

Man-In-the-Middle Anycast (MIMA): CDN User-server Assignment Becomes Flexible

Jeffrey Lai

School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
jeff.lai@ecs.vuw.ac.nz

Qiang Fu

School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
qiang.fu@ecs.vuw.ac.nz

Abstract—Within Content Delivery Networks (CDNs), users are typically geographically load-balanced across multiple servers in order to provide better performance and locality – users are assigned to the content servers close to them. One approach to user-server assignment is the use of IP anycast, where all the content servers use the same IP address. A user request is then routed to the server closest to the user, determined by the routing protocols such as BGP. However, there are problems associated with this anycast-based approach. IP anycast is generally incapable of updating or redirecting users to a different (better) server without breaking already established TCP sessions. Moreover, the CDN operators do not have much control on where to redirect the users – it is completely controlled by Internet routing. In this paper we present our Man-In-the-Middle Anycast (MIMA) architecture, a novel anycast-based design that leverages Network Functions Virtualization (NFV) and Software Defined Networking (SDN) techniques to enable flexible and efficient user-server assignment. We demonstrate that the MIMA architecture is capable of performing flexible user-server assignment and offloading during times of high demand, such as flash crowd events that are becoming more common in a media-focused Internet. These capabilities offered by MIMA provide CDN operators a higher degree of flexibility in network management and content provisioning by enabling flexible user-server assignment.

Index Terms—Anycast, CDN, NFV, SDN, user-server assignment

I. INTRODUCTION

Streaming video is becoming a larger proportion of internet traffic with each passing year. Video data is projected to take up 80% of consumer Internet traffic in 2019, with over 70% of that being delivered via commercial CDNs [1]. High Definition (HD) video streams are also becoming widely available, with popular streaming services such as Youtube and Twitch readily providing HD content. Due to the nature of this content consumption, traditional web caches are becoming unsuitable for caching this kind of traffic. To handle such traffic behaviour and demands, content is usually now cached and load-balanced by CDNs via large, coordinated infrastructures across the Internet [9]. Typically, these CDNs rely on distributed content caches across the world in multiple locations, so as to spread out the effects of global demand and reduce latency. A typical example is the Akamai network [11],

with thousands of servers in just as many locations around the world.

In order for these CDNs to operate, they must perform some kind of user-server assignment in which user requests are assigned to an ‘optimal’ CDN cache surrogate. Typically, CDNs perform these assignments by using a DNS-based or an anycast-based strategy. While these strategies have performed fine for now, their innate design leaves them unable to leverage the growing capabilities of current server and networking technologies. As an example: in the scenario that a CDN provider needs to offload server traffic to a different datacentre, DNS based methods must rely on clients to update their DNS caches and records before they can access updated server resources, and anycast TCP streams will have to break and get swapped across servers mid-session. Neither of these scenarios lead to optimal performance for the end user and can lead to media blackouts in scenarios such as new releases of anticipated online media. A recent example is HBO GO’s crash during a Game of Throne Season 6 premiere [14]. With a growing number of users and servers in this CDN space, the ability to efficiently handle user-server assignment is becoming more important.

With flexibility becoming an increasingly important requirement for CDNs today due to flash crowd events and the large variety of content that needs to be served, it is clear that current networks need to be designed to accommodate such demands. Following this trend, residential ISPs are now also incorporating CDNs into their own networks, in order to ensure customer satisfaction in the content delivery arena. This is usually done in one of two ways: either by hosting in-network caches such as those provided by Google, Netflix, or Akamai [11], or by creating and administering a CDN across their own ISP network. It is clear that ISPs are searching for ways to enable a more flexible and robust approach to content delivery.

While this may be the case, CDN architectures have yet to catch up to the point where flexible user-server assignment can be performed, due in part to the innate shortcomings of current user-server assignment schemes mentioned above. To that end, we have designed a novel yet simple and practical Man-In-the-Middle Anycast (MIMA) architecture that combines the unicast streams with the client simplicity of anycast-based systems and

enables flexible user-server assignment without breaking existing TCP sessions.

By leveraging NFV techniques to perform flexible middlebox functionalities and dynamic capacity provisioning, our architecture can easily be implemented in current ISP-CDN collaborations, or used as part of a standalone CDN. In this paper we present the evaluations of our prototype implementation and demonstrate our architecture’s flexibility and capabilities in scenarios that current architectures are unable to handle.

The rest of our paper is structured as follows: Background information and related work on CDN design and operation is provided in Section 2. We describe our system design and requirements in Section 3. Our experimental setup and evaluation results are provided in Section 4. Finally, we discuss our results and present our conclusions in Section 5.

II. BACKGROUND AND RELATED WORK

With the popularity and large presences of current CDN providers, much research into their efficiency and operation has already been done. Our work is focused on developing an architecture that is capable of using a variety of network and server information and enables flexible user-server assignment without breaking existing TCP sessions. Our architecture does not rely on such information, it is merely capable of utilizing such information if present to enable better user-server assignment.

Recent research has shown that by augmenting a DNS-based user-server assignment architecture in an ISP-CDN collaboration, multiple network engineering goals can be achieved for an ISP-CDN collaboration [4]. However, relying on a DNS-based mechanism incurs a performance tradeoff due to the requirement that users need to actively perform DNS requests in order to receive updated server information and perform flexible user-server assignment, and a global mapping infrastructure (between server and user) needs to be maintained and updated in a timely manner. Indeed, our past research [7] indicates that it is incorrect to assume that users will regularly request updated server information, due to factors such as application behaviour and DNS caching.

Due to these shortcomings of DNS, an increasing number of operators are running anycast-based CDNs, for example, Cloudflare, Edgecast, CacheFly and Microsoft, to name a few [15]. Using TCP anycast is a technique that avoids the pitfalls of DNS caching and the need of a global mapping infrastructure, as each DNS request for content hosted on the CDN always returns the same server IP address. Due to the nature of IP anycasting though, this technique sacrifices fine-tuned control of load balancing across individual cache servers, instead relying on BGP routing to perform load-balancing geographically. That is, user-server assignment is controlled by Internet routing. In this case, when a server is overloaded it may withdraw its route and thus redirect traffic to the next closest server. However, this approach has two problems: 1) the next closest server may be overwhelmed by the large amount of traffic redirected to it, and 2) the existing TCP

sessions will break and have to be reestablished with the new server.

To gain the control of load balancing or redirection, some sort of use of cache server’s unicast address has to be adopted. While SCTP-based [13] extensions exist that bridge some parts of unicast and anycast TCP behaviour, it is yet to be widely adopted, and it requires active client participation to be of use when compared to a transparent CDN. For example, the “set address” option in SCTP allows an anycast-based server to specify its unicast address, and thus an SCTP client may connect to the server using its unicast address. Similarly in [16], a push-based approach is adopted that allows a client to use the unicast address of an anycast-based server.

PIAS [17] is an anycast overlay. In general, it follows the idea of Mobile IP [18], a server’s anycast address being home address and its unicast address being care-of address. A packet sent to the anycast address can be redirected to the unicast address of a server. However, as an anycast overlay PIAS requires a large number of proxies to be deployed. Moreover, when redirecting from one server to another, existing TCP sessions will break and have to be reestablished with the new server. PIAS does not allow flow-level redirection as the port numbers are used as part of anycast addresses. PIAS is a complicated system. The work in [15] shows that anycast-based CDNs generally perform well and ~20% of the clients are directed to sub-optimal servers. This means that the chance of needing redirecting a client to a different server is not high. Therefore, it is desirable to address the issue with a simple solution.

Another piece of work [19] investigates the use of Information Centric Networking (ICN) over OpenFlow/SDN to facilitate content delivery. ICN is a relatively radical approach for content delivery and worth a lot of research efforts. Our focus is on the traditional CDNs.

The most relevant published research to our work is Microsoft’s FastRoute [3] production network, which provides an example of an advanced anycast system that utilizes DNS techniques to maintain control over load balancing whilst still achieving anycast locality performance. FastRoute uses multiple rings/layers of cache servers. Each of the layers has a different anycast IP address. When a DNS query arrives, the IP address of a suitable layer is returned. That is, within the same layer, load-balancing or user-server assignment is handled by anycast. However, this re-incurs the tradeoff of relying on DNS for optimal load-balancing, and thus suffers the DNS caching problem [7]. While FastRoute is capable of offloading traffic to other cache nodes without breaking existing TCP sessions, it’s limited to offloading traffic from an outer to an inner ‘ring’ in order to avoid oscillation problems (offloading to each other between two nodes). This means that the capacity of the inner rings has to be overprovisioned. The native anycast within the same layer makes it difficult to dynamically change the number of servers within the same ‘ring’ in response to real-time load without causing TCP flapping, reducing the flexibility of their system. In contrast, our architecture addresses all these problems, while still offering anycast-based performance and simplicity.

MIMA also has the advantage of being designed from the ground up to make use of NFV technologies, providing extra flexibility in our deployment capabilities compared to FastRoute, while also allowing us to take advantage of dynamic cache provisioning. A recent networking trend is to virtualize middlebox functionality in order to enable higher levels of network scalability and flexibility [2][10]. By utilizing NFV, it is now possible for networks to flexibly instantiate the middlebox functionality they need, using only off the shelf hardware. Research by Sherry et al [12] shows that it is both feasible and beneficial to run middleboxes in a cloud environment, which suggests that using cloud resources to run parts of our MIMA architecture is a viable strategy. In a similar vein, current research shows examples of cache virtualization [4][6], which would provide CDN operators the ability to rapidly respond to current network conditions and requirements such as flash crowd events and changing peak load.

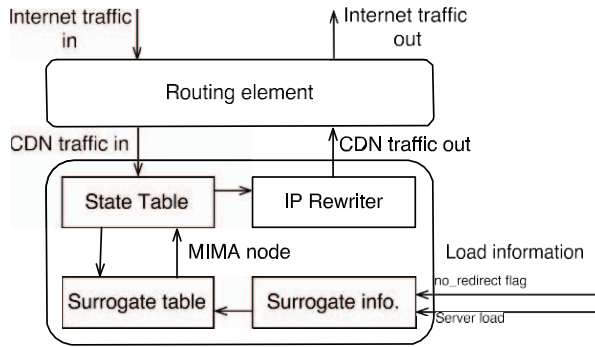


Fig. 1. Typical configuration of MIMA architecture.

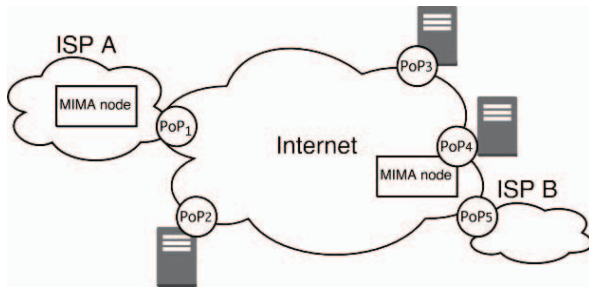


Fig. 2. Example MIMA placement: ISP network edge vs CDN network edge.

III. MAN-IN-THE-MIDDLE ANYCAST (MIMA)

In anycast-based CDNs it is difficult to perform user-server assignment based on factors such as network and server information, as the assignment is inherently controlled by Internet routing. If a cache server becomes overloaded, it is difficult to gradually direct traffic from that server to other servers. Simply withdrawing the route will terminate ongoing TCP sessions and can lead to cascading overloading of nearby cache servers. Moreover, anycast routing changes can cause ongoing TCP sessions to terminate. In order to address this

issue, we propose a Man-In-the-Middle Anycast (MIMA) user-server assignment architecture that combines the unicast streams with the simplicity of anycast-based systems.

The design goals of our MIMA architecture are to increase the flexibility of anycast-based user-server assignment by allowing the use of network and server information in a manner that is transparent to the user.

Figure 1 shows the MIMA architecture, which consists of two main components: Routing Element and MIMA Node. Figure 2 gives a general overview of how the MIMA architecture might be placed across the Internet. Like other anycast-based CDN architectures, MIMA provides an anycast IP address in response to an initial client DNS request, avoiding the caching issues related with DNS-based CDNs. This anycast IP address corresponds to multiple CDN cache servers, such that user traffic gets directed to their closest anycasted CDN server via standard Internet routing. It is at this point that our MIMA architecture differs from the standard anycast CDN implementation, as we also utilize MIMA nodes throughout the network that act on user traffic passing by. We rely on co-operating routing elements to direct anycast traffic towards our MIMA nodes, where they perform a translation operation that converts the anycast CDN destination address into a unicast one, thus performing Man-In-the-Middle Anycast. As such, CDN cache nodes in our MIMA network need to be reachable by both a unicast *and* an anycast IP address. Each MIMA node is capable of receiving and utilizing information about CDN cache utilization and status, allowing it to perform user-server assignment at a fine level of granularity, despite our anycast structure.

A. Routing Element

A requirement of our MIMA architecture is that routing elements logically collocated with a MIMA node redirect anycast MIMA traffic towards that node, as shown in Fig. 1. In this case, a routing element is simply a device such as a switch or router that's modified in such a way that it is 'aware' of MIMA anycast traffic, and is capable of redirecting said traffic to a MIMA node. For example, a router or switch in the same Point of Presence (PoP) or rack as our MIMA node. In our prototype, we utilized OpenFlow switches installed with a specific rule that matched on user traffic destined towards the anycast address. In production any similar technique can be used, such as VLAN/MPLS tagging, specific BGP routing with co-operating routers, etc. While the actions performed by a MIMA node can technically be performed by a routing element itself (Such as performing NAT within a router), our abstraction of this functionality into a separate virtualized component allows for easier scalability and deployment.

In order to maintain transparency, response packets from the CDN server also need their source IP address changed back to the anycast address. This can simply be done at the cache server itself. This source NAT operation can also be performed at the routing element; all packets with the source address matching the unicast address of a cache server can be safely rewritten into the anycast IP address. However, unless there is good coverage of MIMA nodes and associated routing elements, there is no guarantee that a response packet from the

CDN cache passes by a routing element (A TCP return path may not be the same as the forward path).

B. MIMA Node

We designed and built our MIMA nodes using the Click modular software router, which is capable of high-performance network functionality on commodity hardware [5][10]. Due to this virtualization, MIMA nodes can easily be incorporated wherever the requisite hardware is available, such as within a PoP or datacentre.

As mentioned previously, traffic is passed to the MIMA node by a nearby routing element. Figure 1 demonstrates what happens after this process. First, each incoming packet has its TCP 4-tuple (IP source/dest. address, TCP source/dest. port) checked against the MIMA node's internal **State Table** to see if that flow has previously been assigned to a CDN server. If it hasn't, then a new mapping is created in the state table, assigning this new flow to a CDN cache according to the entries in the **Surrogate Table**. Using this mapping, the destination IP address is **rewritten** from the anycast address into the unicast IP address of the assigned CDN cache. It is then sent out of the MIMA node and back to its connected routing element, where it joins the Internet to be routed via unicast IP. Future packets belonging to this flow then use this stored mapping.

Since this redirection process is only performed on user request packets, the total load per TCP flow on a MIMA node is small and infrequent when compared to response traffic, as each packet is typically either a TCP packet such as a SYN or ACK, or a small payload such as an HTTP GET. This also means that the routing element directing packets to/from a MIMA node has minimal extra load placed on it. MIMA isn't limited to performing NAT operations on packets; another possible mechanism for performing an anycast to unicast translation is to use tunnelling, where the anycast packet is encapsulated within another packet. Example techniques include IP in IP or GRE tunnelling. Using this method, MIMA nodes perform user-server assignment by encapsulating the user anycast packet with another set of headers directing it towards the desired cache server. At the cache server, this packet is decapsulated and processed. The server can then respond using its anycast source address without having to manually rewrite each packet. While both methods of ensuring transparency (tunnelling and source address rewriting) are well supported by the Linux kernel, we implemented the rewriting method due to its simplicity.

New TCP flows are assigned to a cache server according to entries in the MIMA node's **Surrogate Table**. This table acts as a simple round robin mechanism; new flows are assigned to the unicast server addresses held in the table in a weighted round-robin fashion. By altering the ratio and number of entries in this table, we are capable of performing user-server assignment across multiple servers in any ratio we desire. This table is populated by the **Surrogate Information** module, which receives Load information via an API. This information can be provided by a central controller, the cache servers themselves, or some other entity. In our prototype we implemented two types of information: server load and

redirection acceptance. The **Surrogate Information** module combines this information with network locality knowledge to fill the **Surrogate Table**. For example, a server with 'low load and accepting redirection' could have a larger proportion of entries in the **Surrogate Table** than one with 'high load and accepting redirection', and a server with 'low load and not accepting redirection' would have no entries at all. We do not specify an algorithm with which to calculate optimal round robin weightings from the server load information, as that is highly context dependent.

Compared to current anycast-based CDN implementations, MIMA provides a larger flexibility in performing user-server assignment, as it can utilize a larger range of information. By combining this with the anycast to unicast translation process, we are also able to avoid the TCP flapping problem associated with anycast-based CDNS such as CloudFlare and FastRoute. By utilizing an anycast IP address, we also avoid the problems associated with DNS caching.

C. Adaptive User-server Offloading

Under normal operating conditions, our architecture operates similarly to other anycasted CDNs, with clients being routed to the closest CDN cache node. For example in Fig. 2, clients from ISP A would be routed via their local MIMA node at the ISP network, and then towards the CDN cache at PoP2. Clients from ISP B would be routed via anycast to the CDN cache at PoP4, where they would be routed through the MIMA node and onto the CDN cache.

This is not always optimal behaviour, however. For example if the cache at PoP2 is under heavy load, it is desirable to offload new flows from ISP A to the cache located at PoP3 instead, despite it being further away. Other current CDN implementations are either incapable of performing such manoeuvres, or require clients to first perform additional DNS requests in order to refresh their DNS cache. In contrast, MIMA requires no user interaction in order for this form of offloading to occur. Because only new flows are offloaded, users currently connected to the cache at PoP2 are not affected, thus avoiding TCP flapping.

In order to avoid situations where MIMA nodes offload traffic to other highly loaded CDN caches, our MIMA nodes also utilize some form of load information, such as available bandwidth or CPU utilization. A cache server itself can also elect *not* to receive any offloaded traffic, by sending a 'no_redirect' flag. This can be useful if a server is under extreme load, or if it's about to be deactivated due to maintenance. In the scenario that all CDN cache servers are under high load, a MIMA node performs no user-server assignment for new flows, instead just returning the packets to the routing element and allowing them to be routed via native anycast. While these flows may suffer poor QoS, it's a better option than terminating the flow completely. This 'default' behaviour can be changed, depending on the operation desired by a CDN operator.

D. Enabling Dynamic Capacity Provisioning

MIMA's capability in allowing redirection at a per-flow level allows a CDN operator to actively enable or disable cache

nodes as CDN usage changes over time. For example in Fig. 2, if there is extremely high load at the cache at PoP2, instead of offloading traffic from ISP A towards the cache at PoP3, a CDN provider could dynamically ‘spin up’ a cache locally at ISP A, and begin offloading traffic to this new cache via the MIMA node at PoP1. Because these servers are typically temporary (If a CDN provider has to constantly spin up extra servers, then this indicates a planning problem), we provide them only with a unicast IP address so as to avoid TCP flapping from clients that may or may not be covered by our MIMA architecture. Note that this does not involve any form of DNS resolution; under our current paradigm, these new servers are *only* accessible via the MIMA architecture. While this could be changed in future iterations of this work, that option was not explored in our evaluations.

This dynamic capability is currently not possible for anycast based CDNs, due to the statefulness of TCP. For example, if additional anycasted CDN servers were introduced onto the Internet in response to rising server load, users near those servers would automatically be routed towards them via BGP routing, thus causing TCP flapping in the process. Similarly, while this capability is possible for CDNs utilizing DNS techniques such as Akamai or FastRoute, reliance on user DNS behaviour and the presence of DNS caching may cause delays and/or coverage issues, during which user QoS may have suffered dramatically.

E. MIMA Placement and Deployment

Figure 2 demonstrates some example MIMA placements. There are several trade-offs to consider with MIMA node placement: one that is close to the user (such as at an ISP’s network edge) will require more nodes for a complete coverage, but each node will handle less traffic. Whereas a placement on the CDN’s network edge will require less nodes, with each node handling a larger load. This consideration also applies to co-operating routing elements: Routers and switches within an ISP’s network already process a large amount of other traffic, and so including MIMA capability with them may be more intrusive when compared to routing elements at the CDN edge, which only handle CDN traffic. In our evaluation, we used MIMA nodes at the CDN edge. Because we effectively reduce the length of an anycast route by performing an anycast to unicast translation between clients and servers, we similarly reduce the chance that any Internet routing changes will affect our users due to TCP route flapping. For example, if a route change happens past the point where the anycast-unicast translation occurs, then the user is unaffected by that routing change, compared to regular TCP anycast. This is an improvement over other anycast based CDNs such as FastRoute and CloudFlare, as any Internet routing change has the potential to cause TCP flapping in anycast-based routing.

Regardless of placement, IP anycast behaviour ensures that any clients not covered by a MIMA node are simply directed towards their closest CDN cache as usual. Similarly, this also provides MIMA with inherent resiliency: Should a MIMA node fail, traffic simply takes the anycast route. This, combined with the virtualized nature of our MIMA nodes, allows for an incremental deployment that is capable of serving clients

regardless of their origin. Compared to other advanced CDN networks in production such as FastRoute, this presents a significantly lower investment of starting capital.

IV. EVALUATION

The fundamental limitation of anycast-based CDNs is the lack of control on user-server assignment. Therefore, our evaluations aim to demonstrate the capabilities of our MIMA architecture in terms of flexible user-server assignment, which the current anycast-based CDNs are lacking, by redirecting user traffic to more suitable cache servers.



Fig. 3. Example MIMA placement: ISP network edge vs CDN network edge.

A. Evaluation Setup

We based our evaluation topology off REANNZ’s ISP network, as shown in Fig. 3. Each dot on the map indicates the location of a PoP. We performed our evaluations using the Mininet network virtualization tool [8], hosted over Amazon EC2. Each PoP in our test network had a user eyeball network attached. We also placed a MIMA node and CDN cache in each PoP cluster located within a major city - Auckland (AKL), Palmerston North (PMR), Wellington (WLG), Christchurch (CHC), and Dunedin (DUD). We used OpenFlow switches for our routing elements which talk to a Ryu SDN controller [20].

Each user eyeball network consisted of a number of clients, each running traffic generation scripts. Traffic was generated

by having clients in the eyeball networks periodically spawn a random number of TCP connections to the anycast address for a random period of time, emulating user web browsing and content access. By repeating this process across our eyeball network, we obtained a traffic pattern capable of bursty behaviour. We parameterized total load as the number of client running the traffic script at any one time. Our CDN cache servers were implemented as simple web servers over node.js that would accept connections and leave them open until they were closed by the client. These servers provided an indication of load for use by MIMA nodes, as well as a ‘no_redirect’ flag. This was done by logging the number of currently open connections. We used this load metric for simplicity during our evaluation process. By default, each MIMA node assigns incoming connections to the CDN cache node collocated within the same PoP cluster. Connections directed to the MIMA node at the Wellington PoP cluster would be assigned to the cache node at the Wellington PoP cluster, for example.

The **Surrogate Information** module of our MIMA nodes employed a hysteresis-based algorithm when deciding how to perform user-server assignment. If a cache server was under low load (<400 open connections), the collocated MIMA node would continue assigning users to that server. At high load (>500 open connections), the collocated MIMA node begins offloading a portion of user traffic to the next nearest cache node. This offloading continues until the collocated cache node next reports low load, at which point the MIMA node stops offloading. Load and redirection status information was collected by a centralized logger, which then pushed that information to each MIMA node. Changing the hysteresis limits allowed us to achieve a variety of system responses, as we will demonstrate. In production, this hysteresis-based algorithm could easily be replaced; one possible example could involve a PID controller, for ex- ample.

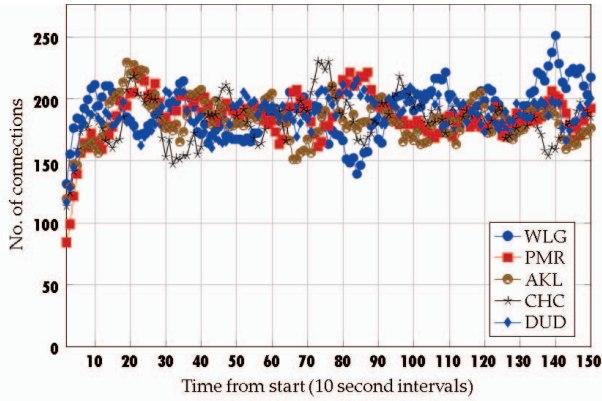


Fig. 4. Number of connections per server over time.

B. Benchmark Performance

First, we demonstrate that our architecture is capable of providing basic anycast performance, and that our traffic generation model is fit to use in our evaluations. This is achieved simply by running the same number of traffic generation scripts across each eyeball network; this should

cause the average load across the CDN caches in each PoP to be roughly equal. The results of this is shown in Fig. 4. As expected, our architecture is capable of performing user-server assignment in such a way that load is geographically load balanced equally, as expected from an anycast-based scheme (each PoP and its surrounding topology was designed such that an equal number of users would be directed towards them in normal load situations). This benchmark evaluation also demonstrates the nature of the test traffic generated by our eyeball networks under normal load - Spiky, but with an average value around 150 to 250 connections per server.

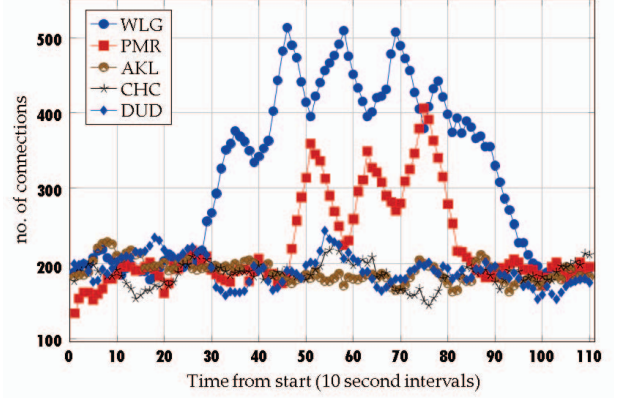


Fig. 5. Number of connections per server over time.

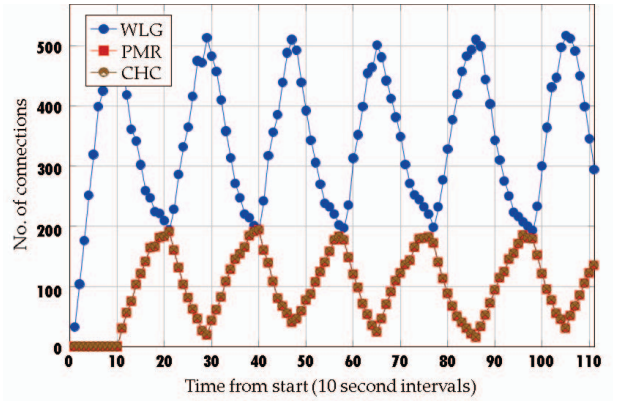


Fig. 6. Wellington undergoes a flash flood with no other background traffic, and offloads to the nearest two CDN caches with a large hysteresis range.

C. Flexible Anycast Offloading

In this section we demonstrate that our MIMA architecture is capable of providing native anycast performance under normal operating conditions, whilst providing load-aware offloading capabilities during times of high load. We start this scenario by running traffic generation scripts across our eyeball networks, and waiting for them to reach an equilibrium (roughly 200 connections per server). This can be seen between T0 and T25 in Fig. 5. Because an equal amount is

traffic being generated across our test network, each server is under roughly equal load during this period.

In the time period between T25 and T80, we tripled the traffic originating from clients around the Wellington PoP, simulating a flash crowd event. As a result, the load at WLG can be observed to rise until roughly T45, at which point the load increases above our ‘high’ limit of hysteresis. As a result of this load information, the colocated MIMA node at the WLG PoP begins offloading a proportion of new user flows to the next closest CDN cache, located at Palmerston North. This causes the rise in load at the PMR cache, and a corresponding decrease at the WLG cache as some traffic is offloaded to the PMR cache. When the traffic at the WLG cache drops below the ‘low’ limit of hysteresis, the MIMA node stops offloading traffic, and starts assigning user flows to the WLG node again. This cycle repeats twice until T80, at which we return the traffic generation at WLG to normal levels. From T95 onwards, equilibrium is reached again.

This scenario confirms that our MIMA architecture is capable of performing flexible user-server assignment in times of low and high load. At low load, users are assigned to their closest cache servers, as expected in an anycast-based CDN. If a cache suddenly is put under high load however, MIMA is also capable of offloading user traffic by assigning new traffic flows to other nearby caches. Despite our simplistic implementation of a user-offload algorithm that uses basic hysteresis parameters, we demonstrate that it is easily capable of maintaining the load at a particular cache within a certain threshold range. For reference and to demonstrate its versatility, Fig. 6 shows a similar scenario to Fig. 5, but with a wider hysteresis range (200 – 500), and with Wellington offloading equally between the Christchurch and Palmerston North caches in response to a flash crowd event.

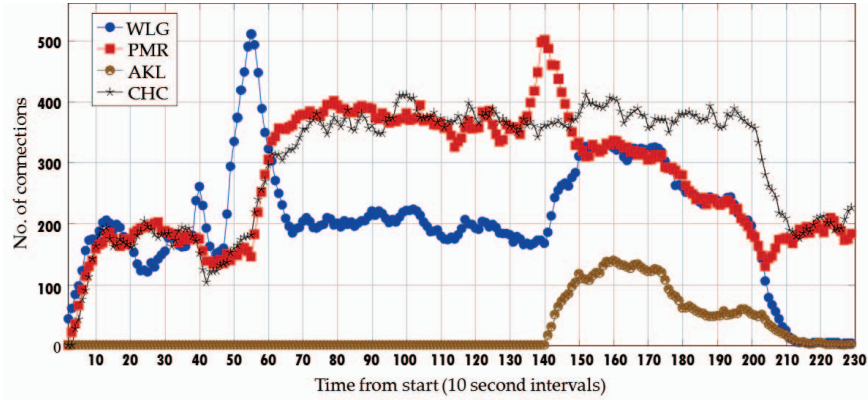


Fig. 7. Number of connections per server over time.

Figure 7 further demonstrates how our architecture is able to handle non-trivial load scenarios with minimal input/configuration. Similar to Fig. 6, the hysteresis threshold has been configured to a large value (150-500), and offloading behaviour has been set such that a cache offloads to its two neighbouring caches in times of high load.

- During T0-T45, load across the WLG, PMR, and CHC caches is fairly even, at around 200 connections each.
- At T45, we initiate a flash crowd event around the WLG server, causing the number of connections to spike.
- At T55, the number of connections at the WLG server breaches the ‘high’ hysteresis threshold, and the WLG MIMA node starts offloading traffic to its two neighbouring caches (PMR and CHC)
- From T70 to T135, equilibrium is reached, and the flash crowd around WLG has been successfully diverted while keeping load at the WLG, CHC, and PMR servers at manageable levels.
- At T135, we increase the demand around the PMR area, causing load at the PMR cache to also spike. This

causes it to offload to its neighbouring caches (WLG and AKL). As a result, load at the WLG and AKL servers can be seen to rise.

- From T170, we begin lowering the demand around the WLG area. As a result, load at the PMR and CHC caches begins to decrease.
- At T205, we bring back load around the PMR cache to its normal levels. Consequently, CHC and PMR return back to their equilibrium load levels as seen at the start of the scenario. AKL, having no base traffic begin with, returns to 0 load level.

While this scenario successfully demonstrates the flexibility and adaptiveness of our architecture under a multitude of conditions and settings, it also highlights a potential flaw in our design. At T140 to T180, the PMR server has started offloading traffic towards its neighbouring caches, WLG and AKL. But at the same time, WLG is also offloading a portion of *its* traffic towards PMR (and CHC) as well. That means that our architecture is in a situation wherein requests are being handled in parallel - server A handles some of server B’s load, and vice versa. Clearly, it would be better for these servers to

instead handle their own load individually, and avoid the added overhead of offloading to another server when it's not needed. In order to avoid this, our architecture also implements a 'no_redirect' flag, as we describe below. Note that this scenario does not create a loop; because the MIMA nodes perform the anycast to unicast translation before offloading, those packets are not *retranslated* at the other node.

D. Avoiding Parallel User-server Offloading

By incorporating a 'no_redirect' flag into our architecture (Fig. 1), we are able to prevent servers from performing 'parallel offloading', wherein servers offload traffic to other servers that are also currently overloaded and may offload back. For example, both WLG and PMR caches need offload. They may offload to each other. This is not desirable.

To solve this problem, Microsoft's FastRoute architecture [3] uses multiple rings / layers of cache servers. The servers cannot offload to the servers within the same layer. They can only offload from outer layer to inner layer. This requires that the inner layer has larger capacity than the out layer, and assumes that the innermost layer has unlimited capacity. While this approach avoids the 'parallel offloading' problem, it requires large amounts of over provisioning.

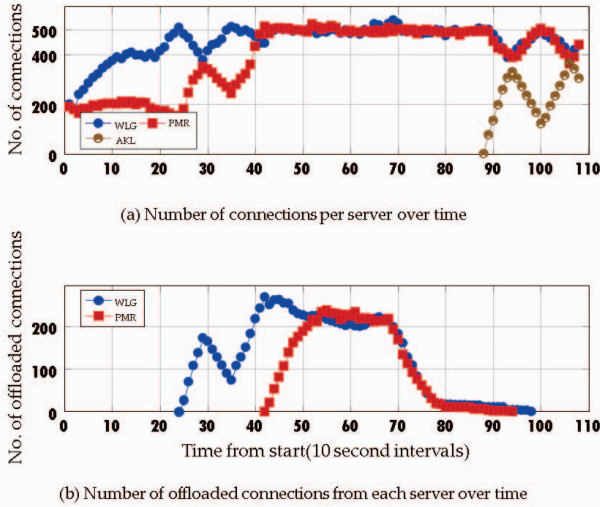


Fig. 8. Number of connections over time under extreme load. We dynamically provision a CDN cache to accommodate this load.

In order to simplify the presentation of our results in this evaluation, we only generated traffic around the WLG and PMR PoPs. Figure 8a shows CDN cache load in this scenario over time. In this scenario, both PMR and WLG start under equilibrium, at around 200 connections each. The hysteresis thresholds have been set to 400-500. We describe the course of the related evaluation below:

- At T0, we begin increasing the demand around the WLG PoP, causing the load at the WLG cache to correspondingly increase.
- At T25, the load at the WLG cache goes over 500 connections, the high hysteresis threshold. The WLG MIMA node begins offloading user traffic towards the

PMR cache. As a result, load at the WLG node drops, and load at the PMR node increases. This is demonstrated in Fig. 8b, which shows the number of connections the WLG and PMR MIMA nodes are offloading towards each other.

- At T30, the load at the WLG cache goes under the threshold for low load, and the WLG MIMA node stops offloading traffic to PMR. This is correlated in Fig. 8b. Around this time, we also increase the traffic load originating from around the PMR PoP, leading to the increase in PMR's load shown in Fig. 8a during T35-T42.
- At T42, the PMR cache has also reached high load, and so it's collocated MIMA node begins offloading its traffic towards the WLG cache node. As expected, the WLG and PMR MIMA nodes begin offloading to each other, as shown in Fig. 8b wherein each server has a high number of offloaded connections. This lasts until T70.
- At T70, we enable our 'no_redirect' flag, informing the PMR and WLG MIMA nodes that they should not redirect to each other's cache servers. The slow decrease in offloaded connections shown in Fig. 8b indicates that currently offloaded connections are not dropped immediately; only *new* connections are handled differently as per the 'no_redirect' flag.

As we have demonstrated, our architecture is flexible enough to allow for the incorporation of a 'no_redirect' flag in order to overcome the difficulties associated with offloading and user-server assignment in times of extreme load. In this scenario, both servers were loaded to the point that there were no suitable offloading candidates; our architecture is capable of handling such scenarios and thus avoids such a 'parallel offloading' situation.

E. Dynamic Cache Provisioning

While implementing this 'no_redirect' flag has solved the problem of parallel offloading, users are still being assigned to heavily overloaded servers. One way to address this problem is via cache server virtualization. As such, at T87 in Fig. 8a new CDN cache server is initialized at the AKL PoP, and we populate its information into the WLG and PMR MIMA nodes, which begin offloading traffic towards this new server, as shown in Fig. 8a. From there, behaviour is similar to our previous scenario, with load at the PMR and WLG cache nodes being kept between the hysteresis thresholds by offloading traffic to AKL. MIMA's performance in this scenario indicates that MIMA is not only capable of load-aware user-server assignment and offloading in times of high cache load, but also capable of reverting back to native anycast performance in order to avoid a parallel offloading situation. In addition, we show MIMA to be capable of utilizing dynamic resource provisioning.

V. CONCLUSION

This paper presents MIMA, a Man-In-the-Middle Anycast (MIMA) CDN architecture that enables flexible user-server

assignment in a variety of network conditions. By returning an anycast IP address for all DNS requests, our architecture avoids the client-side behaviour problems associated with DNS-based user-server assignment. And due to the fact that our MIMA nodes transparently convert an anycast address into a unicast address (performing user-server assignment in the process), we are able to actively redirect and offload users towards different cache servers without needing to worry about disrupting connections in progress. We have built MIMA to be virtualized by design, allowing for simple, rapid implementation across the Internet. These novel capabilities, combined with a growing ability to rapidly provision and instantiate CDN cache nodes, paves the way for future CDN operators in designing and running a flexible, load- responsive network. Similarly, we believe that our work can be used as an enabler for ISP-CDN collaborations, which are becoming increasingly necessary in a media-focused world. Our preliminary work here has barely scratched the surface of what is possible; note that the flexible user-offloading capabilities of our MIMA architecture were achieved with a simple hysteresis-based offloading algorithm. MIMA has been designed from the ground up such that each component can be customized as required, from the signalling used to indicate whether or not a cache is suitable for offloading, to the algorithms used to decide when to offload.

ACKNOWLEDGMENT

This project was funded in part by InternetNZ. Thanks to REANNZ for providing an image of their network map, and their support at the start of this project. We thank Cisco Fellow Fred Baker for his insight on industry research in this field.

REFERENCES

- [1] Cisco Visual Networking Index: Forecast and Methodology, 2014-2019 White Paper.
- [2] D. L. Black, J. Hudson, L. Kreeger, M. Lasserre, and D. T. Narten. An Architecture for Overlay Networks (NVO3). Internet-Draft draft-ietf-nvo3-arch-04, Internet Engineering Task Force, Nov. 2015. Work in Progress.
- [3] A. Flavel, P. Mani, D. A. Maltz, N. Holt, J. Liu, Y. Chen, and O. Surmachev. Fastroute: A scalable load-aware anycast routing architecture for modern cdns. In Proceedings of NSDI'15.
- [4] B. Frank, I. Poese, Y. Lin, G. Smaragdakis, A. Feldmann, B. Maggs, J. Rake, S. Uhlig, and R. Weber. Pushing cdn-isp collaboration to the limit. SIGCOMM CCR, 43(3):34-44, 2013.
- [5] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. ACM Trans. Comput. Syst., 18(3):263-297, Aug. 2000.
- [6] S. Kuenzer, J. Martins, M. Ahmed, and F. Huici. Towards minimalistic, virtualized content caches with minicache. In Proceedings of HotMiddleBox '13.
- [7] J. Lai, Q. Fu, and T. Moors. Rapid ip rerouting with sdn and nvf. In Proceedings of Globecom'15, 2015.
- [8] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: Rapid prototyping for software-defined networks. In Proceedings of Hotnets-IX, 2010.
- [9] T. Leighton. Improving performance on the Internet. Commun. ACM, 52(2):44-51, Feb. 2009.
- [10] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici. Clickos and the art of network function virtualization. In Proceedings of NSDI'14.
- [11] E. Nygren, R. K. Sitaraman, and J. Sun. The akamai network: A platform for high-performance Internet applications. SIGOPS OSR, 44(3):2-19, Aug. 2010.
- [12] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. Making middleboxes someone else's problem: Network processing as a cloud service. In Proceedings of ACM SIGCOMM 2012.
- [13] Q. Xie, R. R. Stewart, S. Maruyama, M. Tuexen, and M. Kozuka. Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. IETF RFC 5061, Oct. 2015.
- [14] <https://fansided.com/2016/04/24/hbo-go-crashes-during-game-of-thrones-season-6-premiere/>
- [15] M. Calder, A. Flavel, E. Katz-Bassett, R. Mahajan and J. Padhye. Analyzing the Performance of an Anycast CDN. In Proceedings of ACM IMC 2015.
- [16] H. A. Alzoubi, S. Lee, M. Rabinovich, O. Spatscheck and J. Van der Merwe. Anycast CDNS revisited. In Proceedings of ACM WWW 2008.
- [17] H. Ballani and P. Francis. Towards a global IP anycast service. In Proceedings of ACM SIGCOMM 2015.
- [18] C. Perkins, Ed. IP Mobility Support for IPv4, Revised. IETF RFC 5944, November 2010.
- [19] S. Salsano, N. Blefari-Melazzi, A. Detti, G. Morabito and L. Veltri. Information Centric Networking over SDN and OpenFlow: architectural aspects and experiments on the OFELIA testbed, Comput. Netw. 57 (2013) 3207-3221.
- [20] <https://osrg.github.io/ryu/>