Packet Caches on Routers: The Implications of Universal Redundant Traffic Elimination

Ashok Anand*, Archit Gupta*, Aditya Akella*, Srinivasan Seshan† and Scott Shenker‡
*UW-Madison, †CMU, ‡UC-Berkeley
{ashok,archit,akella}@cs.wisc.edu, srini@cs.cmu.edu, shenker@cs.berkeley.edu

ABSTRACT

Many past systems have explored how to eliminate redundant transfers from network links and improve network efficiency. Several of these systems operate at the application layer, while the more recent systems operate on individual packets. A common aspect of these systems is that they apply to localized settings, e.g. at stub network access links. In this paper, we explore the benefits of deploying *packet-level redundant content elimination as a universal primitive* on all Internet routers. Such a universal deployment would immediately reduce link loads everywhere. However, we argue that far more significant network-wide benefits can be derived by redesigning network routing protocols to leverage the universal deployment. We develop "redundancy-aware" intra- and inter-domain routing algorithms and show that they enable better traffic engineering, reduce link usage costs, and enhance ISPs' responsiveness to traffic variations. In particular, employing redundancy elimination approaches across redundancy-aware routes can lower intra and inter-domain link loads by 10-50%. We also address key challenges that may hinder implementation of redundancy elimination on fast routers. Our current software router implementation can run at OC48 speeds.

Categories and Subject Descriptors: C.2.2 [Computer Communication Networks]: Routing Protocols

General Terms: Algorithms, Design, Measurement.

Keywords: Traffic Redundancy, Routing, Traffic Engineering.

1. INTRODUCTION

The basic property that some of the content on the Internet is highly popular results some data being repeatedly transferred across the network. A number of existing systems attempt to improve the efficiency of the network by eliminating these redundant transfers. There is wide-spread agreement that these approaches offer significant benefits in practice.

A common property of existing systems is that they typically operate in a localized fashion by eliminating the redundant transfers either on the link, or of the application, directly connected to the system. The goal of this paper is to explore some of the *implications* of *network-wide* deployment of redundancy elimination technology.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGCOMM'08, August 17–22, 2008, Seattle, Washington, USA.
Copyright 2008 ACM 978-1-60558-175-0/08/08 ...$5.00.

A majority of the redundancy-elimination systems have employed application-layer object caching to eliminate redundant data transfers. For example, Web proxy caches are an application-layer approach to reduce the bandwidth usage of static Web content within ISPs and at large stub networks. Numerous studies [25, 11] have explored the effectiveness of such designs. In recent years, a number of systems, both commercial [3, 4, 1, 2] and non-commercial [24], have been developed which operate *below* the application layer and attempt to eliminate *any* redundant strings of bytes that appear on the network. Such systems enjoy two benefits. First, they are not tied to a single application and can eliminate all forms of redundant information. Second, past evaluations have shown that more redundant information can be removed by focusing at the packet and byte levels than at the object level.

In this paper, we consider the *benefits* of deploying of *packet-level redundant content elimination as a primitive IP-layer service* across the entire Internet. We start with the assumption that all future routers will have the ability to strip redundant content from network packets on the fly, by comparing packet contents against those recently-forwarded packets which are stored in a cache. Routers immediately downstream can reconstruct full packets from their own cache. Applying this technology at every link would provide immediate performance benefits by reducing the overall load on the network. It also enables new functionality: for example, it simplifies application layer multicast by eliminating the need to be careful about duplicate transmissions.

However, universal redundancy elimination can yield even greater benefits if existing protocols are *redesigned with redundancy elimination in mind*. In this paper, we describe how wide-spread deployment of redundancy elimination can be leveraged by ISPs to change the way they compute routes giving rise to new and improved techniques for managing network resources. We analyze the benefits of selecting routes which maximize the opportunity to eliminate redundant content, versus routes which minimize hop count or other cost functions; An example is shown in Figure 1.

We consider such "redundancy-aware" modifications to both intra- and inter-domain routing. In our proposed approaches, ISPs first compute estimates of how often content is replicated across different destinations—we call these estimates *redundancy profiles*—and use these estimates in computing forwarding paths for their packets. We describe how ISP routers can compute redundancy profiles in parallel with forwarding packets. We also describe how ISPs can leverage centralized route control platforms (e.g. 4d [13] or RCP [8]) to compute network-wide redundancy-aware routes in a scalable and efficient fashion. In contrast with current state-of-the-art practices, our redundancy-aware approaches can allow ISPs better control over link loads, and offer them greater flexibility in meeting traffic engineering goals and in reacting to sudden traffic surges.

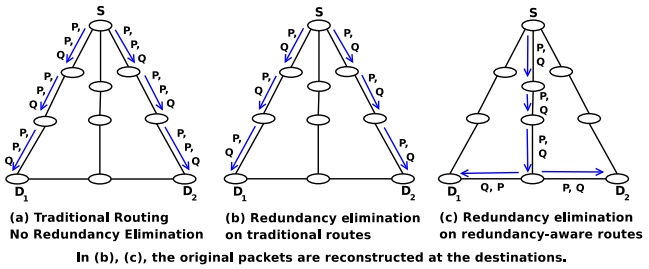


Figure 1: In (a), we show shortest path routing where the network carriers 18 packets in all). In (b), redundant packet elimination is employed on the shortest paths resulting in 12 packets total, or a 33% reduction. In (c), we use redundancy-aware routes which minimize the total number of packets carried by the network resulting in 10 packets total, or a 44% reduction.

We have evaluated the full range benefits arising from a universal deployment of redundancy elimination, and from using our redundancy-aware route selection algorithms. Our evaluations use Rocketfuel topologies and workloads from full packet traces collected at a large US university as well as synthetic traffic models derived from the traces. When traditional shortest path routing is used, we find that applying redundancy elimination on all network links brings down the network-wide utilization by 10-50%. In contrast, when redundancy-aware routing is employed, we find that the network-wide utilization is reduced by a *further* 10-25%. We also study the effect of staleness of redundancy profiles on route quality. We find that the benefits from redundancy-aware routing are significant even when traffic patterns change unexpectedly and the route computation is unable to react to the change (as might happen during flash-crowd events). Overall, we find that a wide-spread deployment of redundancy elimination can be leveraged to obtain very significant network-wide benefits. These benefits can quickly than offset the initial high cost of deployment.

We also consider some key challenges that may hinder the deployment of packet-level redundancy elimination in today's high-speed routers. Starting from the algorithm in [24], we make key enhancements to the design of packet caches and to cache lookup algorithms in order to reduce both the total amount of storage required and the number of memory accesses incurred per packet. We have implemented these improvements in Click [18]. Our simplistic implementation offers a throughput of 1Gbps in software on a 1.8GHz Linux box. We argue that, with better hardware support, the throughput of the software implementation can easily exceed 2.5Gbps. Even higher throughputs can be attained in hardware.

This paper is structured as follows. In Section 2, we discuss a prior approach for packet-level redundancy elimination and outline the issues we consider in this paper. In Sections 3 and 4, we present redundancy-aware intra- and inter-domain routing, respectively. In Section 5, we present a measurement study of key properties of redundant content observed in real packet traces. In Section 6, we evaluate the benefits of universal redundancy elimination and redundancy-aware routing. In Section 7, we present our software router implementation of packet-level redundancy elimination. We discuss related work in Section 8 and conclude in Section 9.

2. BACKGROUND

In this section, we present a brief outline of a popular mechanism for packet-level redundancy elimination, and review current practices in routing and traffic engineering. We then discuss the challenges in updating routing to leverage a universal deployment of redundancy elimination. We end with a preliminary empirical study which points to the likely benefits of modifying routing in this way.

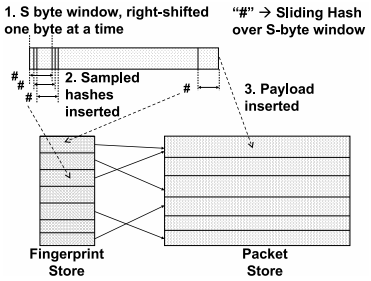


Figure 2: Packet-level redundancy detection.

2.1 Algorithm for Redundancy Elimination

We describe a fast algorithm for identifying chunks of redundant content across packets. This algorithm has been employed in various forms in the past, e.g., for detecting duplicates in a file system [16, 19] and for detecting worms [22]. The algorithm we discuss here was first conceived by Spring et. al [24] who applied it to remove redundant content from access links. Their approach operates at access routers, as packets enter or leave a stub network.

For every packet going in a particular direction, the algorithm computes a set of *fingerprints* by applying a hash function to each 64 byte sub-string of the packet's payload. This choice of sub-string size offers good performance in practice [24]. Thus, for an S-byte packet ($S \geq 64$), a total of $S - 63$ fingerprints are obtained. Rather than use an MD5 hash, the algorithm uses a sliding hash function called Rabin fingerprint [20], which significantly cuts down the hash computation time per packet [24]. A subset of these fingerprints are selected at random per packet as its *representative fingerprints*.

Representative fingerprints of all the packets observed over some past interval of time are stored in a *fingerprint store* at the router. The packet payloads are stored in a *packet store*. Pointers are stored from each fingerprint to the corresponding packet (Figure 2).

After computing representative fingerprints for an arriving packet, each fingerprint is checked against the fingerprint store to check if the fingerprint already exists there. If a match is found, this means that the incoming packet has a 64 byte sub-string that matches with an in-cache packet. The matching packet is retrieved and the 64B match region is expanded left and right to obtain the region of overlap between the two packets. The new packet is inserted into the packet store, and the mapping in the fingerprint store is updated so that the matching fingerprint points to the new packet. The remaining representative fingerprints of the arriving packet are also inserted into the fingerprint store. In some situations, more than one representative fingerprint of the incoming packet can observe a match; this means that different regions of the arriving packet have matched with distinct regions of one or more cached packets.

Each match region is removed from the incoming packet and replaced with a *shim* which *encodes* the redundant content in the packet. The shim provides the fingerprint which caused the match, and byte range for the matching in-cache packet. The shim can be used by a downstream router to reconstruct the packet from its own local cache. It is assumed that that the cache on the downstream router is consistent with the upstream cache.

2.2 Intra and Inter-domain Routing

ISPs make *intra-domain* routing decisions on the basis of a packet's destination IP address. Since selecting static routes per destination (e.g., always using paths with small hop counts) could impact their ability to control the load on network links, many ISPs employ traffic engineering (TE) techniques. These involve estimating expected volume of traffic between different locations (PoPs) in a network [17] and computing routes so as to spread load evenly across network links. Although ISPs are known to overprovision

their links, traffic engineering is crucial to manage resources in the face of traffic variations.

When *selecting inter-domain* routes, ISPs attempt both to reduce usage costs of expensive inter-domain links and to minimize the impact of inter-domain traffic on intra-domain links. Typically, ISPs statically select the most cost-effective AS as the next hop for a destination. Packets are sent to the next hop either along the early-exit route or, in some cases, along an exit point that is chosen based on mutual agreements with the neighbor.

Meeting network-wide traffic engineering objectives effectively is very challenging today (in part because of the difficulty in predicting traffic volumes accurately). In particular, current intra-domain routing and TE practices cannot easily adjust to variations in traffic volumes. The variations could cause the load on intra-domain links to increase beyond acceptable limits. Similarly, the load on expensive and congested inter-domain links can increase significantly as well. In both cases, this could lead to a violation of TE objectives and of service level agreements (SLAs) with neighboring networks.

Applying redundancy elimination on network links improves the effective utilization of the links and provides ISPs greater flexibility in meeting their network-wide objectives. The flexibility is further enhanced when routing and traffic engineering are modified to leverage link-level redundancy elimination.

2.3 Toward Redundancy-Aware Routing

We assume that redundancy elimination approaches such as the one described in Section 2.1 are applied on input and output port of Internet routers in a hop-by-hop manner. An *upstream* router *removes* redundant content as a packet *leaves* it, while the router immediately *downstream reconstructs* the packet as soon as it *arrives* (assuming the packet has redundant content). All routers cache packets that they have forwarded or received over a fixed short interval of time (e.g., 10s). Upstream and downstream packet caches should be the same size and the routers must employ the same hash functions and random seeds to ensure consistency.

To leverage a universal deployment of redundancy elimination and improve network-wide utilization, ISPs must change the way routes are computed today, as well as how routers act on packets.

In particular, ISPs must perform three key tasks: (1) ISPs must first track how packet content is replicated across different points in their network; We call this information the "Traffic Redundancy Profile"; (2) Based on the network-wide profile, ISPs must then construct intra and inter-domain forwarding paths which maximize the likelihood of duplicate data traversing the same network links and, at the same time, allow ISPs to meet their network-wide objectives; We call this "Redundancy-Aware Route Computation". And, (3) Router-level redundancy elimination techniques must operate on every packet at every router along network paths.

Our goal is to systematically understand how ISPs may implement these tasks, and show that ISPs can obtain *significant benefits* in terms of controlling the loads on their links, being able to meet their TE objectives satisfactorily, being better prepared for sudden traffic variations, and reducing usage costs and congestion on inter-domain links. Next, we discuss initial measurements which point to the potential benefits of employing redundancy-aware routes.

Preliminary Study. We conducted a preliminary measurement study where we tracked packets originating from a high volume /24 prefix owned by a large US university (the packets are headed for the commercial Internet). Traffic from the university enters its primary ISP at Chicago. We analyzed this traffic using the algorithm in Section 2.1 and found that 45% of the packet contents were duplicated for a 150s traffic snapshot using a packet store that could hold all packets in the snapshot; that is, the ratio of the total size of the matched regions in all packets to the total size of all packets was 0.45. Further, we dissected the /24 traffic leaving the primary ISP's network from its New York, Washington DC and Boston PoPs. About 22% of the packet contents leaving New York were duplicated in the 150s snapshot. The fraction was 18% for DC and 2% for Boston. Also, the shortest paths from Chicago (close to where the university is located) to these cities were non-overlapping. Thus, simply employing redundancy elimination techniques in a hop-by-hop manner can yield savings of 2-22% (when only considering the bytes due to the /24) on the intra-domain links of the primary ISP.

Interestingly, 10% and 9% of the contents of packets going to New York also appeared in packets going to Boston and Washington DC. Thus, if packets to Boston and Washington DC are routed via New York (this does not cause significant path inflation) and then redundancy elimination applied, the overall utilization of the network links can be brought down even further.

3. INTRA-DOMAIN ROUTING

In this section, we present our redundancy-aware intra-domain routing approach which can help ISPs manage link loads more effectively and reduce overall network utilization. As mentioned earlier, the ISP gathers information on how content is duplicated within its network over a certain interval of time, and construct routes which maximize the potential for redundancy elimination. We assume that all ISP routers employ redundancy elimination.

To begin with, we assume that the ISP has perfect and timely knowledge of the prevailing patterns of redundancy in its network and that it can configure redundancy-aware paths within the network in a negligible amount of time. We also assume that packets are duplicated in full, if at all. We start with a simple setting where we consider packets originate from a single PoP in an ISP. We extend this to a network-wide setting and construct redundancy-aware intra-domain routes between all pairs of PoPs in the ISP.

Following this, we discuss practical issues in redundancy-aware intra-domain routing, such as fast approaches for estimating the redundancy patterns, accounting for partial replication of content across packets, and computing redundancy-aware routes.

3.1 A Single PoP

We use the following abstract model to develop our approach. We represent the ISP using a graph $G = (V, E)$. We focus on traffic originating from a single ISP PoP, denoted by S ($\in V$). We refer to S as the *source* or *ingress*. Nodes $D_1, D_2, \ldots, D_m \in V$ denote the *egress PoPs* or *destinations* through which traffic originating at S exits the ISP. Other vertices in V represent ISP backbone routers.

We now model duplicated packets within the traffic originating from S. Suppose that N distinct packets $\{P_1, P_2, \ldots, P_N\}$ originate at S over a certain time duration T. All other packet originating at S in this interval are duplicates of the N distinct packets. Each distinct packet P_i can have one or more "copies". We use the term "copy" in a loose sense: specifically, we consider the original distinct packet to be the first copy. Some copies of the distinct packet P_i may all be destined for the same destination D_j, while other copies may be headed for other destinations.

We assume that the ISP has all information regarding destinations of the different packet copies. Specifically, the ISP has a list of constants $cpy_{i,j}$ defined so that $cpy_{i,j} = 1$ if a copy of distinct packet P_i is destined for egress D_j. For instance, say that distinct packet P_1 originating at S has four copies overall, two of which are destined for PoP D_1 and one each for PoPs D_3, D_5. Then, $cpy_{1,1} = cpy_{1,3} = cpy_{1,5} = 1$, and $cpy_{1,j} = 0$ for all other destinations D_j.

We call this list of cpy's the *redundancy profile* for the traffic originating from S in the time interval T. In practice, an ISP can

compute the profiles as packets enter its network (Section 3.3.2). Next, we show how the ISP can use the redundancy profile to compute redundancy-aware routes from S to the different D_js. We first define a few variables which we employ in explaining our approach.

We refer to the traffic going from S to a particular destination within the time interval T as a single *flow*. For each flow j (i.e., the traffic to destination D_j), we define variables $rte_{j,e}$ such that $rte_{j,e} = 1$ if the redundancy-aware route from S to D_j goes through edge e, and $rte_{j,e} = 0$ otherwise. Binary values for $rte_{j,e}$ ensure that all traffic between S and D_j is routed along one path.

We use a variable $FP_{i,e}$ for an edge e and distinct packet P_i to denote the *footprint* of the copies of P_i on edge e. The footprint is the amount of resources consumed on edge e when the copies of P_i are routed toward their respective destinations using redundancy-aware routes and all routers perform redundancy elimination. For instance, if none of the copies of P_i is routed over e, then the footprint due to P_i and its copies on edge e is zero, i.e., $FP_{i,e} = 0$. If multiple copies of P_i are routed over the edge e, then effectively only one copy goes through e because the remaining copies are eliminated as being redundant. In this case, the footprint of the copies of P_i on the edge e is a function of the size of the distinct packet P_i. In this paper, we pick the footprint function to equal the size of the packet P_i multiplied by the latency of the link e, or $FP_{i,e} = lat_e \times |P_i|$. The intuition behind choosing this function is that a packet consumes more network resources overall when it traverses high latency links and/or when the packet size is large. Other functions reflecting network usage can also be considered.

The ISP's objective in computing redundancy-aware routes is to compute the rte variables such that *total footprint summed over all network edges is minimized*. In order words, the goal is to compute routes from S which minimize the total network resources consumed by traffic originating at S within the interval T when all routers perform redundancy elimination.

We formulate the ISP's objective as the output of a *Linear Program (LP)*. We first describe the *constraints* for the LP, followed by the *optimization objective*. We have the following constraints per distinct packet P_i, based on the definition of the footprint function:

$$\forall j, FP_{i,e} \geq lat_e \times cpy_{i,j} \times rte_{j,e} \times |P_i|$$

Since the footprint $FP_{i,e}$ cannot exceed resources consumed when routing a single copy of P_i on e , we have, $FP_{i,e} \leq |P_i| \times lat_e$.

Next, we set up flow conservation constraints for nodes in V. For backbones routers v, we have: $\forall j, \sum_{e \in \delta^+(v)} rte_{j,e} = \sum_{e \in \delta^-(v)} rte_{j,e}$, where, δ^+ indicates flow entering node v, and δ^- indicates flow leaving node v. For source S and destinations D_j, we have:

$$\begin{aligned} \forall j, \quad & \textstyle\sum_{e \in \delta^-(S)} rte_{j,e} - \sum_{e \in \delta^+(S)} rte_{j,e} = \quad 1 \\ \forall j, \quad & \textstyle\sum_{e \in \delta^+(D_j)} rte_{j,e} - \sum_{e \in \delta^-(D_j)} rte_{j,e} = \quad 1 \end{aligned}$$

Finally, we require a set of constraints to ensure that link capacities are obeyed. Suppose edge e cannot carry more than Cap_e packets within the interval T (Cap_e can be derived from e's raw capacity). Then, we require: $\forall e, \frac{1}{lat_e} \sum_n FP_{n,e} \leq Cap_e$. We use a normalizing factor $\frac{1}{lat_e}$ to obtain the total size of packets carried by e.

The objective of the LP is to lower the *total network footprint* subject to the above constraints, or Minimize $\sum_e \sum_i FP_{i,e}$.

We allow fractional values for the variables rte in the solution for the above LP. Fractional values indicate how traffic may split across different possible paths between S and a destination.

3.2 Multiple Ingresses, Traffic Engineering

We extend the above approach for computing redundancy-aware routes to a network-wide setting. The goal is to use redundancy-awareness to help ISPs meet their traffic engineering goals more effectively. Our network-wide approach tries to always obtain better network-wide guarantees than existing TE approaches, such as OSPF-based weight tuning [6]. We illustrate our approach using the "Maximum load" objective, wherein the ISP employs traffic engineering to minimize the maximum link utilization in its network. Traffic can originate at any network PoP.

To explain our approach, we introduce a per-ingress parameter $cpy_{P_{n,i},D_j}$ which is 1 if a copy of distinct packet $P_{n,i}$ is destined for D_j. $P_{n,i}$ denotes the i^{th} distinct packet originating from ingress S_n within an interval T. Similarly we extend the link footprint variable to capture the contribution of packets originating from different ingresses to a particular link e; we denote this as $FP_{P_{n,i},e}$. In a similar fashion, we define variables $rte_{S_n,j,e}$ which identify if the flow between S_n and D_j flows through edge e. We assume that packets originating from different ingresses have no content in common. (We omit several details for brevity.)

As with the single ingress case, we first formulate a network-wide LP where the objective of the ISP is to lower the network footprint due to traffic originating from all PoPs, or Minimize $\sum_e \sum_i \sum_n FP_{P_{n,i},e}$. Next, we place link capacity constraints and incorporate the "Max Load" objective as follows: Suppose that, based on the measured network traffic matrix, the ISP estimates that traditional traffic engineering approaches (e.g. OSPF-based approaches [6, 12]) can bound the link loads by a factor $\alpha < 1$. ISPs today try to maintain $\alpha \approx 0.5$. Given this, we normalize each link's capacity Cap_e using $Cap_e \leftarrow Cap'_e = \alpha Cap_e$ and minimize the network-wide footprint subject to the following new capacity constraints:

$$\forall e, \frac{1}{lat_e} \sum_i \sum_n FP_{P_{n,i},e} \leq Cap'_e$$

The solution to this LP is the set of $rte_{S_n,j,e}$ variables. Due to normalization of capacities and our objective of minimizing network footprint, the maximum link load due to this solution is at least as good as, if not much better, compared to traditional TE approaches.

3.3 Centralized Route Computation and Practical Issues

Our redundancy-aware approaches can be applied by an ISP alongside centralized routing platforms such as RCP [8] and 4D [13]. At regular N minute intervals, border routers in the ISP can compute the redundancy profiles (i.e. the constants $cpy_{P_{n,i},D_j}$) for packets observed during the first T seconds of the interval, and transmit the profiles to a logically central route control platform. We discuss how to track the redundancy profiles, especially when packet contents may not be duplicated in full, towards the end of this section.

The controller formulates the network-wide Linear Program and computes the routes (i.e. the $rte_{S_n,j,e}$ variables). The computed routes are configured on routers and used for rest of the N minutes.

In addition to the periodic updates, border routers could also track the redundancy profiles on a minute-by-minute basis and inform the route controllers of significant changes in the traffic redundancy patterns. This allows the controller to respond better to sudden traffic surges and changes in redundancy profiles.

3.3.1 Scalability

The input to the network-wide Linear Program includes the constants for the redundancy profiles. The input size thus grows linearly in number of distinct packets observed during the interval T. The size can be prohibitively large if millions of distinct packets appear in a short time. The amount of data to be transmitted to the central route controller will be high, resulting in excessive control overhead. Also, existing LP solvers cannot handle large input sizes.

We employ two simplifications to address the scalability issues.

Based on an empirical study of real traces, we observed that content at an ingress PoP is rarely duplicated across ≥ 3 destination PoPs (Section 5). Thus, we only consider content duplicated across 2 destinations and ignore duplication across > 2 destinations.

We make another simplification to improve the scalability. We "combine" the redundant content in packets going to an identical set of destinations into a larger *aggregated packet*; copies of the aggregated packet are considered to be destined for the same set of destinations as the individual packets. For example, suppose that distinct packets $P_1, \ldots, P_l$ all have two copies, with one copy going to destination D_1 and another to D_2 (all traffic is observed in a time interval T). We create an equivalent *single* aggregated packet of size $\sum_1^l P_i$ which goes to destinations D_1 and D_2. Thus, the aggregated packet captures the total overlap in the content going to D_1 and D_2. This aggregation approach reduces the total number of cpy variables without changing the quality of the solution obtained for the LP — the number of variables reduces from $2l$ to 2 in the above example.

With these two simplifications, the total number of variables for the entire network is now on the order of the square of number of PoPs in the network and the control overhead is thus much smaller. We refer to the redundancy profiles captured using aggregated packets in the above fashion as the *aggregated redundancy profiles*.

Next, we describe an approximate approach for computing the aggregated redundancy profiles at the ingress routers of an ISP network as packets stream into a network. We also address issues arising from content being partially replicated across network packets.

3.3.2 Computing Redundancy Profiles

We discuss an extension to the algorithm in Section 2.1 to compute the aggregated profiles in practice. The approach we describe is run constantly on each ingress router. Suppose an incoming packet P at an ingress router has a match with a single packet P_{cache} stored at the router, and that P and P_{cache} are headed for different destinations D_1 and D_2. We count the size of the matching region $|P \cap P_{cache}|$ towards the total amount of content *common to destinations D_1 and D_2*. If P and P_{cache} are both headed to the same destination, say D_1, then we count $|P|+|P_{cache}|-|P \cap P_{cache}|$ towards content exchanged between the ingress and D_1; in this manner, we approximately track the total amount of *unique content exchanged between the source and D_1*. If the incoming packet P has a match with more than one cached packet, say $P_{1,cache}$ and $P_{2,cache}$, we count each match region *separately* towards the redundancy profiles; that is, we run the aforementioned tallying approach first for P and $P_{1,cache}$, and then for P and $P_{2,cache}$. We also track packets in the ingress router's packet store which observe no matches during the interval T. We group such packets by their destination and compute the total size of the packets in each group. This total is then added to the total volume of unique content exchanged between the ingress and the corresponding destination.

At the end of interval T, the ingress router gathers aggregated counts for: (1) the size of content shared between pairs of egresses, and (2) the volume of unique content exchanged with different egresses. This forms the aggregated redundancy profile for the ingress PoP, and is transmitted to the route controller. Note that we only focus on content that is duplicated across 2 destinations, if at all.

This approach clearly approximates the true redundancy profile as described in Section 3.1. However, our trace-based evaluation (Section 6) shows that the inaccuracies in our approach do not significantly affect the quality of the routes we compute.

3.3.3 MPLS Networks

As mentioned before, we permit fractional solutions to the network-wide Linear Program. The fractional solution can be implemented in MPLS-based networks by establishing the appropriate "traffic trunks", or label switched paths (LSPs), between ISP PoPs [9]. Care must be taken to construct LSPs and allot packets to them in a redundancy-aware manner. This is crucial in order to extract the maximum amount of redundant content from network traffic. Otherwise, packets may be alloted to LSPs in such a manner that redundant packets destined for different egresses are routed along LSPs which have very few network links in common.

While a thorough investigation of how to establish LSPs and allocate packets is beyond the scope of this work, we have developed a preliminary redundancy-aware heuristic which seems to offer satisfactory performance in practice. The details can be found in [5].

4. INTER-DOMAIN ROUTING

In this section, we present redundancy-aware inter-domain routing which can help ISPs minimize the overall impact of inter-domain traffic on internal and peering links. We consider as "inter-domain" traffic the set of all packets traversing the ISP whose destinations are routable only through peers of the ISP. We consider two approaches: local and cooperative.

The *local* approach applies to an ISP selecting its next-hop ASes in BGP, as well as the particular exit point(s) into the chosen next hop. In this approach, an ISP aggregates its inter-domain traffic over a selected set of next hops and the corresponding exit points so as to aggregate potentially redundant traffic onto a small number of network links. Thus, the ISP can significantly reduce the impact that inter-domain traffic imposes on its internal and peering links. To compute routes in this manner, the ISP must track (1) the amount of redundant content that is common to different destination prefixes and (2) the route announcements from peers to the destinations.

The cooperative approach applies to ISPs which are willing to coordinate their inter-domain route selection decisions. In this approach, the ISPs compute routes which minimize the *overall* impact of inter-domain traffic across the internal links of all ISPs involved and the peering links between the ISPs. We explore the ideal benefits from cooperation and ignore important issues such as the need to maintain privacy of internal information.

4.1 Local Approach for an ISP

The intra-domain approach, presented in Section 3, can be extended in a straight-forward manner to perform next hop-AS selection. This simply requires a change to the input network graph G and the overall objective of the ISP. Our approach described below focuses on inter-domain traffic originating at a particular PoP in an ISP and can be extended to all inter-domain traffic of the ISP. We present the high level ideas and omit the details for brevity.

The ISP's network footprint objective encompasses the footprint $FP_{i,e}$ of both the internal edges of the ISP and its peering links. The input graph $G = (V, E)$ is constructed as follows: the set V is composed of three subsets V_1, V_2, and V_3 (Figure 3). V_1 is the set of all intra-domain routers or the PoPs of the ISP, including the ingress PoP S where the inter-domain traffic originates. V_3 is the set of destination prefixes $D_1, D_2, \ldots, D_m$. These are the prefixes to which the inter-domain traffic from S must finally reach. We assume that the ISP computes aggregated redundancy profiles across the m destinations. To derive the ideal benefits of redundancy elimination, all possible destination ASes must be considered in the set V_3. However, in practice, it may suffice to focus on just the top few destination prefixes by volume. Finally, the set V_2 is composed of "intermediate nodes" which model possible next hop ASes for each destination, as well as their peering locations with the ISP.

The set of edges, E, is composed of three subsets: E_1, the set of intra-domain edges, E_2, the full set of peering edges between the

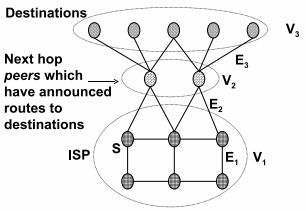


Figure 3: Input graph for the local inter-domain approach.

ISP in question and each of its peers, and E_3, which are "intermediate edges" between nodes in V_2 and V_3. We construct an intermediate edge between an intermediate node v and a destination D_j if the ISP corresponding to v has announced a route to D_j. We only include edges and vertices for a peer if the peer is among those who have a path with the smallest number of AS hops to the destination.

The rest of the inter-domain route selection approach is similar to the intra-domain case. Again, a centralized route controller may be employed to compute routes which minimize the footprint due to inter-domain traffic. The ingress router at S could compute the inter-domain redundancy profile using the approach in Section 3.3.2, and transfer the profile to the router controller. The output from the route controller is the next-hop AS and the internal route to the exit point into the next-hop for each destination prefix.

4.2 Cooperative Approach for Two ISPs

For simplicity we consider the case where just two ISPs coordinate their inter-domain route selection. Our approach can be extended to multiple ISPs. Our approach works as follows: rather than compute inter-domain routes in isolation, each ISP tries to aggregate the redundant content in inter-domain traffic together with the redundant content in its intra-domain traffic, so as to bring down the overall utilization of all participating networks.

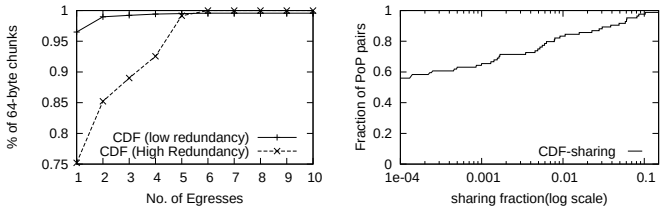
Thus, the key difference from the intra-domain routing formulation is that the input graph used by either ISP is the *union* of the topologies of the two networks and peering links. The inputs to an ISP's Linear Program are its intra-domain redundancy profiles *and* the inter-domain profile for traffic between ingresses in itself and egresses in the neighbor. The output of an ISP's formulation include its intra-domain routes and a list of exit points for traffic destined to egresses in the neighbor (and how to split inter-domain traffic across the exit points).

5. MEASUREMENT RESULTS

We present a brief study of key properties of content redundancy observed at the packet-level in real traces. We focus on the extent to which content is duplicated across two or more destinations. Our observations shed light on the potential for redundancy-aware routing. They also justify the key choices we have made in designing the approaches outlined in Sections 3 and 4. We also leverage the observations to construct synthetic traces which we use extensively in Section 6. For brevity, we only focus on intra-domain settings.

Traces. We collected full packet traces at a large US university's access link to the commercial Internet. We collected multiple 150s-snapshots at the beginning of every hour starting at 10am and ending at 7pm on Jan 26, 2007. In addition, we also separately monitored the traffic originating from a high volume /24 prefix owned by the university, which hosted some of the most popular servers on campus (during the same time-period). The latter traces are likely to be representative of a moderate-sized data center.

Extent of redundancy. We used the approach outlined in Section 2 to quantify packet-level content redundancy. In the case of Internet-bound traffic on the University access link, we found that



(a) 64B chunks duplicated across multiple PoPs (b) Content shared between pairs of PoPs

Figure 4: Duplication across multiple destination POPs

the average redundancy percentage was 16% using 400MB packet store, meaning that the ratio of the total size of the matched regions in all packets to the total size of packets was 0.16. When we used a 2GB packet store, the average redundancy proportion increased to 20%. For traffic originating from the high-volume /24 prefix, the redundancy proportion was 50% on average with a 2GB packet store. These observations show that when redundancy elimination is applied in a localized fashion on individual access links, the utilization of the links can be reduced by 16-50%.

Extent of duplication across destinations. Our redundancy-aware approaches are most effective when content is duplicated across multiple destinations. To examine the benefits of redundancy awareness, we emulate a scenario where traffic originating at our vantage points (i.e. the University access link and the /24 prefix) is routed across the internal network of a tier-1 ISP (SprintLink). For both cases, we assume that the traffic enters the ISP at its Chicago PoP. Using traceroute and undns [23] we mapped the destination prefixes observed in traces to PoPs of the tier-1 ISP. We then examined how often content is duplicated across different ISP PoPs.

In Figure 4(a), we show the number of different PoPs to which a distinct 64B chunk observed in a trace was destined to. We study a full University trace (redundancy proportion of 17%) and a trace of traffic from the high-volume /24 prefix (redundancy of 48%). In the former case, we note that for 97% of the chunks, either there were no duplicates or the duplicates went to the same PoP as the original. In 3% of the cases, the duplicate and original chunks were destined to 2 distinct PoPs. For the trace of the /24 prefix, we see more significant duplication across PoPs, with duplicates destined for 2 PoPs in nearly 10% of the cases. In general, very few chunks are duplicated across $\geq$ 3 PoPs in either set of traces. We examined several other traces and observed similar trends. This justifies our approach of focusing only on the amount of content duplicated across *pairs of destinations* when computing redundancy profiles.

Next, we examine whether content duplicated across a set of destinations amounts to a significant proportion of all traffic sent to the destinations. In Figure 4(b), we show the total volume of traffic originating from the high-volume /24 prefix which is duplicated across a pair of destination PoPs, relative to the total volume of traffic from the /24 to the two PoPs. We see that the proportion of shared content varies significantly across different pairs of destination PoPs. In many cases, there is very little sharing of redundant content: the proportion of shared content is $< 1\%$ for nearly 80% of the PoP pairs. For roughly 10% of PoP pairs, the extent of sharing is very significant, ranging between 5 and 15% of the total traffic. We studied other traces of the /24 prefix and observed a similar trend of a few PoP pairs sharing a significant amount of content. Furthermore, we also found signs of positive correlation between the total volume of traffic of the PoP pair and the extent of content shared (the results are omitted for brevity).

6. EVALUATION

In this section, we present results from an extensive study of the benefits of redundancy elimination both when applied to traditional

routes and when applied along with redundancy-aware routes. We consider both intra and inter-domain settings. We also examine the impact of network topology on the benefits derived. Finally, we study the ability of redundancy elimination and redundancy-aware approaches to absorb sudden traffic surges.

Unless otherwise stated, our metric of comparison is the *network footprint* which reflects the aggregate utilization of an ISP's network resources. A significant reduction in network footprint implies that an ISP is able to better control the usage of its network resources and meet its traffic engineering goals in a more effective fashion.

We mostly focus on benefits in the *ideal* case, in that we assume networks have perfect information regarding the redundancy profiles and can compute redundancy-aware routes instantaneously. We do study the impact of practical limitations from staleness of redundancy profiles. Our evaluation is based both on the real packet traces (Section 5), and synthetic traces which are described next.

Our study indicates that redundancy elimination and redundancy-awareness can reduce network footprint to a very significant extent. Thus, the benefits of a universal deployment of redundancy elimination seem to easily offset the initial cost of deploying the mechanisms on multiple network routers.

Generating Synthetic Traces. Synthetic traces allow us to explore the relationship between various redundancy profiles and the overall benefits offered by our approaches. We construct synthetic traces based on key properties of real packet traces.

In what follows, we first outline how to generate a synthetic *intra-domain* trace for traffic originating at a *single PoP* of an ISP's topology. This can be extended trivially to network-wide intra-domain traces, as well as to inter-domain traces.

Packets are of the same size in all synthetic traces. Each synthetic trace has three parameters: $\rho_{overall} \in [0, 0.5]$ and ρ_{intra}, $\rho_{inter} \in [0, 1]$. These determine if there are duplicate packets, and whether the duplicate packets are all headed for the same destination.

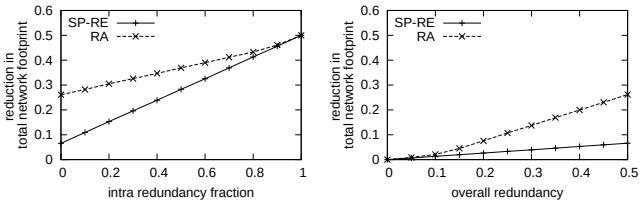
To elaborate, $\rho_{overall}$ is the total fraction of redundancy in the traffic; For instance, when $\rho_{overall} = 0.5$, only 50% of the packets are unique. In general, no packet has more than one duplicate in all our synthetic traces. Thus, we do not model duplication of content across 3 or more destinations. As our empirical study in the previous section showed, this approximation is unlikely to affect the representativeness of our synthetic trace-based analyses.

To construct the trace, we first create a large number of unique packets. Each packet has a duplicate with probability $\frac{\rho_{overall}}{(1-\rho_{overall})}$. If a packet has no duplicate (with probability $\frac{(1-2\rho_{overall})}{(1-\rho_{overall})}$), we "send" the packet to a PoP in the ISP selected with a probability proportional to the population of the city represented by the PoP (this is based on the gravity model for network traffic volumes [21]). If the packet has a duplicate, we create the duplicate, and with probability ρ_{intra}, we send the packet and its duplicate to the same destination, where the destination is selected according to the gravity model; thus, ρ_{intra} controls the number of packets which are duplicated between a pair of PoPs. With a probability $\rho_{inter} = 1 - \rho_{intra}$, we select two different destinations according to the gravity model, and send them a copy each; thus, ρ_{inter} controls the number of packets duplicated across two different destinations.

We assume that routers have sufficient memory to store *all* packets within a synthetic trace that are forwarded to them.

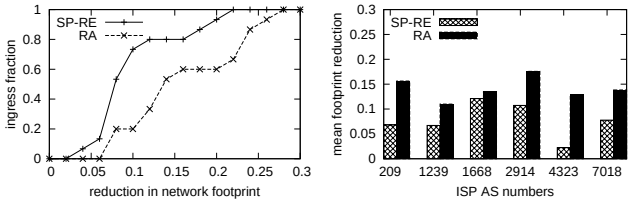
Evaluation strategy. Given a trace, synthetic or real, and a network topology, we compute aggregate redundancy profiles using the approach described in Sec 3.3.2. We compute routes according to the redundancy-aware algorithms of Sections 3 and 4. We use realistic ISP topologies (with link latencies) from Rocketfuel [23].

In all cases, we compare redundancy-aware routing algorithms, denoted as "RA", and traditional shortest path routing with hop-by-hop redundancy elimination, denoted as "SP-RE", against traditional routing without redundancy elimination, denoted as "SP".



(a) $\rho_{overall} = 0.5$ (b) $\rho_{intra} = 0$

Figure 5: Intra-domain, single-ingress (Seattle, WA) for ATT (AS7018). Link capacities are unlimited.



(a) CDF across multiple ATT ingresses (b) View across multiple ISPs

Figure 6: Impact of topology in the intra-domain setting.

6.1 Benefits in the Intra-Domain Setting

We first evaluate the benefits of redundancy-aware routing in an *uncapacitated* intra-domain setting with traffic from a *single ingress*.

Synthetic traces. First, we employ a variety of synthetic traces to examine the benefits. In Figure 5, we compare SP-RE and RA against SP for traffic originating from the Seattle PoP in the ATT network topology (AS7018). In Figure (a), we present the reduction in network footprint under a range of different inter- and intra-flow redundancy proportions (i.e. ρ_{inter} and ρ_{intra} values), but the overall redundancy fraction remains unchanged ($\rho_{overall} = 0.5$). From Figure 5(a), we note that the benefits of redundancy elimination in general are quite significant: the network footprint reduces 27-50% with RA and 6-50% with SP-RE.

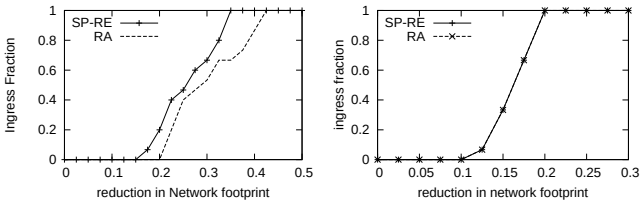
We also note that RA offers substantial reduction in network footprint compared to SP-RE. In particular, when redundant content is duplicated across multiple destination PoPs (i.e., as $\rho_{inter} \rightarrow 1$), RA is significantly better (27% reduction due to RA compared with 6% due to SP-RE). At the other extreme, when most duplicated packets travel between the same source-destination pair (i.e. as $\rho_{intra} \rightarrow 1$), the benefits of RA relative SP-RE start to diminish, and RA eventually becomes identical to SP-RE.

In Figure 5(b) we vary $\rho_{overall}$ while keeping $\rho_{inter} = 1$. At a low fraction of the overall redundancy proportion ($\rho_{overall} < 0.1$), RA and SP-RE are essentially indistinguishable. When $\rho_{overall} \geq 0.2$, we see that RA offers significant benefits compared to SP-RE: RA can reduce the network footprint but a further 6-20%.

These observations indicate that redundancy awareness generally offers substantial improvements in network utilization under a wide range of possible redundancy profiles, compared both to current routing and to simple redundancy elimination.

Next, we analyze the impact of topology on the benefits of redundancy awareness. In Figure 6(a), we plot a distribution of the benefits due to RA and SP-RE as we change the ingress PoP in the ATT network. We set $\rho_{overall} = 0.5$ and $\rho_{inter} = 1$. We see that the benefits from both RA and SP-RE vary with the ingress PoP, but, in general, RA offers significant improvements over SP-RE. While SP-RE alone can reduce the footprint by 2-22%, the benefits of RA are even better: between 6% and 27%.

In Figure 6(b), we compare how the mean improvements in net-



(a) High redundancy trace (50%) (b) Low redundancy trace (17%)

Figure 7: Trace-based analysis for SprintLink.

work footprint vary across 6 different tier-1 ISP topologies, where the mean is computed over all PoPs in an ISP. We see that the mean improvement is between 2 and 12% for SP-RE and between 11% and 17% for RA. We note that in some situations, e.g., AS1668, which has a very sparse topology, the benefits from RA are marginal compared to SP-RE. For sparse networks, simple redundancy elimination is sufficient to bring down the network footprint.

Real traces. Next, we analyze the benefits of RA and SP-RE using real packet traces. We conduct our analysis over the network topology of SprintLink (AS1239). Our approach is to vary the origin PoP of the packet trace and study the benefits of RA and SP-RE assuming *all* packets in the trace are destined for SprintLink's customers. To model where the intra-domain traffic would exit SprintLink's network, we map the top 2500 destination prefixes in the traces to a US city using "undns" [23] and traceroute. We then map the city to the nearest SprintLink PoP. We assume that each router has a 2GB packet store.

Our trace-based analysis is representative of a real-world application of redundancy elimination and redundancy aware routing. Using the traces, we first compute the redundancy profiles (Section 3.3.2). Then, we compute redundancy-aware routes, route packets in the trace on the computed paths, and simulate redundancy elimination on each router (Section 2.1).

In Figure 7(a), we show the distribution (CDF) of the improvement in network footprint when different PoPs in AS1239 are chosen as the ingress. Here, we use a trace of the high-volume /24 prefix, which had a redundancy proportion of nearly 50%. We see that both SP-RE and RA offer substantial reductions in network footprint. In particular, we note that the benefit from RA is $> 40\%$ for roughly 10% of the ingresses. One of these ingresses was Seattle; RA aggregates traffic originating from Seattle and destined for NYC, Boston and Dallas (which receive 36% of traffic in total) together with destined for Chicago (which receives 40% of the traffic), and routes all traffic on the single Seattle-Chicago link.

We also conducted trace-based analysis of a full packet trace of the University access link (Figure 7(b)). The aggregate redundancy proportion in this trace was 17%. We observe little difference between SP-RE and RA. This is because, as shown in Figure 4(a), a very small fraction of the content in this case is duplicated across PoPs. We do note that redundancy elimination was generally very beneficial, resulting in 10-20% reduction in the network footprint.

Benefits in intra-domain Traffic Engineering (TE). Next, we show that redundancy elimination and redundancy-awareness can help an ISP better meet its network-wide TE goals. We use synthetic traces in this analysis. In contrast with the earlier study, we now impose capacity constraints on network links. In particular, given a Rocketfuel ISP topology, we annotate links with capacities chosen uniformly at random from {2.5, 10}Gbps.

We generate one synthetic trace per PoP in the ISP topology. For simplicity, the traffic from all PoPs has the same $\rho_{overall}$ and ρ_{inter}. However, each trace differs in the aggregate traffic volume, which is chosen to be proportional to the population of the PoP's location.

Given the traffic proportions, we compute (redundancy-*agnostic*) routes which minimize the maximum link utilization in the network.

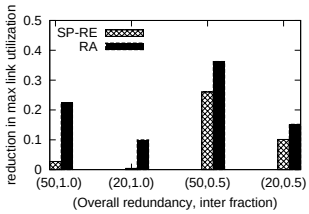
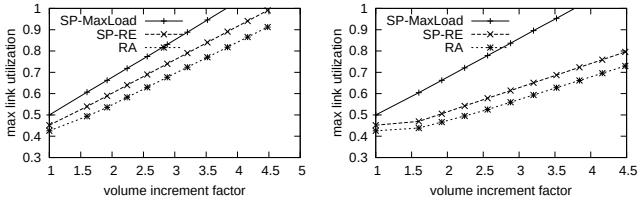


Figure 8: Traffic engineering with different redundancy profiles (ATT network). The baseline for comparison is SP-MaxLoad.



(a) $\rho_{overall} = 0.5, \rho_{intra} = 0.25$ (b)$\rho_{overall} = 0.5, \rho_{intra} = 0.75$

Figure 9: A simulated flash crowd, where traffic from Chicago in ATT increases suddenly. The original redundancy profile was $\rho_{overall} = 0.2$ and $\rho_{intra} = 0.5$.

We refer to this approach as *SP-MaxLoad* (We abuse notation here: the inter-PoP paths may not be *shortest* in terms of latency). The aggregate volumes between PoPs in the network are then scaled up so that the maximum link utilization is 80%.

We first employ hop-by-hop redundancy elimination along the routes obtained above. In Figure 8, the bars labeled "SP-RE" show the resulting improvement in the maximum link utilization (again, we abuse notation here). We see that redundancy elimination can improve maximum link load when coupled with traditional traffic engineering: the improvement ranges between 1% when $(\rho_{overall}, \rho_{inter}) = (0.2, 1)$, and 25% when $(\rho_{overall}, \rho_{inter}) = (0.5, 0.5)$. The bars labeled "RA" show the benefits of employing redundancy-aware routes. We note that the improvements in this case are very substantial: the maximum link load is 10%-37% lower. Such heavy reduction in the maximum link utilization is extremely valuable to ISPs because it creates additional capacity within the networks and allows them to meet service-level objectives more effectively.

Sudden traffic variations. We now examine how our approaches can mitigate the impact of sudden spikes in traffic load as might occur during flash crowd events. We use the same set-up as above for simulating the flash crowd: We start with a network-wide trace where we set $\rho_{overall} = 0.2$ and $\rho_{inter} = 0.5$ for traffic from all ingresses. The traffic volumes are such that the maximum link utilization due to SP-MaxLoad is 50%. Given this set-up, we compute redundancy-aware network routes.

We then make a sudden change - a factor of f increase overall - to the volume of traffic originating from an ingress picked at random. We also change the redundancy profile, i.e. $\rho_{overall}$ and ρ_{inter}, of the traffic from the ingress. However, we *do not* recompute new redundancy-aware routes; instead, we study how routes which match the *stale* profiles perform.

In Figure 9, we show the results from two different flash crowd simulations. In both cases, we increase $\rho_{overall}$ to 0.5.In the first case, the flash crowd causes a *higher* fraction of duplicate packets to be distributed *across multiple* destinations; in particular, ρ_{inter} increases from 0.5 to 0.75. The performance of the different schemes is shown in Figure 9(a). We see that redundancy elimination, whether coupled with redundancy-awareness or not, offers clear benefits in terms of mitigating the impact of the sudden increase. When the traffic volume increases by $f = 3.5$X, the maximum link load due to SP-RE is 85% and that due to RA is 75%. Without any form of redundancy elimination (SP-MaxLoad), the maximum load is 95%.

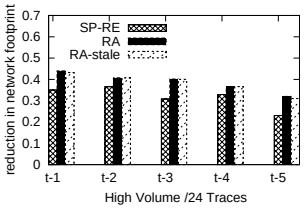
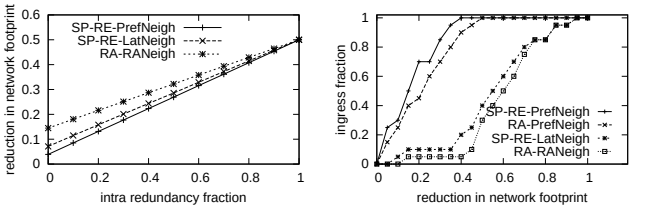


Figure 10: Impact of stale redundancy profiles.



(a) Chicago as ingress, $\rho_{overall} = 0.5$ (b) $\rho_{overall} = 0.5, \rho_{inter} = 1$, all PoPs

Figure 11: Reduction in network footprint for the Inter-domain local approach in ATT's Network topology.

We analyzed another flash crowd situation where a *smaller* fraction of bytes are duplicated across destinations compared to the baseline situation (ρ_{intra} increases from 0.5 to 0.75). The results in this case are shown in Figure 9(b). We see that the benefits from redundancy elimination are much better than the first situation: the maximum link loads at $f = 3.5$X are brought down to 61% with RA and 68% with SP-RE. The difference between RA and SP-RE is small because most of the redundancy is confined to traffic within ingress-egress pairs, and thus redundancy-aware route construction is not highly beneficial compared to shortest-paths.

Staleness of profiles. We conduct a separate analysis of the impact of employing routes computed using stale redundancy profiles. We use real traces corresponding to the high volume /24 prefix in this analysis. We assume that the traffic in the trace originates at the Chicago PoP in the SprintLink Network (AS1239). We focus on SprintLink's intra-domain routes for this traffic. We compute routes that were optimal for the trace collected at a certain time, and evaluate the network footprint when using these routes for the traffic in 5 traces which were collected 10, 20,..., 50 minutes after the original trace. Figure 10 shows the network footprints from employing the stale redundancy-aware routes (RA-Stale) to route the traffic in these 5 traces. We see that RA-Stale is very close to the optimal (wherein the redundancy-aware routes are computed using current profiles; denoted by RA), and significantly better than SP-RE. We repeated the analysis for traces collected at other times of the day and observed that RA-Stale always offered reasonable performance. We also changed the source PoP for the traffic to see if there were topology-related biases in our observations. However, the performance of RA-Stale was consistently good (See our technical report [5] for full results). While a more thorough analysis of the impact of staleness is necessary, these observations seem to indicate that redundancy-aware routes computed at a certain time will continue to offer reasonable performance for few 10s of minutes.

6.2 Benefits in the Inter-domain Setting

We now present a comparison of the benefits of redundancy aware routing, simple redundancy elimination, and traditional routing in the inter-domain context. We assume link capacities are unconstrained. We first consider an ISP's local approach for inter-domain traffic originating from a single PoP in the ISP. Our baseline for comparison is BGP-based choice of the next-hop AS, with early exit routing to the next-hop's peering location.

In Figure 11, we present the reduction in network footprint for the ATT network (AS7018). The footprint is computed over ATT's internal and peering links. We consider inter-domain traffic originating at a single ATT PoP. We use synthetic traces. The destination ASes for the inter-domain traffic are modeled along those observed in real traces: we identify the top 75 destination ASes by volume in the packet traces for which ATT only has peer-announced routes. We assume that the traffic volume to these destinations follows a Zipf distribution. We use Rocketfuel maps to obtain locations where ATT peers with its neighbors. We used ATT's public BGP tables to obtain the preferred next hop ASes for each destination AS.

For the results shown in Figure 11(a) the traffic originates from the Chicago PoP in the ATT network. We first examine the curve labeled "SP-RE-PrefNeigh" which corresponds to ATT using *early-exit* routing internally to reach the *BGP-preferred* next hop neighbor for a destination. Simple redundancy elimination is then employed on all network links. We note that even this simplistic application of redundancy elimination offers substantial reduction in network footprint, ranging between 4-50% for a trace where $\rho_{overall} = 0.5$.

We also study "RA-PrefNeigh", which corresponds to ATT routing via the *BGP-preferred* next hop neighbor, but using a peering location which is selected in a *redundancy-aware* manner. This is not shown in Figure 11(a) since it offered very similar performance as SP-RE-PrefNeigh. The similarity arises because ATT is connected to most of its peers in Chicago, and the exit points chosen by RA-PrefNeigh are the same as that due to early exit routing.

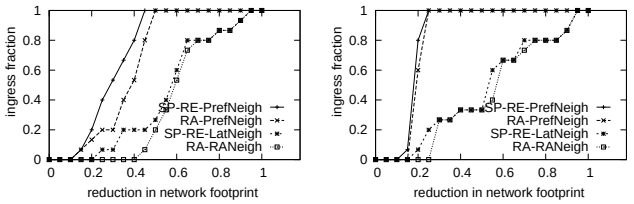
Next we focus on the curve labeled "RA-RANeigh" where, in a departure from traditional BGP route selection, ATT makes a selection of *both* the *next hop* neighbor *and exit point* in a redundancy-aware manner using the algorithm outlined in Section 4.1. We see that by making both choices in a redundancy aware fashion, ATT improves the load on its internal and peering links by 0-11% compared to redundancy-agnostic next hop AS selection (i.e. RA-PrefNeigh, which is identical to SP-RE-PrefNeigh).

In Figure 11(b) we plot the distribution of the reduction in network footprint as we vary the ingress PoP in the ATT network. We see that the benefits of redundancy awareness are very high: in some cases, RA-RANeigh reduces the network footprint by $> 85\%$.

Note that in contrast to traditional BGP routing, an ISP using RA-RANeigh may select a peer which has the nearest exit point as the preferred next hop for a destination. For example, say that peer A_1 is ATT's BGP-preferred next hop for a destination prefix P and A_1's closest exit point is 10ms away from the source PoP. Another peer A_2 which has also announced route to P has an exit point which is just 5ms away. RA-RANeigh may prefer A_2 over A_1 because choosing lower latency internal paths helps RA-RANeigh reduce the overall network footprint significantly.

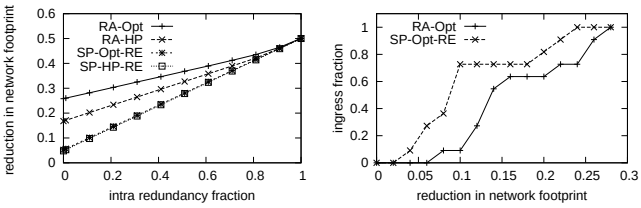
Next, we examine the benefits of an ISP choosing the next-hop AS using the following latency-driven approach: among all peers who have announced a route to a destination, pick the one with the nearest exit point. The key difference between this and RA-RANeigh is that the selection of the inter-domain route is not made in an explicit redundancy-aware manner. We analyze the performance of the above latency-based approach to inter-domain route selection and show the results using the curve labeled "SP-RE-LatNeigh" in Figure 11. Two key points emerge from comparing SP-RE-LatNeigh against RA-RANeigh: For nearly 20% of the ingresses, the performance of SP-RE-LatNeigh is close, if not identical, to RA-RANeigh; In these cases RA-RANeigh selects neighbors with nearest exit points as the next hops just like SP-RE-LatNeigh does.

For the remaining ingresses, however, selecting neighbors purely on the basis of the latency to the exit point seems to be quite sub-optimal. Two factors contribute to the superiority of RA-RANeigh here: (1) First, selecting a peer with a *farther* away exit point as the preferred next hop for a destination may offer better opportu-



(a) High redundancy trace (50%) (b) Low redundancy trace (17%)

Figure 12: Trace-based analysis for ATT.



(a) Chicago as source, $\rho_{overall} = 0.5$ (b) $\rho_{overall} = 0.5, \rho_{inter} = 1$

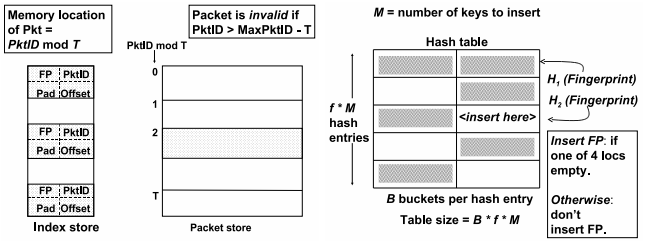
Figure 13: Inter-domain routing, Cooperative approach between ATT and SprintLink networks.

nities for aggregating redundant content. For instance, say a peer A_2 has announced a route to some prefix P and has an exit point located 15ms away from the source PoP. Another peer A_1 has also announced a route to P, and has a closer exit point located just 5ms away. Aggregating on the peering link to A_2 all inter-domain traffic to prefix P, together with traffic to other prefixes for which only A_2 has announced routes, can significantly reduce the overall network footprint. In contrast, simply using A_1 to send traffic to P may not offer similar benefits. (2) Second, RA-RANeigh attempts to aggregate traffic to destinations which share redundant content onto the same peering links. In contrast, SP-LatNeigh may aggregate destinations across which content is seldom replicated.

Trace-Based Analysis. In Figure 12, we present the results from our evaluation of the inter-domain local approach using real packet traces. In Figure 12(a), we present the results for the traffic traces from the high volume /24 prefix, where the overall redundancy proportion was 50%. We observe very significant reductions in network footprint from employing redundancy elimination, irrespective of whether redundancy-aware routing is used or not. Also, as before, we note that the difference between SP-LatNeigh and RA-RANeigh is quite substantial for more than 50% of the ingress PoPs. In Figure 12(a), we present the results for a full trace of the University access link, where the redundancy proportion was observed to be 17%. In this case, there was very little duplication of content across destinations, and hence the benefits from redundancy-awareness are low relative to simple redundancy elimination.

Cooperative Approach. In Figure 13(a), we examine the benefits from cooperation between ISPs in computing redundancy aware routes between each other. We employ synthetic traces in our analysis. We focus our analysis on the Sprintlink and ATT networks both of which are tier-1 ISPs. They peer with each other at multiple locations. We consider traffic originating from Chicago in ATT and going both to PoPs in SprintLink and PoPs in ATT. We assume that 80% of all traffic originating at Chicago in ATT is inter-domain traffic, while 20% goes to intra-domain destinations. We considered other traffic distributions, but the results were qualitatively similar.

As before, we compare RA and SP-RE against SP. We consider two variants of each approach, namely Opt (for Optimal) and HP (for Hot Potato). These two variants model a network's exit point selection for routing inter-domain traffic into its neighbor. When using Opt, a network computes exit points for the inter-domain traffic destined to its neighbor in a cooperative, globally optimal way. In SP-Opt-RE, the cooperative routes minimize the sum total latency of all network paths (inter and intra-domain). In RA-Opt the cooperative routes minimize the network footprint across both networks; RA-Opt is exactly the algorithm we descried in Section 4.2. In early-exit or hot potato routing (HP), each network tries to optimize its own local objective. In SP-HP-RE, each network uses early exit routing into the other network. In RA-HP, each network selects peering locations which minimize its *own* network footprint. The baseline for comparison is SP-HP. Our metric of comparison is the network footprint computed over both ISP networks.



(a) Packet and FP stores (b) Fingerprint Hash table

Figure 14: Architecture of the fingerprint and packet stores.

Comparing RA-Opt with SP, we see that the reduction in network footprint is very impressive, ranging between 28% and 50%. Also, we note that SP-Opt-RE is not much better than SP-HP-RE. This is because early-exit paths origination from the Chicago PoP in ATT already have close-to-optimal latencies to the PoPs in SprintLink. More importantly, we observe that SP-Opt-RE is inferior compared to RA-HP. This further underscores the importance of redundancy-aware route computation in reducing the network-wide utilization.

In Figure 13(b), we show a distribution of the reduction in network footprints when different ATT PoPs are chosen as the sources of the inter-domain traffic. As with our prior analyses, we see that redundancy elimination in general is vastly beneficial, but redundancy awareness offers greater overall improvement.

Evaluation Summary. Our extensive study has shown the vast benefits of network-wide support for redundancy elimination, and in particular, of changing network routing to be redundancy-aware. We see that the impact of traffic on ISP network resources can be reduced significantly. This is especially useful to control link loads in situations of sudden overload. Finally, using routes computed on the basis of stale profiles does not seem to undermine the benefits of our approaches. Of course, the initial cost of deployment of redundancy elimination mechanisms on multiple network routers will be quite high. However, our analysis shows that the long-term benefits of a wide-spread deployment are high enough to offset the cost.

Note that we assumed throughout that each router carries fully decoded packets internally. But our proposals can be extended so that routers switch smaller encoded packets (perhaps combining multiple related small packets into a larger packet), with decoding/reconstruction occurring only where necessary. This can help overcome technology bottlenecks *inside* routers, in addition to saving bandwidth on links.

7. IMPLEMENTATION

In this section, we examine some of the key challenges that may hinder the deployment of redundancy elimination mechanisms on fast routers. We offer preliminary solutions to the challenges. We evaluate the trade-offs introduced by our solutions via a software implementation based on the Click modular router [18]. Our implementation extends the base algorithm of Spring et. al [24].

An important bottleneck in performing redundancy elimination at high speeds is the number of *memory accesses* required during the various stages of redundancy elimination, such as on-the-fly lookup, insertion, deletion, and encoding the redundant region in a packet.

A second challenge is controlling the amount of memory required to store the key data structures at routers, namely the fingerprint and the packet stores. Our implementation is focused on developing *memory efficient ways to organize and access the data structures*. These issues have not been considered carefully in prior work.

Another key component is the computation of the hash function to obtain fingerprints for each packet. Rabin finger-prints used in [24] are well-suited for high-speed implementation. In particular, because Rabin fingerprint computation relies on use sliding hashes, the fingerprints can be computed in parallel with CRC checks, even as the bytes in a packet arrive into a router.

7.1 Packet Store

The layout of the packet store in our implementation is showed in Figure 14(a). We implement the packet store as a FIFO buffer. In particular, we use a *circular buffer* with a maximum of T fixed-size entries. With FIFO buffering, the oldest packet in the packet store is evicted when there is no room to insert a new packet. We considered using other policies for eviction (such as Least-Recently-Used), but a preliminary study of these policies showed that FIFO offers nearly the same performance (in terms of the amount of redundant content identified), but is simpler to implement (See [14] for details).

We use a global variable called "MaxPktID" (4B) to aid packet insertions and deletions. This is incremented before inserting a new packet. The current value of MaxPktID is assigned to a variable *PktID* which becomes a *unique identifier* for the packet. The packet itself is stored at the location $PktID \% T$ in the store. Thus PktID also indicates the starting memory address of the packet's location.

We take a short digression and describe the fingerprint store to provide context for the rest of the design of the packet store. The fingerprint store holds *meta-data* for representative fingerprints, which includes the fingerprint itself, the unique ID for the packet (i.e., the PktID) referred to by the fingerprint, and the byte offset in the packet where the region represented by the fingerprint starts.

When the packet store is full, we simply overwrite the new packet at the tail of the circular store. We must also ensure that the fingerprints pointing to the evicted old packet are invalidated. Rather than invalidate the associated fingerprints one-by-one (which can require a large number of memory accesses), we can leverage the MaxPktID variable and the PktID stored in the fingerprint meta-data: To see why, note that if $(PktID < MaxPktID - T)$, then the packet has been evicted and thus the fingerprint is invalid.

The fingerprints for a new packet are hashed into random locations in the fingerprint store (discussed next).

7.2 Fingerprint Store

The fingerprint store must support fast insertions and efficient lookups when checking for redundant content. To support these properties, we implement the fingerprint store as a hash table.

If we use standard hash table implementations, then we will need the fingerprint table to be very sparse to avoid collisions, and ensure fast inserts and lookups. A quick calculation shows that, at OC48 speeds, if we store 10s worth of packets (i.e., a 3GB packet store), the fingerprint table must be $>$ 20GB in size. Even at this large size, there is no real guarantee of collision-freeness and hash chaining.

To improve hash table storage efficiency while still ensuring $O(1)$ lookups and inserts, we use *CuckooHash* [10] to design the fingerprint store. The CuckooHash-based design is illustrated in Figure 14(b). Each hash entry is divided into B buckets. Each bucket stores a key, which is a fingerprint entry in our case. A set of $k \leq 2$ independent hash functions are used during insertion of a representative fingerprint into the hash table: If any of the $k \times B$ locations are found empty, the fingerprint is inserted at the first empty location. If no bucket is empty, the fingerprint is simply not inserted (in this case, we consider the insertion to have "failed").

NumHashes → f ↓	1	2
1.2	15.1%	11.5%
1.5	12.4%	7.8%
2	9.5%	4.6%

(a) $B = 1$

NumHashes → f ↓	1	2
1.2	5.0%	0.06%
1.5	3.4%	0.02%
2	2.0%	0.003%

(b) $B = 2$

Table 1: Fraction of fingerprints that we fail to insert.

In Table 1, we explore the trade-offs between hash table size, the number of buckets B, and the number of hash functions k. In particular, we examine the fraction of representative fingerprints that we fail to insert for a real packet trace selected at random. The more fingerprints we fail to insert, the lesser the extent of redundancy we can identify and remove. The hash table size is a factor f larger than the target number of fingerprints we want to store; Note that the target number is fixed (approximately) for a given packet store size and a given number of representative fingerprints per packet; We fix the number of representative fingerprints at 16 per packet.

From Tables 1(a) and (b), we note that using multiple hash buckets offers better performance irrespective of the number of hash functions used. We see from Table 1(b) that for $k = 1$ and $f = 2$, we fail to insert just 2% of the fingerprints. When $k = 2$ hash functions are used, the probability of failure is essentially zero for any f. Note, however, that we incur twice as many memory accesses (during lookups) when using two hash functions instead of one.

Our implementation uses a single hash function, two buckets, and $f = 2$, as this offers a reasonable middle-ground in terms of the redundancy identified and memory accesses per packet. This design also brings the fingerprint store size to $\leq$ 1.5GB at OC48 speeds.

7.3 Encoding

The approach we use to encode duplicated chunks in an incoming packet is the same as that used by Spring et. al [24]: For each duplicated byte string found, we remove the matched region from the incoming packet and replace it with a "shim layer" containing the matching packet's PktID (4B), 2B each for the starting byte positions of the current and the matching packet, and 2B for the length of the match. When a packet matches multiple fingerprints, we store one shim per match, ensuring that the matching byte regions identified by each shim are non-overlapping.

In summary, memory accesses are incurred by routers during insertion of a packet and its representative fingerprints, and during retrieval of matching packets to perform encoding. Since we use 16 representative fingerprints per packet (by default) and not all packets see matches, the former set of accesses are likely to dominate the per packet memory access overhead. Note that the number of memory accesses grows with the number of finger-prints stored per packet, but so does the amount of redundancy identified.

7.4 Benchmarking Results

We have implemented packet-level redundancy elimination using the aforementioned data structures in the Click modular router [18]. Our current implementation runs on a 1.8GHz AMD Opteron processor with 8GB of RAM (64-bit Linux version 2.6.9). We configured the packet store to use 400MB of memory. This results in a 200MB fingerprint store when using 16 fingerprints per packet. Hash computation, packet and fingerprint insertion, and encoding are all done serially in our software implementation.

We evaluated the throughput performance of our implementation using real packet traces. To estimate the maximum possible throughput, we read the packet trace from main memory instead of receiving packets over the network (to avoid delays due to interrupt processing). Our implementation achieved a throughput of

Max FPs per Pkt	Overall speed	No Click	No Click or Hashing	Updated machine No Click or Hashing	Redundancy percentage
32	0.67	0.71	1.0	1.39	17.3%
16	1.05	1.17	1.34	1.93	15.8%
10	1.15	1.3	1.62	2.31	14.67%

Table 2: Throughput of software implementation (in Gbps) for a random packet trace.

1.05Gbps on average across multiple packet traces.

We profiled Click's processing overhead and, upon accounting for it, found that we achieved a throughput of 1.17Gbps (Table 2).

Next, we examine how memory access latencies affect the performance of our implementation. To do this, we precomputed the finger prints for all packets to avoid hash computation. The throughput due to the rest of the components of our implementation is shown in Table 2. This includes fingerprint insertions, packet insertions, packet retrievals, match region computations, and encoding the match regions. We ran microprocessor performance benchmarks to confirm that the software is memory-bound. We see that when using 16 FPs per packet, our implementation runs at 1.4Gbps.

Memory benchmarks for our test machine showed read/write latencies to be 120ns per access. In contrast, today's high-end DRAMs operate at 50ns or faster. To understand the likely improvement with faster DRAMs, we ran our implementation on an updated machine with a 2.4GHz processor running a 32-bit Linux (see Table 2). The memory latency on this desktop was benchmarked at 90ns. We consistently observed a speed-up of 1.4X: with $\leq$ 16 fingerprints, we were able to obtain close to 2Gbps. With fewer fingerprints (10 per packet, which resulted in an 18-22% drop in redundancy proportion identified), we obtained 2.3Gbps. Thus with 50ns DRAM latencies, it seems likely that we can easily reach OC-48 speeds in software.

8. OTHER RELATED WORK

We discussed past studies most relevant to our work in Section 2. Below, we discuss a few other pieces of related work.

Several past studies have examined the benefits of cooperative caching of Web objects [25, 11]. These studies are similar in spirit to our work, but we take the much broader view of making redundant content elimination as a universal router primitive.

Our redundancy-aware routing algorithms are somewhat similar to multicast routing algorithms [7]. The algorithms we develop essentially build efficient multicast distribution trees. The shape and structure of our trees are influenced by destinations which observe significant overlap in bytes accessed. In contrast, multicast tree construction simply tracks the location of multicast participants.

Recent traffic engineering proposals have tried to improve the responsiveness to real time traffic variations [15]. While we leave a full comparison of our techniques against these approaches for future work, we do believe that the benefits of the recent approaches can be further enhanced by making them redundancy-aware.

9. CONCLUSIONS

In this paper, we explored the implications of deploying packet-level redundant content elimination as a primitive service on all routers. Using real packet traces as well as synthetic workloads, we showed that applying redundancy elimination on network links can reduce resource utilization by 10-50% in ISP networks. However, the network-wide benefits can be further enhanced by modifying network protocols, in particular, the routing protocols, to leverage link-level redundancy elimination. We presented and analyzed a suite of redundancy-aware intra- and inter-domain routing protocols. We showed that they offer ISPs much better control over link loads, a great degree of flexibility in meeting traffic engineering objectives, and greater ability to offer consistent performance under sudden traffic variations. We have developed a software prototype of a high-speed packet-level redundancy elimination mechanism. Our implementation uses simple techniques to control the amount of memory and the number of memory accesses required for redundancy elimination. Our implementation can run at OC48 speeds. Hardware implementation speeds are likely to be much higher.

Our focus was on studying the benefits in the context of a universal deployment. However, our redundancy-aware techniques can be applied to limited-scale partial deployments of redundancy elimination across specific network links (e.g. across cross-country intra-domain links, or congested peering links).

Of course, deploying redundancy elimination mechanisms on multiple network routers is likely to be expensive to start with. However, we believe that the significant long term benefits of our approaches offer great incentives for networks to adopt them.

Acknowledgments. We wish to thank the following people for their advice: Fred Baker, Paul Barford, Mike Blodgett, Perry Brunelli, Paul Francis, Bruce Davie, Randy Katz, George Varghese, Jia Wang and Ty Znati. We thank the anonymous Sigcomm reviewers whose comments helped improve our paper. This work was supported in part by NSF grants CNS-0746531, CNS-0626889 and CNS-0435382.

10. REFERENCES

[1] Netequalizer Bandwidth Shaper. `http://www.netequalizer.com/`.
[2] Packeteer WAN optimization solutions. `http://www.packeteer.com/`.
[3] Peribit WAN Optimization. `http://www.juniper.net/`.
[4] Riverbed Networks. `http://www.riverbed.com`.
[5] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker. Packet Caches on Routers: The Implications of Universal Redundant Traffic Elimination (Extended Version). Technical Report 1636, UW-Madison, June 2008.
[6] B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *Infocom*, 2000.
[7] T. Ballardie, P. Francis, and J. Crowcroft. Core based trees (CBT). *SIGCOMM Comput. Commun. Rev.*, 23(4):85–95, 1993.
[8] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and implementation of RCP. In *NSDI*, 2005.
[9] B. Davie and Y. Rekhter. *MPLS: technology and applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.
[10] U. Erlingsson, M. Manasse, and F. McSherry. A cool and practical alternative to traditional hash tables. In *WDAS*, 2006.
[11] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: a scalable wide-area Web cache sharing protocol. In *ACM SIGCOMM*, 1998.
[12] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. In *Infocom*, 2002.
[13] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A clean slate 4D approach to network control and management. *SIGCOMM Comput. Commun. Rev.*, 35(5):41–54, 2005.
[14] A. Gupta, A. Akella, S. Seshan, S. Shenker, and J. Wang. Understanding and Exploiting Network Traffic Redundancy. Technical Report 1592, UW-Madison, April 2007.
[15] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: responsive yet stable traffic engineering. In *ACM SIGCOMM*, 2005.
[16] U. Manber. Finding similar files in a large file system. In *USENIX Winter Technical Conference*, 1994.
[17] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: existing techniques and new directions. In *ACM SIGCOMM*, 2002.
[18] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek. The Click modular router. *SIGOPS Oper. Syst. Rev.*, 33(5):217–231, 1999.
[19] A. Muthitacharoen, B. Chen, and D. Mazières. A low-bandwidth network file system. *SIGOPS Oper. Syst. Rev.*, 35(5), 2001.
[20] M. Rabin. Fingerprinting by Random Polynomials. Technical report, Harvard University, 1981. Technical Report, TR-15-81.
[21] M. Roughan, M. Thorup, and Y. Zhang. Performance of estimated traffic matrices in traffic engineering. In *ACM SIGMETRICS*, 2003.
[22] S. Singh, C. Estan, G. Varghese, and S. Savage. Automated worm fingerprinting. In *OSDI*, 2004.
[23] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP Topologies with Rocketfuel. In *ACM SIGCOMM*, 2002.
[24] N. Spring and D. Wetherall. A protocol-independent technique for eliminating redundant network traffic. In *ACM SIGCOMM*, 2000.
[25] A. Wolman et al. On the scale and performance of cooperative Web proxy caching. In *ACM Symposium on Operating Systems Principles*, 1999.