Prediction-based Redundant Data Elimination with Content Overhearing in Wireless Networks

Haiying Shen[◇], Shenghua He[*], Lei Yu[†] and Ankur Sarker[◇]
[◇]Department of Computer Science, University of Virginia, USA
[*]Department of Computer Science and Engineering, Washington University in St. Louis, USA
[†]College of Computing, Georgia Institute of Technology, USA
[◇]{hs6ms,as4mz}@virginia.edu, [*]shenghuahe@wustl.edu, [†]leiyu@gatech.edu

***Abstract*—This paper aims to improve wireless network throughput by suppressing duplicate data transmissions from network links. It has been demonstrated that IP-layer Redundancy Elimination (RE) with content overhearing can significantly improve the goodput and utilization of wireless channels in wireless environment. However, the integration of IP-layer RE and wireless overhearing introduces a challenge. That is, probabilistic wireless overhearing and the possibility of a receiver overhearing from multiple transmitters cause the caches of a sender and a receiver far from synchronization, which can disrupt IP-layer RE's correctness and degrade its performance. The previous work deals with this challenge by the overhearing probability estimation, which however is not efficient or scalable. In this paper, we propose a Prediction-based Redundancy Elimination with Content Overhearing method (PRECO) to address this challenge. By exploiting prediction-based RE, PRECO does not require cache synchronization and overhearing probability estimation, which enables its efficient and scalable deployment. Based on PRECO, we exploit the benefits of deploying sub-packet level RE as a primitive IP-layer service on all nodes in wireless mesh networks by proposing a redundancy-aware routing protocol. Trace-driven performance evaluation shows the effectiveness and efficiency of PRECO compared with other RE methods.**

***Index Terms*—Redundant Data Elimination; Data Cache; Data Prediction and Redundancy; Wireless Signal Overhearing;**

I. Introduction

Wireless networks have been a widely used communication paradigm for providing mobility, city-wide Internet connectivity and outdoor computing with low cost and fast deployment [1], [2], [3], [4], [5], [6]. However, interference and poor link quality severely limit the throughput of wireless networks especially for dense large networks [7], [8]. This paper focuses on an important class of techniques [9], [10], [11], [12] that aim to improve wireless network throughput by suppressing duplicate data transmissions from network links. These techniques can eliminate the transmissions of packets or data contents that have been previously transmitted. Accordingly, they can be classified to two categories: packet-based Redundancy Elimination (RE) [9] and content-based RE [10], [11], [12], [13]. Compared with packet-based RE, content-based RE can exploit the data locality in the workload transferred over wireless networks. The data locality is resulted from the user accesses to the same popular content on the Internet and also the prevalent similarity among different Internet data objects [10].

Recent studies have shown that vast majority of traffic redundancy in Internet arises from duplicate data chunks of size less than 150 bytes [14]. Accordingly, IP-layer RE at a sub-packet level has been shown to provide significant performance benefits in wired networks. It makes use of tightly synchronized caches between the sender and the receiver [15], [16], [17], [18]. The sender removes the duplicate byte string (as small as 64B) from the packet payload by comparing it against prior transmitted packets, and inserts a shim instead; the receiver replaces the shim by the corresponding byte string in prior received packets to reconstruct the full packet.

Exploiting the IP-layer RE technique with content overhearing in wireless networks can significantly improve the goodput and utilization of wireless channels [12]. However, it introduces a challenge. Probabilistic wireless overhearing and the possibility of a receiver overhearing from multiple transmitters cause the caches of a sender and a receiver far from synchronization, which can disrupt RE's correctness and degrade its performance. REfactor [12] deals with the challenge by overhearing probability estimation. Considering a single access point (AP) infrastructure with multiple associated clients, the sender AP estimates whether the receiving client is likely to have overheard an outgoing chunk from previous transmissions to some other clients. Based on that, the sender computes the expected reduction of transmission time resulting from the removal of the chunk, in consideration of possible additional time for missing data request and retransmission in case of cache miss at the receiver. AP removes the chunk only when the expected reduction of transmission time is high.

Wireless overhearing probability, however, is difficult to estimate accurately. REfactor determines the overhearing probability merely by the data transmission rate, which is simple but very likely to be insufficiently accurate because of dynamic channel conditions and interference in dense large wireless networks. Insufficiently accurate estimation can cause wrong decisions on whether or not to remove redundancy, and consequently degrade the performance of RE. Moreover, to extend to multiple AP infrastructures and mesh networks, REfactor requires wireless nodes to periodically send overhearing notifications and communicate cache contents, which incurs significant communication cost and complex coordinations among nodes. Similarly, to eliminate traffic redundancy over a hop in wireless mesh networks, a node must collect such cache information from all other nodes of which the traffic can be overheard by the next-hop node.

Instead of using overhearing probability estimation, this paper addresses the challenge of wireless content overhearing for IP-layer RE using *predictions*. Accordingly, we propose

Prediction-based Redundancy Elimination with Content Overhearing (PRECO). In PRECO, the receiver stores the received and overheard data stream in a chain of chunks. It compares the chunks of the incoming packet with the chunk chains in the cache. Upon a match, it is expected that the future incoming data is very likely to match with the previously stored chunks on the chain. The receiver sends to the sender future data predictions that include the hashes of chunks on the chain. The sender removes the chunk of the outgoing packet if it finds that its hash matches with a prediction. In this way, PRECO does not rely on overhearing probability estimation; it removes a duplicate chunk only if the receiver has cached the chunk. It ensures effective and robust content-overhearing based RE over wireless links. It can also be directly used in multiple AP infrastructure and mesh networks without overhearing notifications and cache content communication.

PRECO has fundamentally different challenges from previous Prediction-based RE approach called PACK [19] proposed for the cloud environment. PACK uses a large chunk size of 8KB which causes it to fail to identify finer-granularity content redundancy. It also leads to ineffective content overhearing since many nodes may not overhear such large chunks in full in the wireless environment. In contrast, PRECO aims to identify duplicate chunks of hundreds of bytes at a sub-packet level and work on lower-bandwidth wireless links. Thus, the transmission cost of predictions has to be considered in order to realize the overall benefits of RE. Another new issue in PRECO is the possibility of missing data in chunk chains. PACK makes predictions and matches not only on the received TCP streams but also on the overheard data streams. Therefore, PRECO must be able to identify the stream among the multiple streams containing matching chunks that lead to more accurate prediction in spite of missing data in some overhead data streams.

Furthermore, we exploit the benefits of deploying sub-packet level RE as a primitive IP-layer service on all nodes in wireless mesh networks. Such network-wide deployment can provide performance benefits by eliminating redundancy on every link. Thanks to PRECO enabling efficient network-wide deployment in a wireless environment, we further propose redundancy-aware source routing in wireless mesh networks in order to derive greater performance gains from network-wide RE. We introduce a "redundancy-aware estimated transmission time" (RETT) metric. RETT predicts the total amount of time it would take to send a data packet along a route by taking into account link-level RE. When forwarding traffic from the Internet to the mesh network, a gateway chooses the route with the lowest RETT.

In summary, the contribution of this work is as follows.

- PRECO provides an effective, efficient and scalable solution for content-overhearing based IP-layer RE over wireless links.
- A redundancy-aware routing algorithm is proposed to further exploit the benefits of network-wide RE deployment in wireless mesh networks.

The remainder of this paper is organized as follows. Section II reviews existing schemes for redundancy elimination in wired and wireless networks. In Section III and IV, we present the design of PRECO and redundancy-aware routing algorithm, respectively. Section V presents performance evaluation. Section VI concludes this paper.

II. Related Work

A. *Traffic Redundancy Elimination in Wired Networks*

Several Traffic Redundancy Elimination (TRE) techniques have been proposed for wired networks in recent years. A protocol-independent TRE approach was first proposed in [20], which identifies duplicate chunks at sub-packet level with content-based hashing. Several commercial vendors have developed such techniques into their "WAN optimization" middle-boxes [21], [22], [23]. The successful deployment of TRE solutions in enterprise networks motivated the exploration of TRE deployment across the entire Internet, and redundancy-aware routing was proposed to further enhance the benefits of network-wide TRE [15], [18]. EndRE was proposed [17] for eliminating traffic redundancy from server to client. The RE method used in [16] eliminates the traffic redundancy with the approach of delta compression. All of these approaches above make use of tightly synchronized caches or the same reference file between the sender and receiver, so they cannot be used with overhearing for RE in wireless networks. PACK [19] was proposed for the cloud environment [24]. It introduces prediction-based RE, in which the receiver compares the incoming data with previously received chunk chains and notifies the sender the matched chunks. In this way, PACK does not require cache synchronization between the sender and receiver. However, PACK uses a large chunk size of 8KB. This causes it to fail to identify finer-granularity content redundancy. It also leads to ineffective content overhearing since many nodes may not overhear such large chunks in full in the wireless environment. Yu *et al.* [25] proposed cooperative end-to-end traffic redundancy elimination in both sides of the sender and receiver for both large and small chunks for reducing cloud bandwidth cost.

B. *TRE in Wireless Networks*

In Asymmetric Caching (AC) [11], the receiver finds in its cache the matched flow segment which has the maximum number of matching chunks compared with the ongoing traffic flow it received, and sends back their chunk hashes. The sender then performs RE operations based on its feedback cache storing the received hashes and regular cache. SmartEye [26] aggregates similar images into the same group via a semantic hashing in the cloud and then eliminate the redundancy in the data transmission for images collected from the smart terminates similar as existing images in the cloud. However, these methods fail to leverage overhearing. Several methods have been proposed to leverage wireless overhearing to eliminate redundant data transmissions. In RTS-id [9], the receiver caches the overheard packets, and the sender adds a special ID to the 802.11 RTS packet so that the receiver can check if the data packet to be transmitted is in its cache. Ditto [10] and REfactor [12] are two content-based RE approaches. In Ditto, wireless mesh routers reassemble overheard TCP streams from the server to the client to reconstruct application data chunks of size roughly 8KB, and store chunks into their caches. It

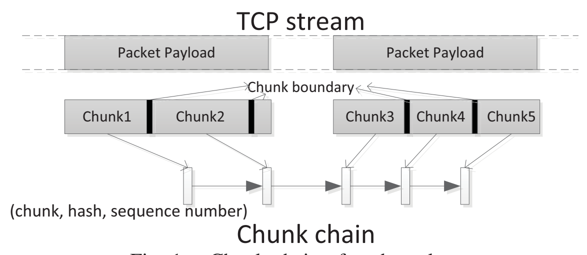
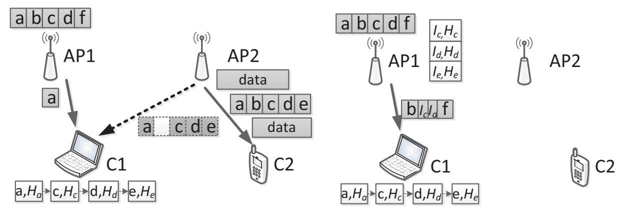


Fig. 1. Chunk chain of each packet.



(a) Overhearing and caching (b) Prediction and RE
Fig. 2. PRECO in multiple AP infrastructure.

avoids data transfers by serving the requests of clients from the mesh routers rather than the server. REfactor exploits finer-granularity redundancy at the sub-packet level by IP-layer RE with content overhearing. The sender estimates the overhearing probability of data for the receiver, and removes duplicate chunks only if the expected transmission time reduction resulting from the redundancy removal exceeds some threshold. However, the performance gain of REfactor is vulnerable to inaccurate estimation of overhearing probability, and also it incurs significant communication cost when being extended to multiple AP infrastructure and mesh networks.

III. Prediction-Based RE With Overhearing

A. Overview and An Example

In PRECO, a wireless node parses the payload of a received or overheard packet into chunks and computes the hash value for every chunk. The chunks from the same data stream are linked sequentially into a chain and stored with their hash values in a local chunk cache (Figure 1). When a receiver receives or overhears a packet from a sender, it compares the chunks of the packet to its local chunk cache. If a matching chunk is found, the receiver retrieves the sequence of subsequent chunks after the matching chunk and sends their hashes to the sender as a prediction.

The sender performs chunking on the payload of outgoing packets, and matches the chunks against the predictions from the receiver. Once it finds a match, the sender removes the chunk and replaces it with a shim containing a prediction ID to confirm the corresponding chunk prediction. The receiver then replaces the shim with its corresponding chunk in its local chunk cache. PRECO is advantageous than previous RE approaches because the predicted chunks are guaranteed in the receiver's local chunk cache, which increases the prediction accuracy and hence the transmission throughput. Further, PRECO brings benefits and efficiency in multiple AP infrastructure, as explained below.

In Figure 2, client $C1$ can overhear the data transmission from AP2 to Client $C2$. For successive chunks a,b,c,d and e from multiple packet payloads in a data stream transferred from AP2 to $C2$, $C1$ successfully overheard chunks a,c,d and e, and cached them in a chain with their hashes H_a,H_c,H_d and H_e. Note that chunk b is missing due to the probabilistic wireless overhearing. AP1 later on had a similar data object with chunks a,b,c,d and f to send to $C1$. When $C1$ received chunk a, it found that a had a matching chunk on the chain, and then sent to AP1 the chunk predictions that include hashes of subsequent chunks c, d and e on the chain. In the outgoing data, AP1 found that the chunks c and d have matching hashes with the predictions, and replaced them with the shims "I_c" and "I_d". $C1$ can reconstruct the full packet using the cached chunks corresponding to the predictions. Since the sizes of prediction and shim are far less than the chunk size, PRECO reduces the number of bytes transferred in the air and improves overall network throughput. In contrast, REfactor [12] requires overhearing notification communication between AP1 and AP2. AP1 needs to periodically obtain AP2's cache content for $C1$, which contains entries and estimated overhearing probabilities of all chunks presumably overheard by $C1$, which cost a mount of network overhead.

PRECO needs to resolve the following issues:

- How to conduct chunking and caching for efficient redundant data detection in data transmission? (Section III-B)
- How to choose a stream and a matching chunk for prediction to increase the prediction accuracy? (Section III-C)

B. Chunking and Caching

1) Chunking Algorithm: PRECO divides the payload of a packet into chunks by a content-based chunking algorithm. This algorithm determines the chunk boundaries using content instead of offset, so localized changes in the data stream only affect chunks that are near the changes, which enables efficient and robust duplicate content identification across different data objects. A number of content-based chunking algorithm have been proposed, including Rabin fingerprinting [20], [10], MAXP [14], SAMPLEBYTE [17], and XOR-based rolling hash [19]. Like REfactor [12], PRECO uses MAXP [14] to define chunk boundaries, because MAXP provides uniformly distributed chunk boundaries across the payload and imposes a lower bound on chunk length and low computational overhead. MAXP selects a position as chunk boundary if its hash value is the maximum (or minimum) over all hashes computed over the p-byte region centered at that position. The packet payload is divided into chunks by these boundaries, as shown in Figure 1. The expected chunk size is p and all chunks must have length at least $\lfloor p/2 \rfloor$ except the last one at the end of payload [27]. We ignore the last chunk if its size is less than $\lfloor p/2 \rfloor$ such as the chunk following Chunk2 in Figure 1. p should be significantly larger than the sum size of a chunk hash and shim in PRECO because the effective bandwidth saving resulted from a successful chunk prediction is the chunk size minus the sum size. Addtionally, we also limit the maximum length of a chunk to B_{max} bytes. The determination of chunk size should consider the trade-off between prediction overhead and bandwidth savings. A larger chunk size reduces the number of prediction messages, while a smaller chunk size can increase the detection efficiency of redundant bytes.

2) Caching Received and Overheard Chunks: In PRECO, wireless nodes overhear the transmissions of TCP streams. PRECO identifies each stream by (src, dst, src port, dst port) tuple, referred to as *stream ID*. A wireless node maintains a stream list to record the IDs of TCP streams, which it is currently receiving and overhearing. When a new packet arrives, the node checks the list to determine whether the packet is from a new stream or an existing stream. If the packet is from an existing stream, its chunks are linked with previously overheard chunks in the same stream. Otherwise, the node creates an entry for the new stream in the list and caches the chunks of the packet. A timeout period is specified for each entry, and is reset every time when a new packet is received from the stream of that entry. It ensures the transmission time locality of chunks in the same chain. An entry is deleted if no packets from the corresponding stream are overheard or received within the timeout period.

Recall that a stream consists of packets, a packet consists of chunks, and a chunk consists of data bytes. The wireless nodes store the overheard/received chunks in their local chunk caches along with the hashes and sequence numbers of the chunks. A chunk's sequence number is the sequence number of its containing TCP packet plus the offset of the chunk in the packet payload. The chunks of packets from the same TCP stream are linked into a chain in the order of their sequence numbers, as shown in Figure 1. Also, any byte in the chunk has a sequence number which is the chunk's sequence number plus the offset of the byte in the chunk. The chunks on the chain may not be consecutive data in the stream, because of missing data due to the probabilistic overhearing and ignored chunks by the chunking algorithm. Still, the node in a multi-hop wireless network can have a chance to fill the missing data incurred by the probabilistic overhearing, since it may overhear the same packet over different hops.

The wireless nodes may receive/overhear the same packet multiple times, either incurred by retransmission mechanisms in 802.11 MAC and TCP protocols, or because the nodes in multiple-hop networks can overhear the transmission of a packet over different hops. To detect duplicate packets, PRECO stores a packet's TCP sequence number along with its chunks in the cache. PRECO considers a packet to be duplicate if it has repeated TCP sequence number in the stream. By ignoring duplicate packets, PRECO ensures that only a single copy of the packet is chunked and stored, and thus avoids unnecessary processing and storage cost.

C. Prediction-Based Redundancy Elimination

1) Prediction Algorithm: Upon receiving a new packet from the sender, the receiver performs chunking and computes the hash for each chunk in the payload, and then looks up these hashes in its local chunk cache. If a chunk's hash is found, it means that a duplicate chunk exists in the cache. Based on the chains which contain the matching chunks, the receiver finds the chunks which would mostly likely appear in the next incoming data, and sends to the sender their hashes as predictions.

Below, we explain how to find the chunks as predictions. The previous approach PACK [19] makes predictions for every single chunk match because new chunks arrive sequentially in TCP stream. Every time when a matching chunk is found, it retrieves a sequence of chunks subsequent to the matching chunk on the chain for predictions. Such a prediction algorithm, however, is not efficient for PRECO. This is because in PRECO, a packet may have multiple chunks that have duplications in the cache. If the prediction is made for every chunk match, multiple sequences of chunks, each following a matching chunk, would be retrieved for predicting the same incoming data, which incurs extra prediction transmissions. Moreover, these matching chunks may be scattered over different chains, but some chains may not have much in common with the data object in transmission, and predictions from those chains are useless. To address these issues, PRECO determines one matching chunk, which leads to the chunk sequence most likely to appear in the future incoming data, and uses the sequence of chunks following it on the chain for prediction. We refer to such a matching chunk as a *prediction anchor*.

To determine the best prediction anchor, we first find the chain that has "maximum matching" with the received chunks and then decide which matching chunk in it is used as the prediction anchor. For this purpose, a straightforward method is to choose the chain which contains the largest total size of matching chunks, and use the largest one among all the matching chunks on that chain as the prediction anchor. In detail, denote the chains by $L_1,...,L_k$. Suppose each chain L_i has matching chunks $\{C_{i,1}, C_{i,2}, ..., C_{i,l_i}\}$. Let $|C_{i,m}|$ be the size of chunk C_{im}. The total size of matching chunks in chain L_i is $S_i = \sum_{m=1}^{l_i} |C_{i,m}|$. Suppose $S_n = \max\{S_i | 1 \leq i \leq k\}$, then the chain L_n is selected, and the matching chunk with size of $\max\{|C_{n,i}| 1 \leq i \leq l_n\}$ is used for prediction. This method, however, may not generate efficient predictions since it does not consider the distances between the matching chunks in TCP stream. The effectiveness of prediction-based RE comes from the continuity of duplicate content. If the matching chunks are loosely scattered on the chain with large gaps between each other, the chain is not well matched with the data object in transmission and can provide poor predictions. Thus, we introduce a chunk-merging approach to find the "maximum matching" chain with the consideration of the distances between matching chunks.

Chunk-merging based prediction Suppose $C_{i,m}$ and $C_{i,m+1}$ are two neighboring matching chunks on the same chain L_i and $C_{i,m}$ is before $C_{i,m+1}$ in terms of receiving order. We define the distance between two chunks $C_{i,m}$ and $C_{i,m+1}$ (denoted by d) as the difference between the sequence number of the last byte of $C_{i,m}$ and the first byte of $C_{i,m+1}$. We set a threshold d_T for the distance. When d reaches d_T, we can ignore the connection between $C_{i,m}$ and $C_{i,m+1}$ and only consider the larger chunk. If the distance d is less than a threshold d_T, we merge them into a virtual chunk $C'_{m,m+1}$ with a virtual size computed by

$$|C'_{m,m+1}| = (1 - \frac{d}{d_T})(|C_{i,m}| + |C_{i,m+1}|) + \frac{d}{d_T} \max\{|C_{i,m}|, |C_{i,m+1}|\} \quad (1)$$

Here the virtual chunk size actually is a measure of matching degree. As we can see, when $d \to 0$, $|C'_{m,m+1}| \to |C_{i,m}|+|C_{i,m+1}|$. That is, when the distance between two chunks (C_m and C_{m+1})

decreases, the matching degree between the chain and these two chunks gets close to the matching degree achieved by a duplicate chunk of size $|C_{i,m}| + |C_{i,m+1}|$. Similarly, when $d \to d_T$, the matching degree is close to that achieved by the larger matching chunk, which means the combination of two chunks does not have additional effect on the matching degree anymore when their distance becomes large enough.

Assume a list of matching chunks of a packet $\{C_{i,1}, C_{i,2}, C_{i,3}, ...\}$ on a chain are numbered in the order of their sequence numbers. We need to calculate the virtual size of their merged virtual chunk. The merging is conducted iteratively from $C_{i,1}$ in sequence until no two neighboring chunks have a distance less than d_T. If $C_{i,1}$ and $C_{i,2}$ can be merged (i.e., their distance is less than d_T), they are replaced with the merged virtual chunk $C'_{1,2}$ in the list, and the merging iteratively starts from $C'_{1,2}$. The distance between $C'_{1,2}$ and $C_{i,3}$ is equal to the distance between $C_{i,2}$ and $C_{i,3}$. If $C_{i,1}$ and $C_{i,2}$ cannot be merged, $C_{i,1}$ is ignored and the merging iteratively starts from $C_{i,2}$.

For a received packet, it may have matching chunks in several chains in the receiver's local cache. For each of these chains, PRECO calculates the virtual size of the merged virtual chunk using the method introduced above. It then chooses the largest chunk as the prediction anchor among unmerged matching chunks and virtual chunks, and uses the chunks subsequent to it on the chain for predictions.

Our chunk-merging based prediction enables to utilize multiple packets to improve the prediction accuracy. Instead of attempting to make predictions at every time of receiving a packet, the receiver can accumulate multiple packets from a data stream before making predictions. With more chunks from multiple packets, the chunk-merging approach can better identify the concentration place of the matching chunks, which indicates the chain matched best with the data stream being received and accordingly the best position for choosing the prediction anchor.

2) Prediction Transmission and Shim Decoding: For the effectiveness of predictions, PRECO limits the data range predicted from a prediction anchor by setting a prediction window W. In the chunk sequence following the prediction anchor, a chunk can be used for prediction only if the distance between it and the prediction anchor is less than W. The receiver retrieves the chunk hashes within the prediction window, and uses them for chunk predictions. Each chunk also is assigned with a prediction ID which is its sequence number in the window. A chunk prediction includes a hash for the expected incoming chunk. The receiver sends the chunk predictions to the sender in a prediction message. The chunk predictions in the same prediction message form a sequence. The receiver also maintains a prediction cache for storing predictions recently sent to the sender.

When receiving a prediction message from the receiver, the sender extracts the chunk predictions and stores them into a local prediction cache. It proceeds to compare the predictions with its outgoing data. For each outgoing packet, the sender performs the chunking algorithm (in Section III-B1) to divide the payload into chunks, and tries to match the predictions with these chunks. If it finds that a chunk matches with a prediction, the sender replaces the content and insert a shim instead, including the offset of the chunk in the packet and the sequence number of the chunk in the prediction message. Once receiving a packet containing a shim from the sender, the receiver finds the chunk corresponding to the sequence number in the shim, and reconstructs the full packet by replacing the shim with the chunk.

IV. Redundancy-Aware Source Routing In Wireless Mesh Networks

In wireless mesh networks, every mesh router shares Internet access by communicating with a few gateway nodes and provides Internet connectivity to mobile clients. It is very promising to improve the throughput of wireless mesh networks by eliminating the redundancy in the traffic transferred from the gateways to mesh routers. In this section, we consider to deploy PRECO as a primitive IP-layer service on all nodes in wireless mesh networks, and exploit the benefits of routing to maximize the opportunity to reduce redundant content.

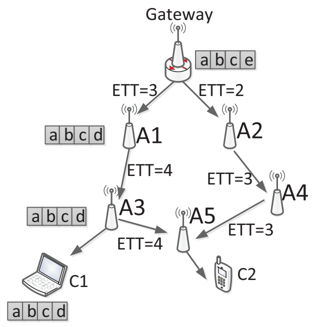


Fig. 3. RE with on-path caching.

Figure 3 gives an example to show the benefit of the redundancy-aware source routing. Each link is labeled with its ETT (Expected Transmission Time) metric [28] defined as the expected MAC layer duration for a successful transmission. Client $C1$ requests a data object $D1$ consisting of chunks a, b, c and d, and the gateway chooses the route "Gateway$-A1-A3$" to transfer the data object. Traditional ETT-based source routing protocol chooses the route with the minimum sum of ETT of links on the path. Thus, when $C2$ requests a data object $D2$ consisting of chunks a, b, c and e, the gateway chooses the route "Gateway$-A2-A4-A5$" with ETT metric of 8. Suppose each router on a path caches the transferred data object and is capable of removing redundant content. Then, if the route "Gateway$-A1-A3-A5$" is used for transferring $D2$, the redundant content a, b, c can be removed over links "Gateway$-A1$" and "$A1-A3$", and only e is required to transfer over these links. Suppose the size of each chunk is equal to a packet size, then according to the computation of ETT, the time required to transfer $D2$ over link "Gateway$-A1$" and "$A1-A3$" is 3 and 4 respectively. The total ETT for $D2$ is 3+4+16=23. The traditional ETT-based route "Gateway$-A2-A4-A5$", however, has a total ETT for $D2$ is 8+12+12=32.

A. Redundancy-aware Routing Metric

We propose a "redundancy-aware estimated transmission time" (RETT) metric of a link derived from the ETT metric [28]. ETT is computed by ETX$\times\frac{S}{B}$ where ETX [29] is Expected transmission Count which estimates the number of retransmissions needed to deliver a packet over the link, S is the average packet size and B is the bandwidth. Suppose that a packet has an average redundancy ratio α compared with the packet cache over a link. Then, the expected transmission time for the packet with the consideration of redundancy elimination over the link, i.e., RETT, can be computed by

$$RETT = \text{ETX} \times \frac{S(1-\alpha)}{B} \quad (2)$$

This link metric can more accurately reflect the transmission time for content transmission taking advantage of the redundant content in the routers. However, it also has problems. First, it ignores the overhead of prediction transmissions. Second, the redundancy ratio α varies among packets and it depends on the content of the packets. Third, it is extremely expensive for a routing protocol to compute an optimal route for every single packet. To handle these problems, our solution is to consider optimal routing for a whole TCP stream or large bulk-data in the stream, such that an optimal route can be decided for a group of packets which are most likely to have the similar redundancy profile. For the prediction transmission overhead, since a prediction message contains multiple chunk predictions. The overhead of a prediction message is amortized over all chunk predations in the message. Moreover, each chunk prediction has a much smaller size (eg., 32bit hash + 10bit prediction ID) than the average chunk size (eg., 256byte). Therefore, the prediction overhead is negligible, especially when the data has sufficient redundancy. The RETT metric of a route is the sum of the link metrics. It predicts the total amount of time it would take to send the bulk data along a route, taking into account link-level RE.

B. Routing Protocol

Based on the above discussions, we propose a redundancy-aware source routing protocol for a gateway to forward traffic from the Internet to the mesh network.

For RETT computation, we can use the method in previous work [29] to compute ETX of every link. Each mesh router broadcasts link probes of a fixed size at an average period, and computes the delivery ratio by counting the number of probes received during a time window. Then, the routers report the statistics back to the gateway. To obtain the redundancy ratio for a bulk-data, the gateway stores all the transferred bulk-data along with their routing path information. Using the previously introduced chunking algorithm, the gateway also divides each transferred bulk-data into small chunks, computes the hash for each chunk, and stores them in its chunk cache. When forwarding a new bulk-data, the gateway divides it into chunks and look ups their hashes in the chunk cache to find the bulk-data in the cache which has the most number of duplicate chunks with the new bulk-data, denoted by N_{max}. Suppose the new bulk-data has N number of chunks. Then, the redundancy ratio of the new bulk-data is $\alpha = N_{max}/N$.

Suppose that the route for the bulk-data that has N_{max} number of duplicate chunks consists of nodes n_1, n_2,...,n_k. With PRECO that performs RE based on receiver's prediction, any link with n_i as the receiving node can possibly obtain the benefit of RE. To ensure negligible prediction transmission cost compared with the redundancy to be removed, we can give a threshold α_T. That is, if $N_{max}/N > \alpha_T$, RETT metric is computed by Equation (2), otherwise, it is equal to ETT. After computing the RETT metric of all links, the gateway runs Dijkstra's algorithm to find the route with the lowest RETT.

Democratic Candidates Debate How to Lower Cost of College Education
by ABC News
1,000 views
7:39

Democratic Candidates Debate Gun Control Laws
by ABC News
255 views
8:00

Fig. 4. Two different videos.

In the above, the computation of the link RETT metric in the proposed routing protocol does not consider the effect of overheard content to redundancy elimination of PRECO. It only considers the effect of data cached by the on-path transfer. Some links may not have on-path nodes but they have nodes which overheard the duplicate content. With these overhearing nodes as receiving nodes on links and redundancy elimination service, these links may have smaller expected transmission time than their ETT, and thus can be used to find a better route. However, to compute RETT for these links, the gateway needs to have an accurate estimate of the overhearing probability of nodes, which has been shown to be difficult. Because of this, our routing protocol computes the routes based on that the gateway has the definite routing information of previously transferred bulk data. Meanwhile, the benefit of content overhearing can be opportunistically exploited as introduced previously in our RE method. That is, each receiver predicts the data that will be transmitted to it based on its received and overheard data in its cache for redundancy elimination in data transmission.

V. Performance Evaluation

In this paper, we developed a simulator using Java to conduct real trace driven simulation experiments to evaluate the performance of PRECO. We collected wireless data traces from running the YouTube App on two smartphones, which are iPhone 6 plus and Xiao Mi 3. In order to capture the traffic, we let the traffic go through a laptop (Lenovo T420 with Windows 10) and used the Wireshark software to capture it. Specifically, we connected the laptop to the Internet with the wire cable and set up the Hotspot mode on the laptop. The two smartphones are connected with the laptop using WiFi shared by the laptop. As a result, the two smartphones can access the Internet.

The two smartphones watched the same randomly selected videos for 20 minutes, and then watched two different videos with similar contents for 10 minutes, as shown in Figure 4. This process repeated twice. We collected data 60 minutes a day for 7 days in total. The data trace towards iPhone 6 plus is denoted as $Trace_1$ and the other one is denoted as $Trace_2$. Finally, we got 1.71GB data for $Trace_1$ and 1.64GB for $Trace_2$. We compared PRECO with EndRE [17], Asymmetric Caching (AC) [11] and REfactor [12] using the following metrics:

- RE efficiency. It is the ratio of the total bytes of reduced redundant data to the total data volume and computed by $(V_{No-RE} - V_{RE})/V_{No-RE}$, where V_{No-RE} is the total data volume and V_{RE} is the total volume of the reduced data. A higher RE efficiency means a higher prediction rate, eventually, a higher prediction accuracy.

TABLE I
POINT-TO-POINT BANDWIDTH SAVINGS RATIO

	No-RE (MB)	PRECO (MB)	Network overhead (MB)	Bandwidth saving ratio
Trace 1	1714.2	1420.1	18.1	16.10%
Trace 2	1643.5	514.7	37.2	66.42%

- Network overhead. It is calculated by $\frac{V_{Overhead}}{V_{data}}$, where $V_{Overhead}$ is the data volume of prediction messages and hashes in content transmission and V_{data} is the size of the total transmitted content data.
- Bandwidth saving ratio. It is computed by $(V_{No-RE} - V_{RE} - V_{Overhead})/V_{No-RE}$ in order to show the final bandwidth saving caused by both RE and network overhead reduction.

A. Simulation Setup

We evaluate the efficiency of PRECO with three different simulation scenarios. First, we deploy an AP infrastructure with one associated client in our simulator to evaluate the performance of PRECO without content overhearing. Then, we simulate a small network with two APs, each with one associated client, to evaluate the performance of PRECO using content overhearing. Finally, we built a wireless lattice-type mesh network to evaluate the benefits of redundancy-aware routing. PRECO uses MAXP to find the chunk boundaries and the trace is divided into chunks by these boundaries. We limited the chunk sizes to [256,1024] bytes. We set the merging distance threshold $d_T = 1500$ bytes, and the initial prediction window size $W_0 = 4KB$.

B. Point-to-Point RE Efficiency

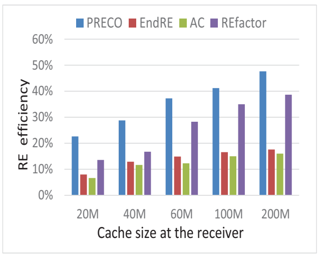
In this simulation, the AP node successively transfers $Trace_1$ and $Trace_2$. PRECO enables the client to utilize $Trace_1$ in its cache to make predictions when AP started to transmit $Trace_2$. The AP removed redundant bytes based on the predictions from the client. Table I shows experimental results for the different metrics without redundancy elimination ("No-RE") and with PRECO.

As we can see, for $Trace_2$, PRECO reduces the traffic volume from 1.64GB to 0.51GB. Additionally, when transferring $Trace_1$, there is also considerable redundancy, over 16%, detected by PRECO even without previous data transmission. This indicates such redundancy exists in $Trace_1$ itself.

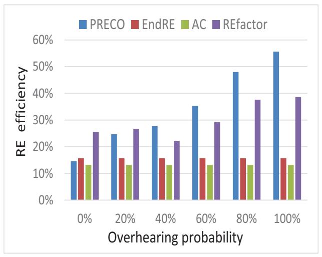
Compared with the significant traffic reduction, the network overhead is negligible. We can see that PRECO is effective in reducing bandwidth cost for both $Trace_1$ and $Trace_2$. Furthermore, when transferring $Trace_2$, PRECO can achieve higher bandwidth saving ratio, compared with $Trace_1$ transmission. This is because during $Trace_2$'s transmission, PRECO can reduce the redundant data not only using its own previous traffic but also using the previous traffic of $Trace_1$.

C. Benefits of Content Overhearing

In this simulation, two AP nodes, donated by AP1 and AP2, and two clients, denoted by C_1 and C_2, are deployed. AP_1 transfers the $Trace_1$ to C_1, and C_2 works in promiscuous mode so that it can overhear the transmission between AP1 and C_1. At the same time, AP2 transfers $Trace_2$ to C_2. We measured



(a) Different receiver's cache size



(b) Different overhearing probability

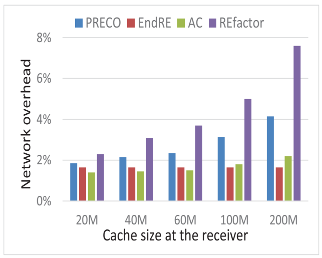
Fig. 5. RE efficiency.

the RE efficiency, network overhead and bandwidth saving ratio of PRECO, EndRE, AC and REfactor with different cache sizes and overhearing probabilities. Unless otherwise specified, the overhearing probability was set to 80% and the receiver's cache size was set to 200MB.

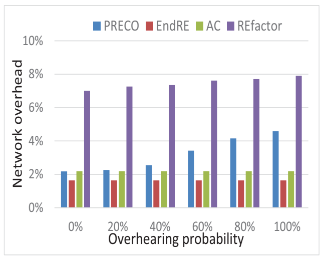
Figure 5(a) and 5(b) show respectively the RE efficiency of different methods with the cache size at the receiver varying from 20MB to 200MB, and that with the overhearing probability changing from 0% to 100%. We see that the RE efficiency follows PRECO>REfactor>EndRE>AC. EndRE and AC generate lower RE efficiency because they do not support overhearing, and they miss the opportunities to eliminate redundancy based on overheard traffic. In addition to using overhearing, PRECO can make more accurate prediction on redundant data than AC. AC chooses the feedback message in receiver's cache by finding the matched flow segment with the maximum number of matched chunks compared with the received flow. PRECO's prediction algorithm used in the receiver considers not only the number of matching chunks but also the distance between them, which provides more accurate prediction than AC. We further see that though REfactor also supports overhearing, it produces lower RE efficiency than PRECO. REfactor makes decision of the chunk removal based on the overhearing probability estimation, which may not be accurate since the estimation result is vulnerable to the dynamic changes in the network. If the estimation is not accurate, REfactor may make the wrong decision on redundancy elimination.

From Figure 5(a), we can also find that with the increase of the cache size, the RE efficiency increases. The reason is that cache with a larger size will cache more chunks, which will improve the probability of redundancy elimination and then RE efficiency. Also, we see that, for PRECO and REfactor, the RE efficiency increases with overhearing probability. This is because that higher overhearing probability brings much more redundant chunks from other transmissions, which improves the prediction accuracy and then the RE efficiency.

Figure 6(a) and Figure 6(b) show respectively the network overhead of different methods with cache size varying from 20MB to 200MB and that with overhearing probability changing from 0% to 100%. We see from both figures, REfactor has the highest network overhead among all these RE methods. The reason is that in REfactor, in order to estimate the overhearing probability for transmitted data, two APs need to communicate with each other, which incurs a large amount of network overhead. For EndRE, the network overhead comes

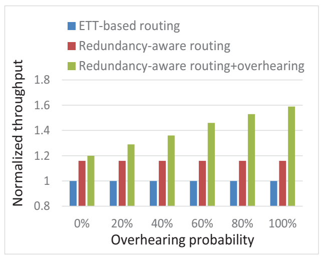


(a) Different receiver's cache size

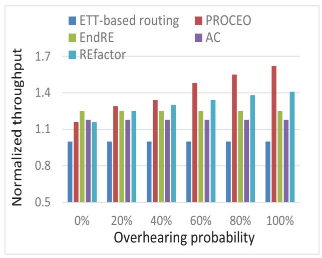


(b) Different overhearing probability

Fig. 6. Network overhead.

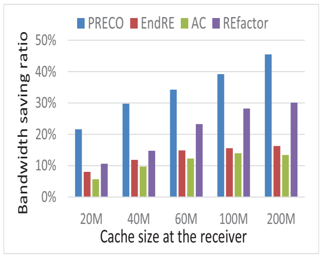


(a) Different overhearing probability

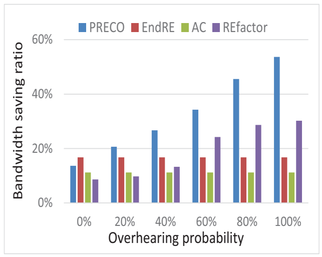


(b) Different overhearing probability

Fig. 8. Normalized throughput vs. overhearing probability.



(a) Different receiver's cache size



(b) Different overhearing probability

Fig. 7. Bandwidth saving ratio.

from the chunk hashes in content transmission. In AC, the network overhead includes the chunk hashes in feedback messages and content transmission. For PRECO, the network overhead includes chunk hashes and IDs in prediction messages and IDs in content transmission. As a result, EndRE, AC and PRECO generate far less network overhead than REfactor. From Figure 6(a), we also find that with the cache size increasing, the network overhead increases. For AC and PRECO, the reason is that a cache with a larger size will bring more data reduction and lead to more hash overhead in the transmission. For REfactor, as the cache size increases, the sender will cache more chunks and produce more overhead for overhearing probability estimation. In EndRE, in order to offload the hash computing from the receiver to the sender, the sender transfers the chunk hashes of all data contents no matter whether the data chunks are redundant or not, so its hash overhead keeps constant. Figure 6(b) shows that the network overhead in PRECO and REfactor increases with the increase of the overhearing probability. Higher overhearing probability produces higher RE efficiency, which leads to more hashes transmitted because of data reduction. For EndRE and AC, the increase of overhearing probability has no effect in their network overheads because they do not consider content overhearing.

Figure 7(a) and Figure 7(b) show respectively the bandwidth saving ratio of different methods with cache size at the receiver varying from 20MB to 200MB, and that with the overhearing probability changing from 0% to 100%. We see that the bandwidth saving ratio follows PRECO>REfactor>EndRE>AC. The result is consistent with that in Figure 5(a) and Figure 5(b) since the bandwidth saving is mainly caused by RE efficiency and the network overhead only accounts for a very small part of the bandwidth cost.

D. Benefits of Redundancy-aware Routing

In the simulation, we deployed a mesh network with 5 rows and 5 columns in total. We mark the node as $A_{m,n}(m, n \in \{0, 1, 2, 3, 4\})$, where m and n are the row and column index respectively. $A_{0,0}$ is the gateway of this mesh network. The gateway as the source node sends the packets to client C_1, associated with $A_{4,3}$, client C_2, associated with $A_{4,4}$, and client C_3, associated with $A_{3,4}$. In this scenario, we set the overhearing coverage to 1. It means that only the sender node's neighbor node can overhear the data transmission, which confirms the practical scenario since the nearest nodes have much more chance to overhear the data sent by the sender. C_2 can overhear the data transmission from $A_{4,3}$ to C_1 and $A_{3,4}$ to C_3. The gateway sends the $Trace_1$ to C_1 and C_3 successively, at the same time, it sends $Trace_2$ to C_2. We care about the network throughput from $A_{0,0}$ to C_2. The average date rate for each link varies from 800Kps to 1200Kps.

In order to investigate the benefits of our proposed redundancy-aware routing protocol, we deployed three routing approaches. (1) ETT-based routing: the gateway determines the optimal route to a receiver using ETT metric, and there is no network-wide PRECO deployment to perform RE over every links; (2) redundancy-aware routing without content overhearing: network-wide PRECO is deployed but without content overhearing, i.e., only on-path caching is enabled for RE, and the gateway computes the RETT metric to select the optimal route to a receiver; (3) redundancy-aware routing with content overhearing: it is like the second approach except that the content overhearing is enabled in PRECO. The cache size at the receiver was set to 200MB. We normalized the throughput (bytes/second) of redundancy-aware routing by that of the ETT-based routing.

Figure 8(a) shows the normalized throughput from the gateway to client C_2 when the overhearing probability was varied from 0% to 100%. From the figure, we can see that compared with the ETT-based routing, our proposed redundancy-aware routing can produce 20% more throughput. This is because unlike the ETT-based routing, the redundancy-aware routing uses RE. That is, the gateway steers the traffic through the nodes with high redundancy, which helps reduce transmission time and improve throughput. When the content overhearing is considered, the throughput is further improved. We can see as the overhearing probability increases, the throughput increase grows. When the overhearing probability is 90%, the throughput is improved over 60%. The reason is that as the overhearing probability increases, the node receives more redundant chunks from other transmissions, which will improve the prediction accuracy in PRECO and further reduce the redundant data. We also tested our redundancy-aware routing protocol using different RE methods. The normalized throughput between the gateway and the client is plotted in Figure 8(b). We see that compared with the ETT-based routing approach, both the RE approaches can improve the throughput due to the redundancy-aware routing and RE methods. We also see that the throughput improvement follows PRECO>REfactor>EndRE>AC. This is because high RE efficiency means less data transmission at each link, which reduces more transmission time on the total path and produces higher throughput improvement. Since the RE efficiency follows PRECO>REfactor>EndRE>AC, the PRECO outperforms other methods on throughput improvement. The throughput improvements grow in PRECO and REfactor as the overhearing probability increases due to the same reasons as in Figure 5(b).

VI. Conclusion

In this paper, we proposed a prediction-based IP-layer RE method with content overhearing named PRECO for wireless networks. In PRECO, wireless receivers compare incoming packets with previous received or overheard packets, predict future incoming data chunks, and send their hashes to the senders. A wireless sender removes redundant data chunks that already exist in the receiver's cache by comparing the hashes of outgoing data chunks with the predictions from the receiver. We also proposed novel prediction algorithms that allow PRECO to effectively improve prediction accuracy and overall bandwidth saving. PRECO is advantageous than previous overhearing-based RE methods in two aspects. First, by prediction-based IP-layer RE, PRECO does not require overhearing probability estimation. Second, it does not need cache content communication and complex coordination among wireless nodes when being deployed in multiple AP infrastructures and mesh networks. Thus, it enables efficient network-wide RE with content overhearing for wireless networks. We exploited the network-wide IP-layer service in wireless mesh networks, and proposed a redundancy-aware routing protocol to further enhance its benefit. Trace-driven simulation results show that PRECO provides significant performance benefits in comparison with other RE methods. In the future work, we will explore how to enable the gateway to efficiently learn the overhead data streams of all nodes for route determination in mesh networks.

Acknowledgements

This research was supported in part by U.S. NSF grants ACI-1661378 and CNS-1254006, and Microsoft Research Faculty Fellowship 8300751.

References

[1] C. Qiu, H. Shen, and L. Yu, "Energy-efficient cooperative broadcast in fading wireless networks," in *Proc. of INFOCOM*, 2014.

[2] A. Sarker, C. Qiu, and H. Shen, "A decentralized network with fast and lightweight autonomous channel selection in vehicle platoons for collision avoidance," in *Proc. of MASS*, 2016.

[3] J. Liu, L. Yu, H. Shen, Y. He, and J. Hallstrom, "Characterizing data deliverability of greedy routing in wireless sensor networks," in *Proc. of SECON*, 2015.

[4] Z. Li, H. Shen, and K. Chen, "Learning network graph of sir epidemic cascades using minimal hitting set based approach," in *Proc. of ICCCN*, 2016.

[5] A. Sarker, C. Qiu, H. Shen, A. Gil, J. Taiber, M. Chowdhury, J. Martin, M. Devine, and A. Rindos, "An efficient wireless power transfer system to balance the state of charge of electric vehicles," in *Proc. of ICPP*, 2016.

[6] L. Yan, H. Shen, J. Zhao, C. Xu, F. Luo, and C. Qiu, "CatCharger: Deploying wireless charging lanes in a metropolitan road network through categorization and clustering of vehicle traffic," in *Proc. of INFOCOM*, 2017.

[7] S. He, Z. Lu, X. Wen, Z. Zhang, J. Zhao, and W. Jing, "A pricing power control scheme with statistical delay qos provisioning in uplink of two-tier ofdma femtocell networks," *Mobile Networks and Applications*, vol. 20, no. 4, pp. 413–423, 2015.

[8] S. He, Z. Lu, X. Wen, Z. Zhang, Y. Sun, and L. Zhang, "Energy-efficient power allocation with qos guarantee in ofdma wireless networks," in *Proc. of WPMC*, 2014.

[9] M. Afanasyev, D. G. Andersen, and A. C. Snoeren, "Efficiency through eavesdropping: link-layer packet caching," in *Proc. of NSDI*, 2008.

[10] F. R. Dogar, A. Phanishayee, H. Pucha, O. Ruwase, and D. G. Andersen, "Ditto: a system for opportunistic caching in multi-hop wireless networks," in *Proc. of MobiCom*, 2008.

[11] S. Sanadhya, R. Sivakumar, K. Kim, P. Congdon, S. Lakshmanan, and J. Singh, "Asymmetric caching: improved network deduplication for mobile devices," in *Proc. of MobiCom*, 2012.

[12] S.-H. Shen, A. Gember, A. Anand, and A. Akella, "Refactor-ing content overhearing to improve wireless performance," in *Proc. of MOBICOM*, 2011.

[13] L. Chen, H. Shen, and S. Platt, "Cache contention aware virtual machine placement and migration in cloud datacenters," in *Proc. of ICNP*, 2016.

[14] A. Anand, C. Muthukrishnan, A. Akella, and R. Ramjee, "Redundancy in network traffic: findings and implications," in *Proc. of SIGMETRICS/Performance*, 2009.

[15] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, "Packet caches on routers: the implications of universal redundant traffic elimination," in *Proc. of SIGCOMM*, 2008.

[16] Y. Hua, X. Liu, and D. Feng, "Neptune: Efficient remote communication services for cloud backups," in *Proc. of INFOCOM*, 2014.

[17] B. Agarwal, A. Akella, A. Anand, A. Balachandran, P. Chitnis, C. Muthukrishnan, R. Ramjee, and G. Varghese, "Endre: An end-system redundancy elimination service for enterprises," in *Proc. of NSDI*, 2010.

[18] A. Anand, V. Sekar, and A. Akella, "Smartre: an architecture for coordinated network-wide redundancy elimination," in *Proc. of SIGCOMM*, 2009.

[19] E. Zohar, I. Cidon, and O. O. Mokryn, "The power of prediction: cloud bandwidth and cost reduction," in *Proc. of SIGCOMM*, 2011.

[20] N. T. Spring and D. Wetherall, "A protocol-independent technique for eliminating redundant network traffic," in *Proc. of SIGCOMM*, 2000.

[21] "Riverbed networks : Wan optimization." http://www.riverbed.com/solutions/optimize, [Accessed in Sep. 2016].

[22] "Juniper networks: Application acceleration." http://www.juniper.net/us/en/products-services/application-acceleration/, [Accessed in Sep. 2016].

[23] "Cisco wide are application acceleration services," http://www.cisco.com/en/US/products/ps5680/ Products_Sub_Category_Home.html, [Accessed in Sep. 2016].

[24] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[25] L. Yu, K. Sapra, H. Shen, and L. Ye, "Cooperative end-to-end traffic redundancy elimination for reducing cloud bandwidth cost," in *Proc. of ICNP*, 2012.

[26] Y. Hua, W. He, X. Liu, and D. Feng, "Smarteye: Real-time and efficient cloud image sharing for disaster environments," in *Proc. of INFOCOM*, 2015.

[27] A. B. N. Bjorner and Y. Gurevich, "Content-dependent chunking for differential compression, the local maximum approach," Microsoft Research, Tech. Rep. 109, 2007.

[28] J. Bicket, D. Aguayo, S. Biswas, and R. Morris, "Architecture and evaluation of an unplanned 802.11b mesh network," in *Proc. of MobiCom*, 2005.

[29] D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proc. of MobiCom*, 2003.