbers of peers, layers, and file chunks to upload. Compared with the traces collected in the wireless environ-

**Table 1. Workload summary**

Workload Name	Duration (sec)	Downstream Bytes(MB)	Upstream Bytes(MB)	Up/Down Ratio(%)	Downstream Packet #	Upstream Packet #	Downstream Tput(Mbps)	UDP bytes Ratio(%)
PPL-LAN	953	50.504	509.304	<b>1008.45</b>	406,804	561,117	0.444	0.66
PPL-WL	14989	697.566	497.906	71.38	1,123,534	1,213,308	0.390	2.58
TVA-LAN	5054	234.765	890.948	379.51	884,892	1,189,054	0.390	21.72
TVA-WL	8073	415.741	859.389	206.71	979,864	1,314,854	0.432	73.89
ESM-LAN	10845	403.512	110.399	27.36	758,170	600,122	0.312	1.30
ESM-WL	11237	389.513	178.394	45.80	782,350	663,400	0.291	1.17



**Figure 2. Workload analysis**

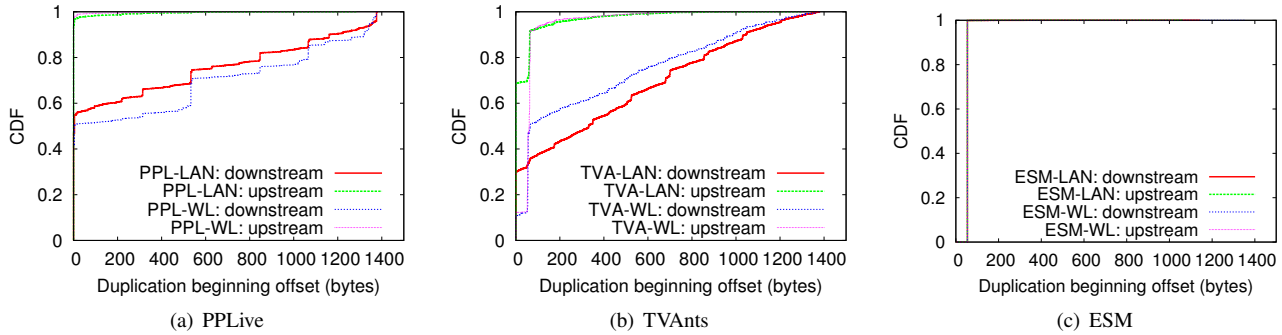
ment, PPLive and TVAnts workloads collected in the LAN have a larger volume of upstream traffic (up to 10 times more than the downstream traffic), while ESM has a modest amount of upstream traffic. Most of PPLive and ESM's traffic is delivered in TCP, while TVAnts is more favorable in UDP. The average downloading throughput is around 300 to 400 Kbps. Figure 2(a) shows the upstream packet length distribution for our workloads (packets with zero payload length are not included). As shown in the figure, most of the upstream packets of PPLive are sent in 1,380 bytes, and most of ESM are in 800 bytes, while TVAnts' are in between. A larger packet length could lead to more benefit from our traffic reduction scheme because the overhead of transmitting a upstream packet in 802.11 protocol would be better amortized by the heavier compression of the payload.

Figure 2(b) shows the duplication ratio, which is the detected upstream packet duplication bytes over the total upstream bytes. The *RF-BufAll* represents the computed duplication ratio when all the downstream packets are buffered with the RF-based algorithm. The figure shows that different applications have different duplication ratios for upstream traffic, and in our long term run, such trends do not vary significantly. In addition, the duplication ratio of traffic through the LAN does not differ much from that in the WLAN. As indicated in the figure, RF achieves better performance on average (above 70% duplication ratio on all workloads, and about 90% for the popular PPLive and 80% for TVAnts) with the exception on ESM. For ESM, FH

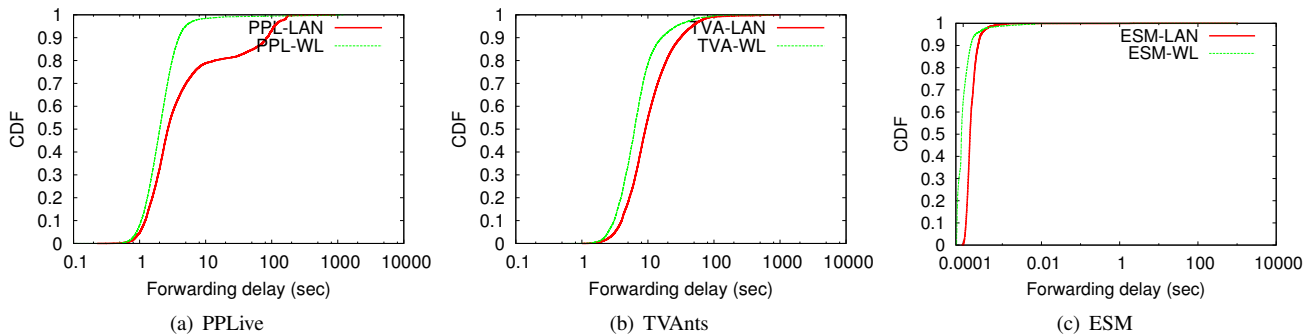
achieves an even slightly better result. Our further analysis indicates that this is because ESM does not impose any local processing. Thus, the randomized fingerprint selection in RF performs a little bit worse than fixed hashing in FH.

Figure 2(c) shows the processing throughput for each algorithm on different workloads. The throughput of the RF-based algorithm ranges from 90 Mbps to 256 Mbps, which implies that the computing overhead will be very small for manipulating P2P streaming traffic with about 400 Kbps throughput.

Comparing the RF-based algorithm with the FH-based one, we find that while the RF-based detection costs more CPU cycles, it generally performs better than the FH-based algorithm due to its flexible duplication offset handling. In practice, an upstream packet may contain the content from two consecutive downstream packets depending on the application implementation of handling the data to be transferred to another peer. The FH-based algorithm cannot effectively detect this kind of duplications since it will only compute the hash value of the first several bytes of the packets by assuming an upstream packet is exactly a replica of its corresponding downstream packet (except the application level header). The RF-based algorithm avoids this problem by randomly generating fingerprints throughout the packet content for later lookup. Figure 3 shows the duplication beginning offset distribution for each workload computed with the RF-based algorithm. If the beginning offset in the downstream packet is not equal to that in the upstream



**Figure 3. Duplication beginning offset in the downstream and upstream packet**



**Figure 4. Duplication packets forwarding delay**

packet, or if they are equal, but both are larger than the presumed application level header size, the FH-based algorithm fails to find the duplications. As shown in Figure 3, TVAnts performs more local processing than PPLive, while ESM does not do any. These results explain the difference of RF and FH detected duplication ratio for different applications shown in Figure 2(b).

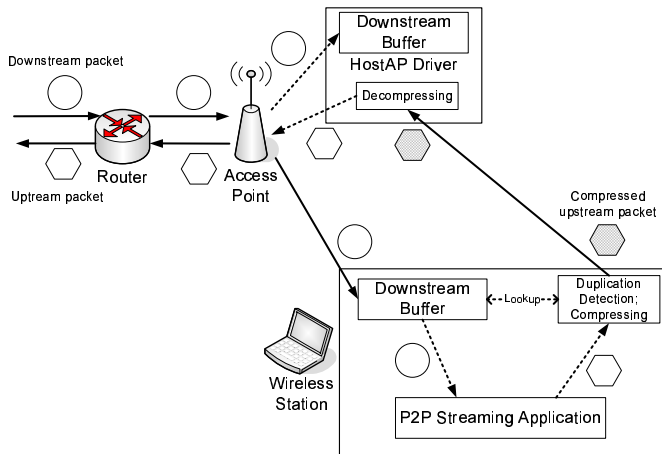
Figure 4 shows the distribution of the duplication packets forwarding delay in upstream and downstream flows. As shown in the figure, more than 70% of PPLive duplication upstream packets are forwarded in 10 seconds after they are received, and more than 60% of TVAnts duplication upstream packets are forwarded in 10 seconds. Both PPLive and TVAnts forward most of their received traffic within 200 seconds. The average forwarding latency is 20.5 seconds for PPL-LAN, 27.0 seconds for PPL-WL, 46.1 seconds for TVA-LAN, and 51.6 seconds for TVA-WL. Most of ESM's upstream packets are forwarded within 1 second. Small forwarding latency means that a small buffer size for downstream packets can be sufficient for duplication detection.

Our measurement results show that there is a significant portion of data duplications in the upstream and downstream channels in WLANs, which can be detected with very low overhead. The results also show that the upstream

throughput is comparable to the downstream throughput, which may lead to severe unfairness to the other downloading traffic [14]. If such P2P traffic is not well managed with in a WLAN, the users of the WLAN could be severely impacted, including the wireless peer itself, which would further affect the performance of other Internet peers who rely on its service. In the next section, we will present the design and implementation of our application-independent traffic reduction scheme to address these problems.

## 4 Design and Implementation of SCAP

Figure 5 illustrates the workflow of our design. For the RF (Rabin fingerprinting) based duplication detection algorithm, both the AP and the station need to scan the entire downstream packets in order to generate the fingerprints and to store them to the downstream buffer (a FIFO queue). When the station has an upstream packet to send, it computes several fingerprints that end with  $\gamma$  zeros for the packet. Each fingerprint is looked up in a hash table recording the index of the downstream packet and the starting offset of the bytes mapping to that fingerprint. Once a fingerprint is found in the hash table, the upstream packet is compared with the corresponding downstream packet(s) byte-by-byte to expand that duplication area. Then, the upstream



**Figure 5. Traffic reduction scheme workflow**

packet is compressed to the length of non-duplication area with additional headers indicating the information of duplication. The AP will decompress the received upstream packet to its original state by identifying the duplication downstream packet(s) in its buffer and copying the duplication area back to the packet. For the FH (fixed hashing) based algorithm, the AP and the station only need to scan  $\beta$  (64) bytes to get one hash value for a downstream/upstream packet.

The buffer size of downstream packets is an important factor for our scheme to be cost-effective. For example, as shown in Figure 4, most of PPLive and TVAnts' duplication packets are sent in 200 seconds. Thus, assuming the downstream bit rate is 300 Kbps, a buffer size to store 200-second traffic will be 7.5 MB, which is very small for a modern computer device. However, considering that an AP may need to serve many wireless stations, its memory space is more precious than a station. In fact, an adaptive algorithm that dynamically adjusts the buffer size according to the detected duplication ratio on line is more effective. A simplest design could work as follows: the system monitors the duplication ratio in a small time interval (e.g., the last few minutes ( $OR_m$ )) and in a larger time interval (e.g., the last hour ( $OR_h$ )). If  $OR_m$  is less than a threshold, which could be set as a certain percentage of  $OR_h$ , the buffer size is increased by a corresponding scale; if it is larger than another threshold, the buffer size is reduced accordingly. On the other hand, the thresholds for  $OR_m$  can also be adjusted. For example, if the upstream traffic is much heavier than the downstream traffic, the low threshold should be increased to reduce more upstream traffic. Furthermore, each station only needs to monitor the duplication ratio of its own traffic for changing the buffer size, while the AP needs to monitor the duplication ratio for each station in order to dynamically partition the buffer for it.

Another important design issue is that the downstream content of a station should always be present in the AP. Otherwise, the compressed upstream packet cannot be decompressed in the AP. Thus, when a station is running a P2P streaming application, it notifies the AP to buffer the downstream buffer for it. If a compressed upstream packet cannot be decompressed in the AP, the TCP flow will be stalled by trying to send the compressed packet again and again. To recover from this kind of buffer inconsistency, the station caches several copies of the latest sent original upstream packets, and sends the original copy when the station finds the compressed one is being resent 3 times.

As shown in Figure 1, to implement the scheme, we need to modify the network stack of the AP and wireless stations. Although this scheme can be implemented in the IP layer of the network stack, since it is mainly compressing and decompressing the content of TCP/UDP payload, we actually implement it in the wireless network adapter driver so that only wireless packets are affected. In addition, we can also get more header information there than in the IP layer.

We have implemented a prototype system by modifying the HostAP driver in the Linux 2.6.16 kernel for the AP and wireless stations based on [6]. Our wireless cards are based on the Intersil Prism 2.5 chipset.

In the implementation, the identification of the downstream packet is critical, since when an upstream packet is found to be duplicated with a downstream packet in the station buffer, the AP needs to determine which downstream packet it should locate in its buffer. Although it is straightforward to use the Sequence Control field (2 bytes) in the 802.11 frame header together with the station's MAC address for such identification, this method fails because the sequence number is not generated by the driver, but by the firmware of the card. Thus, in our implementation, we use the first fingerprint (or hash value for FH) of the downstream packet instead, which introduces extra overhead for its length (8 bytes).

## 5 Performance Evaluation

In order to verify the effectiveness of the traffic reduction scheme, we evaluate the prototype system in our experimental environment. The wireless stations are placed close to the AP to ensure the signal quality. The data transfer rate is 11 Mbps. We set the buffer size to 500000 packets (75 MB memory), and set  $\alpha = 54$ ,  $\gamma = 8$ ,  $\beta = 64$ . We manually set the AP to only buffer the downstream packets for the wireless station using the modified driver, thus the two buffers in the AP and the wireless station are always synchronized.

We first tested a microbenchmark on our prototype system. The benchmark runs on a wireless station and a wired station in the LAN. The wireless station first down-

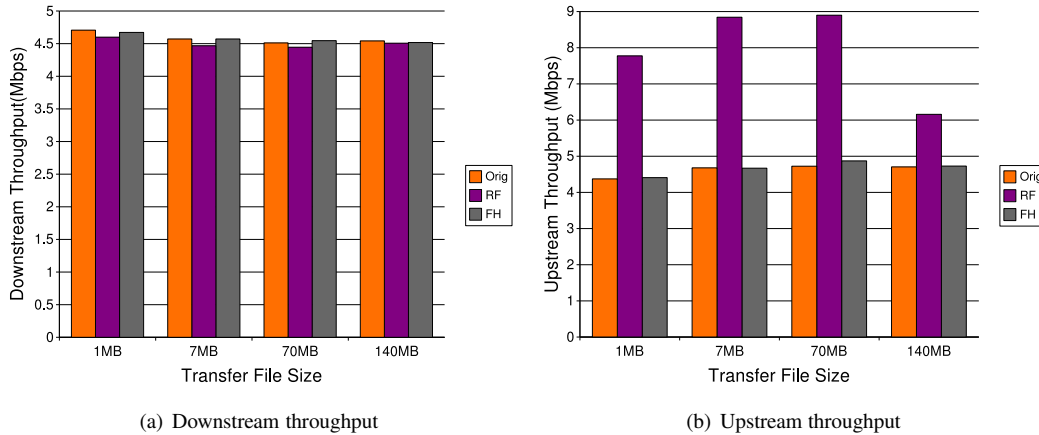


Figure 6. Microbenchmark results

loads a file from the wired station, then sends it back. Figure 6 shows the downstream and upstream throughput by running the benchmark on the original kernel, the RF-based one, and the FH-based one. The RF-based algorithm demonstrates little overhead for the downstream throughput (1.5% decrease) and about 88% improvement of the upstream throughput in the 70 MB file transfer case. When the file size gets larger, the improvement decreases because of fewer buffered downloading packets. However, the FH-based algorithm does not show any improvement because the downstream duplication beginning offset is always 1024 bytes, while the upstream duplication beginning offset is 0. Thus fixed hashing cannot detect any duplication content. The overhead of FH-based algorithm is close to zero comparing with the original kernel.

In the next experiment, we evaluated PPLive, TVAnts, and ESM as representative P2P streaming applications. Because they are all Windows based applications, we used VMware to run the application in a Windows XP guest OS. Our modified driver in the host OS can capture the packets from the guest OS and compress the upstream packet when duplication is detected. We run the P2P application on one wireless station with our modified kernel. On another station with unmodified kernel driver, we run one TCP session using iperf 1.7.0<sup>1</sup>, and run a Ping session with a packet size of 1500 bytes, in order to observe the impact of throughput and response time of our schemes. Because our experiments are running in an open environment with different peers each time, we run the experiment multiple times to capture the trials with similar P2P downstream and upstream throughput for comparison. Each trial runs for 10 minutes.

Table 2 shows the average results of PPLive by running the original, RF and FH based kernel drivers. As shown

Table 2. PPLive evaluation

	TCP Tput (Mbps)	DS Tput (Mbps)	US Tput (Mbps)	Ping RTT (ms)	Duplication Ratio(%)
Orig	3.25	0.410	0.533	179.6	-
FH	3.35	0.396	0.462	172.7	22.25
RF	3.52	0.408	0.532	165.6	71.82

Table 3. TVAnts evaluation

	TCP Tput (Mbps)	DS Tput (Mbps)	US Tput (Mbps)	Ping RTT (ms)	Duplication Ratio(%)
Orig	1.59	0.413	1.555	316.9	-
FH	2.19	0.433	1.841	276.9	34.83
RF	2.54	0.399	1.766	234.2	86.60

in the table, RF can detect more (a larger duplication ratio) than FH, which leads to better performance for the TCP session and the Ping session. By reducing about 70% of the upstream traffic (which is 0.38 Mbps), RF successfully increases the average throughput of the TCP session by 0.27 Mbps (71% of the reduced traffic). RF also decreases the average Ping round-trip time (RTT) by 14 ms, which implies that the delay to the serviced Internet peers of P2P streaming is also reduced.

Table 3 shows the corresponding results of TVAnts. The upstream throughput for TVAnts is much larger than that of PPLive. Reducing 86% of the upstream traffic (1.53 Mbps), RF increases the TCP throughput by 0.95 Mbps (62% of the reduced traffic), and decreases the Ping round-trip time by 83 ms. FH also shows good performance improvement by reducing over 30% of the upstream traffic (0.64 Mbps) to increase the TCP throughput by 0.60 Mbps and decrease the Ping round-trip time by 40 ms.

<sup>1</sup><http://dast.nlanr.net/Projects/Iperf/>

**Table 4. ESM evaluation**

	TCP Tput (Mbps)	DS Tput (Mbps)	US Tput (Mbps)	Ping RTT (ms)	Duplication Ratio(%)
Orig	3.86	0.319	0.319	143.3	-
FH	3.97	0.319	0.312	139.2	89.17
RF	3.96	0.318	0.320	139.0	85.94

The results of ESM are shown in Table 4. Because of the smaller amount of upstream traffic and the smaller packet size, the improvement is not as high as that of PPLive and TVAnts. The performance of FH is comparable to that of RF because the detected duplication ratios are comparable.

## 6 Other Related Work

An early study [17] has noticed that even after the proxy caching, a significant amount of traffic that a proxy cannot cache carries similar information. A protocol independent technique is proposed to detect and reduce such redundant traffic. Similarly, LBFS in [13] also exploits the similarities between files and versions of the same file on the slow or wide area network file systems.

In work [4], network coding was first proposed to improve the throughput of a network by considering the routing characteristics. The work in [10] further improves the throughput of multi-hop wireless networks by utilizing the broadcasting nature of wireless networks. Our proposed traffic reduction scheme differs from network coding in the sense that the effectiveness of our scheme is highly related to the traffic pattern of the P2P streaming applications (traffic duplication ratio) instead of other factors.

## 7 Conclusion

With the increasing popularity of P2P streaming applications on the Internet and the pervasive deployment of 802.11 WLANs, an increasing number of peers in P2P networks are wireless users. In this work, we study the impact of the increase of wireless users in Internet P2P applications to the performance of wired and wireless users. Focusing on quality sensitive live streaming applications, we show that, without a proper control of P2P traffic between the access point and wireless peers, the performance of both the Internet users that rely on the wireless peer's service and other wireless users in the WLAN can be significantly affected. We have designed and implemented an effective traffic reduction scheme through caching and coordination between the AP and the wireless users. Our prototype based performance evaluation results show that such a P2P streaming traffic reduction could effectively improve the throughput of the WLAN and reduce the delay to other Internet users.

## 8 Acknowledgment

We thank the constructive comments from the anonymous referees. This work is partially supported by the National Science Foundation under grants CNS-0405909, CNS-0509054/0509061, and CNS-0621629/0621631.

## References

- [1] <http://www.pplive.com/en/>.
- [2] <http://www.tvants.com/>.
- [3] <http://esm.cs.cmu.edu/>.
- [4] R. Ahlswede, N. Cai, S. Li, and R. Yeung. Network information flow. In *IEEE Transaction on Information Theory*, volume 46, pages 1204–1216, July 2000.
- [5] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang. Measurement, analysis, and modeling of BitTorrent-like systems. In *Proc. of IMC*, Oct 2005.
- [6] L. Guo, X. Ding, H. Wang, Q. Li, S. Chen, and X. Zhang. Exploiting idle communication power to improve wireless network performance and energy efficiency. In *Proc. of IEEE INFOCOM*, Barcelona, Spain, April 2006.
- [7] L. Guo, E. Tan, S. Chen, Z. Xiao, O. Spatscheck, and X. Zhang. Delving into internet streaming media delivery: a quality and resource utilization perspective. In *Proc. of IMC*, Oct 2006.
- [8] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross. Insights into PPLive: A measurement study of a large-scale P2P IPTV system. In *Proc. of IPTV Workshop*, Edinburgh, Scotland, UK, May 2006.
- [9] A. Jardosh, K. Ramachandran, K. Almeroth, and E. Belding-Royer. Understanding congestion in IEEE 802.11b wireless networks. In *Proc. of IMC*, Oct 2005.
- [10] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. XORs in the air: Practical wireless network coding. In *Proc. of ACM SIGCOMM*, Pisa, Italy, September 2006.
- [11] D. Kotz, C. Newport, and C. Elliott. The mistaken axioms of wireless-network research. Technical Report TR2003-467, Computer Science, Dartmouth College, July 2003.
- [12] LAN/MAN Standards Committee of the IEEE Computer Society. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Technical Report IEEE Std 802.11, the Institute of Electrical and Electronics Engineers, Inc., 1999.
- [13] A. Muthitacharoen, B. Chen, and D. Mazieres. A low-bandwidth network file system. In *Proc. of SOSIP*, 2001.
- [14] S. Pilosof, R. Ramjee, D. Razl, Y. Shavitt, and P. Sinha. Understanding TCP fairness over wireless LAN. In *Proc. of IEEE INFOCOM*, Apr. 2003.
- [15] M. O. Rabin. Fingerprinting by random polynomials. Technical Report TR-15-81, Department of Computer Science, Harvard University, 1981.
- [16] S. Ren, L. Guo, and X. Zhang. ASAP: an AS-aware peer-relay protocol for high quality VoIP with low overhead. In *Proc. of IEEE ICDCS*, July 2006.
- [17] N. Spring and D. Wetherall. A protocol-independent technique for eliminating redundant network traffic. In *Proc. of ACM SIGCOMM*, Stockholm, Sweden, 2000.