Eliminating Communication Redundancy in Wi-Fi Networks

Zhenyun Zhuang
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332
Email: zhenyun@cc.gatech.edu

Raghupathy Sivakumar
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332
Email: siva@ece.gatech.edu

***Abstract*—Studies have shown the presence of considerable amounts of redundancy in Internet traffic content. Recent works are exploring possibilities for exploiting network traffic redundancy, but these works invariably focus on fixed wireline networks. Unlike wireline networks, wireless and mobile environments exhibit unique challenges and opportunities in the context of redundancy elimination.**

In this work, we explore leveraging network traffic redundancy, but exclusively focus on wireless and mobile environments. We first analyze real Wi-Fi traces, and based on insights obtained from the analysis, we propose *Wireless Memory (WM)*, a two-ended AP-client solution to effectively exploit traffic redundancy for such environments. Trace-driven evaluation results show that WM can help deliver significant throughput improvement.

***Keywords*-Wireless memory; Traffic redundancy;**

I. Introduction

Several recent studies [1]–[4] have shown the presence of considerable amounts of redundancy in Internet traffic content. Such redundancies in content can be explicitly eliminated to improve communication performance. There are various approaches [1]–[3], [5]–[7] that have been proposed to eliminate such redundancy. Ranging from application-layer to network layer strategies, these works invariably focus on fixed wireline networks.

Similar to the above works, we too explore leveraging network traffic redundancy, but exclusively focus on wireless and mobile environments. Unlike wireline networks, wireless and mobile environments exhibit unique challenges and opportunities in the context of redundancy elimination. On one hand, the broadcast nature of wireless communication enables techniques such as packet sniffing to be performed with ease, while on the other hand, mobility and location based channel variances could impose challenges that have to be effectively addressed. Perhaps most importantly, given the typical resource constraints of wireless environments, redundancy elimination could have a profound impact on performance delivered to users.

In this paper, we focus on one popular type of wireless networks: 802.11b/g (or Wi-Fi). We first study the traffic redundancy along multiple dimensions using traces obtained from multiple real wireless network deployments. Specifically, we consider three buildings and two Wi-Fi network deployments in a major university campus. One of the buildings is a mixed-use environment that houses several small-medium businesses. Based on the insights obtained from the analysis, we propose *Wireless Memory (WM)*, a two-ended AP-client solution to effectively exploit traffic redundancy in wireless and mobile environments. Generically, WM equips AP and clients with *memory* to enable memorization of content as it flows naturally through the wireless network, and more importantly use the memory to lower the actual cost of delivering any content to its intended destination. We evaluate WM through simulations driven by the collected Wi-Fi traces, and show that WM can improve the network throughput by up to 93% in certain scenarios.

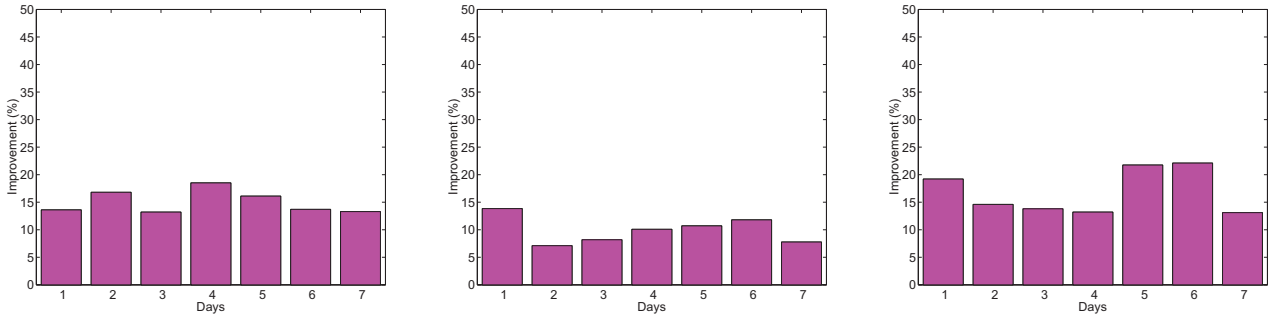
The remaining paper is presented as follows. In Section II, we motivate our design by presenting a set of observations. In Section III, we present the design and operations of wireless memory. We perform trace-driven evaluation and show the results in Section IV. Finally we present related work and conclude the work in Section V and VI, respectively.

II. Motivation

In this section we motivate our design of Wireless Memory by analyzing collected Wi-Fi traces. The use of Wireless Memory helps only when content stored in the memory will be referenced for "future" communications. Consequently, a necessary condition for Wireless Memory to provide benefits is redundancy in traffic content. Though an extensive study of the nature of redundancy in wireless traffic is a non-trivial task, we present some preliminary indicators of traffic redundancy that motivate the design of Wireless Memory. We perform studies primarily to verify that redundancy does exist for practical users. In addition, these results also shed light on our Wireless Memory design.

Data redundancy has long been observed and studied in literature, and depending on the nature of redundancy, there are two types of redundancy: *intra*-redundancy and *inter*-redundancy. It is well known that data redundancy inside a data unit (e.g., a packet) can be eliminated by applying compression mechanisms such as GZip. However, conventional compressions are unable to eliminate redundancy that exists across data units (e.g., between two packets or two html files) unless these data units are processed with the

978-1-4577-0351-5/11/$26.00



(a) Building A (b) Building B (c) Building C

Figure 1. **Inter-user Dimension (Dominant user vs all other users)**

same compression session. We show that, by effectively eliminating inter-redundancy, data size can be significantly reduced, and in turn, the throughput can be improved.

In this section we study the potential improvement on data reduction by eliminating inter-redundancy. Specifically, we compare the resultant data size of eliminating all redundancy (both intra- and inter-) to that of eliminating only intra-redundancy. For simplicity, we refer to an ideal compression that eliminates both intra- and inter- redundancy as *memory*-based compression, since the elimination of inter redundancy essentially treats the previous data as "memories". Correspondingly, we refer to the naive compressions that only remove intra-redundancy as *non-memory*-based compression.

A. Methodology

Our study is based on real Wi-Fi traces. Specifically, we perform a 4-month wireless sniffing in 3 buildings of a major university campus. The Wi-Fi networks sniffed are two 802.11g networks, and we use four Ubuntu PCs equipped with Wi-Fi cards and MadWifi (http://madwifi-project.org/). The two networks use WEP to encrypt traffic. By running MadWifi in Managed mode, the PCs are able to decode the live traffics of other wireless users in the same network. [1]

To evaluate the potential improvement of eliminating inter-redundancy, we adopt the following process. First, the captured live data stream is split into packets. We consider a naive approach to redundancy-elimination by compressing each individual packets before transmission. Note such compression is independent from application-layer compression, as the captured live traffic is the as-is traffic, and they may have been "compressed" by applications. Such a naive packet-based compression can effectively eliminate intra-redundancy (i.e., redundancy inside the packet) only. Second, an ideal approach would eliminate both intra- and inter-redundancy. So to mimic such an approach, we treat the previous packets as memory, and compress the current packet based on the memory. This method is based on the fact that most typical compression algorithms (e.g., LZW [8]) process byte streams sequentially, memorize encountered byte sequences, and represent later repeated byte sequences with codes. Thus, with such a method, the packet size after eliminating both types of redundancy is estimated as the *incremental* coded size, which is the size difference of: (i) only compressing the data consisting all previous packets, and (ii) compressing the data consisting all previous packets and the current packet.

We choose a compression utility of Rzip [9] [2]. Specifically, for a particular live traffic data set, we treat the trace as a byte stream of D and split it into data pieces of d_i, where $0 \leq i \leq I$. We also use D_i to denote the set of data pieces from d_0 to d_i, so we have $D_i = \{d_0+, ..., +d_i\}$. Assuming existing compression algorithm (e.g., Rzip) can remove all intra-redundancy, the coded size of d_i is thus $Rzip(d_i)$. Similarly, the coded size of D_i is $Rzip(D_i)$. The incremental coded size of d_i is the difference of the codes of D_i and D_{i-1}. Denoting the incremental coded size is C_i, we have $C_i = Rzip(D_i) - Rzip(D_{i-1})$, and we assume C_i is the ideal coded size of d_i with memory of D_{i-1}. So compared to the non-memory-based solution which has the coded size of $Rzip(d_i)$, the effectiveness of an ideal memory-based solution is measured by $1 - \frac{C_i}{Rzip(d_i)} = [1 - \frac{Rzip(D_i) - Rzip(D_{i-1})}{Rzip(d_i)}]x100\%$. The value also shows the degree of data reduction by using memory-based solution when compared to the non-memory-based solution, and the larger the value is, the higher benefit can be achieved by eliminating inter-redundancy. For all the following results, we choose $d = 1.5KB$ for the simple reason that a typical IP packet is about that size.

B. Results

We study the potential improvements of an ideal memory-based approach in both inter-user and temporal dimensions. We correspondingly show representative results which will be used to motivate our design. Inter-user dimension studies the potential benefits of eliminating the redundancy between users. Briefly, considering a Wi-Fi network, for a particular user U_k, other users' data can be used as the base for compressing U_k's data. We analyze the potential improvement of compressing each user's trace by eliminating the user-user redundancy with other users. Specifically, given a data

[1] Though by associating to an AP, the sniffing desktop is able to see the decrypted traffics of other users, we explicitly perform hashing operations to store only the hash values of captured data to ensure anonymity.

[2] RZip is a huge-scale compression software designed to find and encode duplicated data over very long distances (e.g., 900 MB) in the input file.

User Pair	1	2	3	4	5	6	7	8	9
Building A	12	12	14	27	7	3	19	11	17
Building B	8	13	10	7	10	14	9	27	11
Building C	49	42	33	26	17	31	29	11	8

Table I
REDUNDANCY OF USER-PAIRS (%): DOMINANT VS. OTHER TOP USERS)

piece of d_i which contains multiple users' traffic, considering a user U_k we denote his data as $d_{i,k}$ and other users' data as $d'_{i,k}$, so we have $d'_{i,k} = d_i - d_{i,k}$. Similarly we denote the cumulative data that eliminate U_k's data as $D'_{i,k} = D_i - D_{i,k}$. For the particular user U_k and his data piece of $d_{i,k}$, the compressed size with non-memory-based compression is $Rzip(d_{i,k})$, and we can get the incremental coded size of $d_(i,k)$ as $C_{i,k} = Rzip(D'_{i-1,k} + d_{i,k}) - Rzip(D'_{i-1,k})$.

We show the results in Figures 1 for all 3 buildings. For each building, we choose 1-week of traces and study the dominant user (i.e., having the largest portion of traffic) by eliminating inter-user redundancy between himself and all other users. We observe that *there are substantial improvements by exploiting inter-user redundancy*, and the improvement ranges between 7% to 22%.

We further study the redundancy between individual users for the same day. For each of the 3 data sets, we choose the dominant user and study the redundancy between himself and the other top 9 users. The results are shown in Table I. We see that *some user-pairs have more inter-user redundancy than other user-pairs*. More studies into this dimension suggest that the results relevant to user-pairs are caused by the web access patterns of these users. Briefly, users with higher inter-user redundancy tend to visit the same set of web sites.

We also study the temporal redundancy for individual users. We consider the top user in data sets, and choose a representative 15-day period. Starting from the second day, for each day we treat the previous day' data as the base data (i.e., memory), and compress the particular day's data by eliminating the inter-redundancy between the base data and the particular day's data. We observe that *there are considerable inter-redundancy across time for individual users*, and in certain days it can be 75%.

C. Summary

Traffic redundancy occur in multiple dimensions, and users' actually transmitted data sizes can be reduced by eliminating redundancy. Reduced data size will result in improved network performance including increased throughput and lower response time. This is particularly true for wireless networks, since the wireless media is often shared by multiple users, and data transmission is subject to various collision scenarios where smaller packet sizes are preferred.

III. WIRELESS MEMORY: CONCEPT AND BASIC COMPONENTS

For the sake of easier presentation, the design Wireless Memory (WM) is split into two parts: basic components and advanced component. Basic components contain the essential parts of WM, while advanced components enhance Wireless Memory to deliver improved performance.

A. Concept

Though many elements of WM can be applied to any wireless data networks, the basic network model we consider is the popular Wi-Fi networks with APs and mobile clients. In the following we will use Wi-Fi to describe WM. The overall benefits of using WM are better network delivery performance in terms of higher throughput, lower response time, and higher network utilization levels, through the exploitation of redundancy that naturally exists in wireless traffic.

With the basic network model, WM works between AP and clients. The simplest fashion in which the wireless memory works is as shown in Figure 2. Consider a scenario where there is an AP and a client. Initially both the AP (denoted by S) and the client (denoted by C) have empty memory at time T_0. Assume the AP S has to deliver certain data D to the client C at time T_1. That data is memorized by both S and C. Later at time T_2, S sends another information which contains D, S can retrieve D from the C's memory by sending D's reference (i.e., d) to the client. The reference d sent from S to C is generally much smaller than the raw data D. For the data that are not available in memory, S sends them directly.

B. Basic design elements

The basic elements of WM are illustrated in Figure 3. WM maintains memory space on both AP and clients. When AP communicates with multiple clients, it maintains separate memory for each of the clients. For any AP-client pair, their memories are synchronized in the sense that they contain identical data. The synchronization is achieved implicitly as both AP and the client see identical data being transmitted and received. In addition, identical memory operations such as data referencing and replacement will be performed.

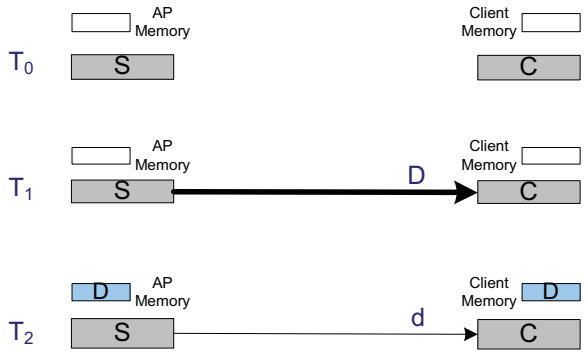


Figure 2. **Concept of WM**

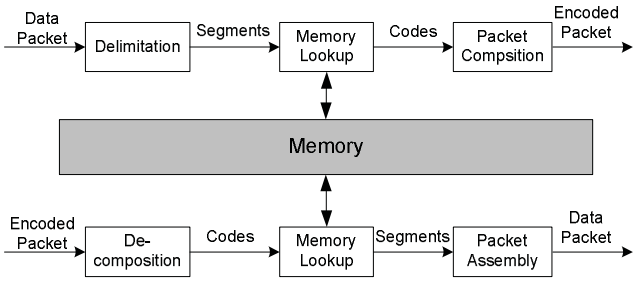


Figure 3. **Basic component of WM**

WM works at packet-level and has two types of operations: *Memory Referencing* and *Memory De-referencing* to encode and decode the data packets, respectively. Their pseudo-codes are shown in Figure 4. Specifically, (i) Memory Referencing sequentially invokes three components of Delimitation, Memory Lookup and Packet Composition. When WM receives a data packet, it firstly delimitates (i.e., splits) the data payload into a sequence of data segments (i.e., data pieces). For each segment, it performs memory lookup to determine whether the segment is in memory or not. If present, the segment will be replaced with a code, and the code can be simply the index of the corresponding memory entry. If not present, then the segment is left as-is. After all segments are processed, a new packet will be composed containing both raw segments and codes. (ii) Memory De-referencing contains the *complementary* components of Packet De-composition, Memory Lookup and Packet Assembly. When a coded packet is received, WM firstly separates the raw segments and codes. For each code, it performs memory lookup to recover the corresponding segment. When all segments are recovered, it assembles them to form the original data packet.

C. Other Considerations

There are some other considerations regarding the aforementioned operations which we briefly describe below. (i) The Delimitation component is based on Rabin-based delimiters [10], which has been shown to have many advantages over fixed delimiters when used to identify redundancy [1]. (ii) Packet Composition results in a coded packet which consists of two regions. The first region is the data region, which contains all segments that cannot be found in memory. The second region is the code region, which consists of a list of <*code, offset*> entries. Each entry represents a redundant segment with the code and the starting offset in the data region. (iii) When Packet De-composition receives a coded packet, it can insert the recovered segment back to the data region to recover the original packet. (iv) When a WM-enabled client firstly associates to a WM-enabled AP, they exchange certain WM-related information to initialize and synchronize WM operations. These information include the memory space size and delimitation parameters. (v) Though memories maintained on both sides are designed to enforce synchronization, when errors do occur (e.g., a reference being unable to decode), WM will report error back to the sender and the sender will retransmit the original packet.

Variables
- P: Current packet
- Set_P: Segment set of P
- Set_C: Code set of P

Received a packet P:
If P is an outgoing packet
- Delimitate P into segment set of Set_P
- For each segment S in Set_P
  - Do memory-lookup
  - If Cache-hit
    - Reference the segment with code
    - Update the corresponding memory entry
  - Else (// Cache miss)
    - Create and enqueue the memory entry

Else (// Incoming packet)
- Extract code set Set_C
- For each code C in Set_C
  - Do memory-lookup
  - If found in memory
    - De-referencing the code
    - Update the memory entry
  - Else (// Cache miss)
    - Report error back to sender

Figure 4. Pseudo code for Basic WM Elements

D. Advanced Element: MFE

The aforementioned basic elements are enhanced by an advanced element of Memory Fidelity Enhancement (MFE), which allows clients to eliminate user-user redundancy by overhearing other clients' traffic. MFE can function independently and can be selectively loaded, thus allowing module-based structure to cap the processing overhead and in turn the response time. For clarification, from now on, we will use WM to refer to the complete design of WM, while refer to the basic elements of WM as Memory Referencing and De-referencing (MRD). WM is designed to be application transparent, meaning no application needs to be changed, and WM can improve the performance of all applications. We assume a design residing at layer-2.5 between the Network layer and Link layer.

Since inter-user redundancy exists, clients can exploit such redundancy to reduce the sizes of their own data. Memory Fidelity Enhancement (MFE) is designed for this purpose. To learn about other users' traffic, a client should explicitly sniff network traffic. Such sniffing is a trivial task in non-encrypted networks, as all data are transmitted openly. For encrypted traffics such as WEP-based, a client can still easily decode other users' raw data.

After a client learns other users' traffic, in order to encode his traffic with AP, AP needs to know what the particular

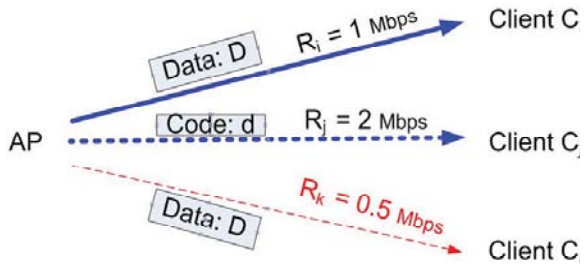


Figure 5. **Illustration of Memory Fidelity Enhancement**

client has overheard. Though a straightforward solution is to ask clients to acknowledge the packets overheard, the additional traffic associated with such acknowledgement makes the approach prohibitive. Instead, MFE allows the AP to "intelligently" estimate what clients overhear which clients' downloading traffic. Specifically, each client C_i has an associated data rate R_i which is determined solely by the channel situation. Since clients can always overhear and decode traffics that are sent with lower rates, a client C_j can learn another client C_k's traffic provided that $R_j \geq R_k$. Since AP knows each client's rate, it can estimate the overhearing capability of each client at any time. One exemplary scenario is shown in Figure 5, where AP has three clients of C_i, C_j and C_k. Assuming their respective data rates are 1Mbps, 2Mpbs and 0.5Mbps. When AP sends data D to C_i, C_j is able to decode since $R_j > R_i$ and later transmission of D can be replaced by its reference d. On the other hand, since $R_i > R_k$, C_k is unable to take advantage of this. The advantage of such technique is that it eliminates the necessity of explicitly verifying the overhearing results.

E. Overall Process on AP and Clients

The operations of MRD and MFE *complement* each other on both sides. We now briefly describe the overall operations on both sides. For simplicity, we only consider the downlink traffic, with AP sending and clients receiving. The processing of the other direction of traffic only differs slightly. (i) When AP receives a packet, the packet is delimited into segments with MRD and replaced with codes when possible. (ii) Then MFE will determine whether other clients are able to overhear this packet. If another client can overhear the packet, the corresponding segments will be put into the client's memory. (iii) When clients receive a encoded packet from AP, MRD extracts the codes and decodes them back to original segments. MFE requires clients to actively overhear other clients' traffic whenever possible, and the overheard data will be put into its memory.

IV. Evaluation

We evaluate WM with trace-driven simulations, and the traces are described in Section II. The direct result of applying WM is the reduced data size, which in many network environments translates to higher throughput. Thus, the performance metrics we consider are the resultant data size and the network throughput with a typical network setup. The simulation software we use is NS2, with which an 802.11 Wi-Fi network is configured. We integrate the seven components of WM inside NS2 so that the input traces can be processed by WM before being transmitted by NS2.

We compare WM to a baseline scenarios where data packets are sent as-is. For WM, we evaluate each of the 4 components separately, as well as the integrated solution. Except the baseline scenario, we assume all clients are WM-enabled, unless otherwise stated. To ensure consistency of various evaluations, we always use the following network setup. The 802.11 network consists of a AP and 8 wireless clients with random placement. Each of the clients sets up a single TCP connection with another fixed host behind AP. The RTT of the wired network is 60ms and bandwidth is 100Mbps. Some other important parameters about the 802.11 setup are: Slot-Time 20us, SIFS 10us, Preamble-Length 72 bytes, Data-Rate 11Mbps, and no RTC/CTS.

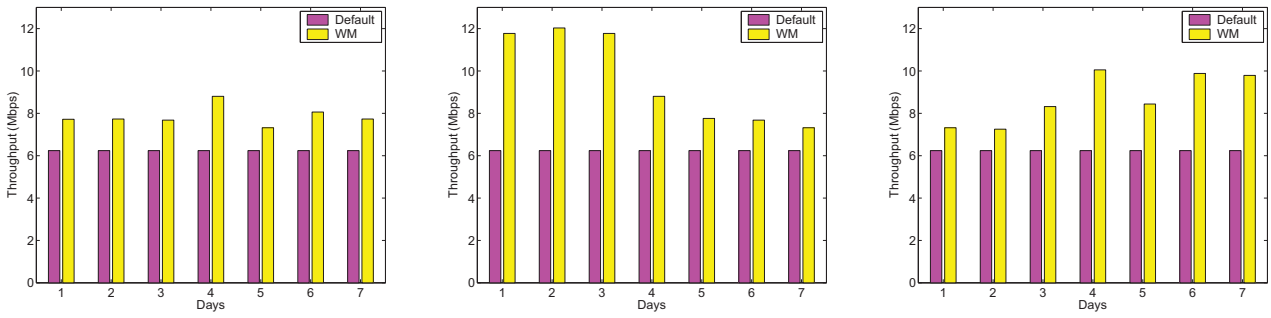
With our collected traces, we evaluate WM along the following dimensions. First, we choose the same data sets as used in Section II (i.e., three buildings) and study the aggregate network throughput. Second, we study the impact of redundancy level on both coded packet size and aggregate throughput. Since our data set is very diversified in terms of varying redundancy degrees, we choose three typical users who have comparatively low, medium and high redundancy, respectively.

A. Aggregate network throughput

We use the same data sets as described in Section II-B. For each data set, we choose the top 8 users based on traffic volumes and use their traffics as the simulation inputs. The results are shown in Figure 6. We observe that the aggregate throughput for the baseline scenario is about 6.24 Mbps. For all data sets and different days, the improved throughput vary between 7.25 Mbps and 12.03 Mbps, or an improvement between 16% and 93%. The average improvement is about 40%. Note that the throughput results are the *effective* throughput as experienced by applications rather than raw throughput. Since WM can significantly reduce packet size by eliminating traffic redundancy, the effective throughput can be larger than the physical bandwidth limit of 11 Mbps.

B. Impact of redundancy level

We now examine the impact of redundancy level on both coded packet size and aggregate throughput. We choose three representative users that exhibit different levels of redundancy and use their respective data as the inputs of all clients. The redundancy level is estimated based on the averaged packet sizes when coded by MRD. Specifically, the three users have about 10%, 35% and 60% redundancy, respectively.



(a) Building A (b) Building B (c) Building C

Figure 6. **Aggregate network throughput based on three data sets**

1) Low redundancy: We observe that though MRD can effectively reduce the data size by 11%, complete solution of WM can achieve 17% of reduction. We also see that WM improves the default throughput of 6.24 Mbps by more than 11%, or 6.90 Mbps.

2) Medium redundancy: For medium-redundancy, we observe that WM can achieve more than 40% of data size reduction. The throughput improvement is about 32% (i.e., 8.22 Mbps vs. 6.24 Mbps).

3) High redundancy: In high-redundancy scenarios, we observe that WM can reduce the packet size by about 62%, and the throughput improvement is about 57% (i.e., 9.79 Mbps vs. 6.24 Mbps).

V. Related Works

Primarily motivated by the temporal dimension of traffic redundancy on Internet, several approaches are proposed to exploit such redundancy and reduce users' response time. Squirrel [11] provides a decentralized, peer-to-peer web cache by enabling web browsers on desktop machines to share their local caches and form an efficient and scalable web cache. [12] develops a novel caching algorithm for P2P traffic. These caching are performed on file-level, which significantly limit their effectiveness.

Various approaches are also proposed to eliminate traffic redundancy at finer granularity than packet-level. A value-based web caching [1] is motivated by the facts that web files may be changed gradually and aliased, and proposes to split files into blocks. Also, a protocol-independent technique [3] proposes a mechanism to detect repetitive traffic on a communication link and provides a protocol-independent idea to eliminate the repetitive segments. [4] uses digests for packets to directly suppress redundant transfers in networks by using a proxy on either end of a low bandwidth connection. Work [6] proposes to deploy packet-level memories on Internet routers and change routing protocols to explicitly remove redundancy, and Work [7] further presents redundancy-elimination design as a network-wide service.

VI. Conclusion

In this work, we investigate the traffic redundancy problem in wireless networks. We propose a solution suite called Wireless Memory which can help deliver better performance by eliminating redundancy.

References

[1] S. C. Rhea, K. Liang, and E. Brewer, "Value-based web caching," in *Proceedings of WWW '03*, Budapest, Hungary, 2003.

[2] J. C. Mogul, Y. M. Chan, and T. Kelly, "Design, implementation, and evaluation of duplicate transfer detection in http," in *Proceedings of NSDI'04*, San Francisco, CA, 2004.

[3] N. T. Spring and D. Wetherall, "A protocol-independent technique for eliminating redundant network traffic," *SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, pp. 87–95, 2000.

[4] J. Santos and D. Wetherall, "Increasing effective link bandwidth by suppressing replicated data," in *Proceedings of ATEC '98*, New Orleans, LA, USA, 1998.

[5] G. Pallis and A. Vakali, "Insight and perspectives for content delivery networks," *Commun. ACM*, vol. 49, no. 1, pp. 101–106, 2006.

[6] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, "Packet caches on routers: the implications of universal redundant traffic elimination," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 219–230, 2008.

[7] A. Anand, V. Sekar, and A. Akella, "Smartre: an architecture for coordinated network-wide redundancy elimination," in *Proceedings of ACM SIGCOMM '09*, Barcelona, Spain, 2009.

[8] J. Ziv and A. Lempel, "Compression of individual sequences via variable-rate coding," *Information Theory, IEEE Transactions on*, vol. 24, no. 5, pp. 530–536, Sep 1978.

[9] "Rzip," http://rzip.samba.org/.

[10] M. O. Rabin, "Fingerprinting by random polynomials," *Technical Report TR-15-81,Center for Research in Computer Technology*, 1981.

[11] S. Iyer, A. Rowstron, and P. Druschel, "Squirrel: a decentralized peer-to-peer web cache," in *Proceedings of PODC '02*, Monterey, CA, 2002.

[12] O. Saleh and M. Hefeeda, "Modeling and caching of peer-to-peer traffic," in *Proceedings of ICNP '06*, Washington, DC, 2006.