

# A Content-Based Traffic Engineering Policy for Information-Centric Networks

Serhat Nazim Avci, Cedric Westphal  
Huawei Innovation Center  
{serhat.avci, cedric.westphal}@huawei.com

**Abstract**—Information-Centric Networks offer an opportunity to re-think traffic engineering, by allowing to schedule resource at the granularity of content. We describe an architecture for performing content-based path selection and resource allocation. It assigns content objects to paths in the network by considering the time it will take to deliver the existing objects in the network and optimizes with respect to this objective. We have extensively simulated this mechanism to demonstrate significant gain up to 72% in response time when compared with minimum backlog policy (MBP), round-robin format (RRF), and the traffic engineering policy which minimizes the maximum link utilization (minMLU).

## I. INTRODUCTION

New resource allocation mechanisms are required to achieve high link utilization and to improve network performance [14][6]. The current Internet architecture provides a flow abstraction for traffic engineering that is unfortunately ill suited for high link utilization. This is due to several reasons: first, an IP flow matching a filter with a certain source and destination addresses may encompass many different applications, especially in the case of NAT'ed clients. Second, an IP flow is a poor descriptor of the amount of resource to allocate, as it is difficult to predict the flow length just by looking at the network layer information in the packet header. Third, there is no explicit semantics to signal the end of an IP flow, and it is typically identified by waiting for a timer to elapse after the last packet of the flow. This leads to resource being allocated to flows after they are completed.

Typical mechanisms to perform traffic engineering then rely on historical and probabilistic methods, by considering the traffic matrix observed during a previous period, and computing some weights to select paths to efficiently load balance the traffic over the network; this is probabilistic in the sense that flows are assigned to paths with the hope that in the long run, the behavior will converge to the average value obtained from the traffic matrix.

The flow length distribution in the Internet (and many other networks) however follows a power law distribution [13]. This means that a few misallocated flows may skew the performance of the network, as seen for instance in the case of the data center network [10]. As the traffic demands increase, making more efficient resource allocation is primordial, but the current IP architecture comes up short in providing the proper abstractions to achieve this.

New architectures have been proposed to alleviate some of the issues of IP. In particular, Information-centric networks

(ICN, [7][5][15]) have been proposed to use content as the network abstraction. Content distribution, and video in particular, is predicted by Cisco to amount to 90% of the Internet traffic by 2017. ICNs use a unique content name to identify the data crossing the network, and some cryptographic mechanisms to securely bind the name to the data.

This enables two important features: one is that the content can be stored anywhere in the network, as it is the content that is authenticated, not the server providing it; and the second is that the network can identify properties of the content, such as its size, and associate these to the content name [1]. This is a profound shift, as allocating resource to a piece of content with known attributes becomes deterministic.

We argue that ICNs offer the proper abstractions to perform a finer grained resource allocation than IP, and in this paper, we investigate how to leverage these abstractions to perform content-based traffic engineering<sup>1</sup>. We design a content-allocation architecture in ICN which provides the proper functions for content-based resource allocation, and design some scheduling policies which take advantage of these functions to achieve a significant gain over current IP resource allocation procedures.

The contribution of this paper are as follows:

- We define a content-based traffic engineering problem, inspired by the work of [1][25] and we show that this problem is NP-Hard;
- We therefore develop a heuristic approach and a resource allocation policy which reduces the response time of the network for delivering a piece of content;
- We observe that the control loop for resource allocation requires knowing the actual rate of data transfer for a flow after it has been allocated, which in turns require for the content allocation mechanism to be able to either *estimate* the performance of the transport layer, or to *learn* it;
- We define a resource allocation mechanism which learns the desired allocation over a training period, then exploits this policy for a while, before training again and repeating the cycle;
- We thoroughly evaluate the proposed mechanisms both in a Java simulator built for this purpose, and using NS-2

<sup>1</sup>Content-based TE here means that we assign resource to each piece of content. We respect the network neutrality principle and do not prioritize any one content over others. Even though some provider may want to give better performance to content from, say, Disney, we only consider a resource allocation scheme that is blind to the type and producer of content.

simulations; we show significant gains in response time for a wide range of network conditions in both WAN-like and Data Center-like topologies.

We believe that fine-grained resource allocation sets ICN apart from the current IP network architecture. Despite being a heuristic, the policies we describe in this paper demonstrate a significant gain over the minMLU algorithm, over the RRF algorithm, and over the results of MBP algorithm in [25]. In RRF, every source-destination pair has a set of candidate paths. These paths are selected in a round-robin format every time a new flow comes in to the network between these two nodes. However, these are heuristic policies and they only point in the direction towards even more gain for the optimal policy.

The paper is organized as follows. We first present some background and related works in Section II. We next define the problem statement and the requirements for a content-based traffic engineering mechanism in Section III and prove there our problem formulation is NP-Hard. In Section IV, we propose our network architecture. Section V discusses the evaluation environment and presents the simulation results. We discuss these results in Section VI, prior to concluding the paper in Section VII.

## II. RELATED WORK

Information-centric networks [19][7][5][15] have been proposed to securely name data and route based upon the data name. A lot of work in ICN has been on separating the content from the server which hosts it, in order to facilitate content replication and eventually, content delivery.

However, it has quickly become clear that ubiquitous content caching came up with resource allocation implication, and TECC [27] for instance considered how to jointly provision the content placement in the cache with the path selection for the delivery. TECC solved an optimization problem taking as input the traffic matrix implied by the users' demands, and proposed an off-line content placement and traffic engineering mechanism, while our focus is on dynamic, fine-grained resource allocation.

Our work build on that of [1] and [25] which respectively observed the potential of ICN for traffic engineering and proposed an initial proof of concept and simple resource allocation policy. However, [25] offered only a basic heuristic and the results of this policy vary greatly upon the network conditions and traffic demands. We propose here a scheduling policy which performs consistently well over a wide range of input.

[22] considered the issue of traffic engineering in ICN, but by splitting the flows into multiple paths and without taking into account the object size. We on the other hand consider a single path for an object, but utilize the size awareness offered by the ICN paradigm. In the Content-aware Traffic Engineering (CaTE) paper [21], a different framework is presented where a data can be stored in multiple server locations. This knowledge is leveraged to find not only the optimum path but also the optimum source location. There is a joint source node and path optimization. Therefore, it benefits

both the internet service providers (ISP) and content providers (CP).

Traffic engineering in IP networks has been well studied, but most of the work is content agnostic. For instance, [18] computes OSPF link weights to allocate IP traffic. To include content-based differentiation, CDN heuristics have been used as in [11]. [4] considers history-based strategies to perform joint traffic engineering and content distribution.

Integrating the CDN with the network has been the main approach to optimize the content-based path selection. [21] shows that the proper CDN server selection coordinated with the recommendation of the network operator improves performance. Our set-up differs as there is no coordination of the content location selection with the network operator. Using game theory, [17] and [3] consider joint traffic engineering and content selection.

QoS routing is a mature topic to find optimal TE policy on certain constraints. The main approach is to find the optimal paths typically in terms of cost, loss probability, available link bandwidth, link propagation delay, delay jitter, hop count [29]. Our goal is to minimize total flow completion time. In addition, QoS routing does not have the fine grained flow abstractions available in ICN.

The shortcomings of minimizing the maximal link utilization of the network (the so-called minMLU traffic engineering policy) were pointed out in [24]. We believe our paper answers their call for better TE tools in terms of application performance. [8] also attempted to replace minMLU to account for the unpredictability of the traffic. Here, predictability is improved by working with a content abstraction which can precisely define the object to allocate in the network.

Our policies are heuristic, and improve the performance. We hope our paper will instigate some theoretical work on optimal scheduling policies. In networks of parallel servers, the Join-Shortest-Queue policy has been studied by, for instance, [9] or [16]. This analytical framework needs to be extended to the case of networks of queues, and a theory of resource allocation in ICN is clearly required.

## III. PROBLEM STATEMENT AND REQUIREMENTS

### A. Network assumptions

Fine grained resource allocation creates a host of requirements on the underlying networks and on the mechanisms used to provide the allocation. Current networks do not support such mechanisms, for lack of the proper tools and abstractions.

We consider a network where a flow is initiated by a request for content from a consumer of the content. The consumer may be inside the network, or in a different domain. The network receives the request, and the content will flow back through the network. We make a somewhat restrictive assumption that a content object will be mapped to a single flow and that all chunks composing one content object will be allocated to the same path through the network. ICNs support delivery of the chunks from multiple sources, but for this initial proof of concept, we make the simplifying assumption that all chunks follow the same path.

We do not assume a strict path symmetry, as in [19], but we note that if we were to enforce such symmetry, the traffic allocation decision can be made on the request. Namely, if a customer sends a CCN/NDN interest for packet, we can perform the path selection by routing this interest to the location hosting the data over the desired return path *for the content*.

For ease of explanation however, we assume that a content flow<sup>2</sup> enters the network, either from a host directly attached to the network or from a different domain, and that the network will make a content-based routing decision to deliver the content to the requester. The decision is made either by a control layer as in [1] (for instance, using SDN tools adapted for content-based routing) or by the strategy layer as in [19].

We assume that a content is uniquely associated with its identifier (in ICN, its name) and that the content size can therefore be associated with this name as well. Many mechanisms can be used to create a mapping of the content name to its size, and we do not presume any specific one.

More formally, we consider a graph  $G = (V, E)$  with  $V$  the set of nodes and  $E$  the set of links. Each link  $e$  has capacity  $c_e$ . Each content (or flow)  $z_{s,d}$  entering the network at vertex  $s$  and leaving at vertex  $d$  can select a path from a set of  $K_{s,d}$  distinct paths ( $P_{s,d}^k, k = 1, \dots, K_{s,d}$ ), where a path is an acyclic sequence of links in  $E$  going from  $s$  to  $d$ . If link  $e$  belongs to path  $P_{s,d}^k$  for some  $s, d, k$ , we say that  $e \in P_{s,d}^k$ .  $K_{s,d}$  is assumed to be relatively low to simplify the allocation decision and the management complexity. In our evaluations,  $K_{s,d}$  ranges from 3 to 5. In other words, we assume content placement is given as input and focus on TE.

According to classical networking models, flows from  $s$  to  $d$  are generated according to a Poisson process with rate  $\lambda_{s,d}$ . Since each flow corresponds to a piece of content, and since the network has access to the content size, we can assume that, upon arrival of a flow  $f$  in the network, the network has access to the flow size (which we also denote by  $z$ ). We assume that the size  $z$  is drawn from a known distribution with mean  $\bar{z}$ . In our evaluations, we consider the content size to be Pareto distributed.

We further assume that the amount of traffic under the arrival rate  $\lambda_{s,d}$  and the distribution for  $z$  is stable and can be allocated to the paths  $P_{s,d}^k$  in a manner such that the load allocated to each link is less (on average) than this link's capacity. Namely, we assume that there exist coefficients  $\pi_{s,d}^k, 1, \dots, K_{s,d}$  with  $0 \leq \pi_{s,d}^k \leq 1$  and  $\sum_k \pi_{s,d}^k = 1$ , such that the link utilization  $u_e$  of link  $e$  satisfies the following *feasibility condition*:

$$\forall e \in E, u_e = \sum_{P_{s,d}^k: e \in P_{s,d}^k} \lambda_{s,d} \pi_{s,d}^k \bar{z} < c_e \quad (1)$$

Note that the matrix  $\{\bar{z} \cdot \lambda_{s,d}\}_{(s,d) \in V \times V}$  corresponds to the traffic matrix in the network, and the  $\pi_{s,d}^k$  corresponds to a static traffic engineering decision. For instance, a possible

<sup>2</sup>We use the term "flow" interchangeably with content, defined as the sequence of all chunks belonging to a single data object.

traffic engineering policy could be to randomly split the flows arriving from  $s$  to  $d$  with probability  $\pi_{s,d}^k$  onto the  $K_{s,d}$  possible paths  $P_{s,d}^k$ . We denote by minMLU the random splitting policy where the choice of coefficient  $\pi_{s,d}^k$  minimizes  $\max_{e \in E} u_e$ . This is the typical *min-MLU* traffic engineering policy which minimizes the Maximum Link Utilization.

One important aspect is that we only modify the path of the objects through the network, but not the amount of traffic that is offered to the network. Therefore, if there is a solution to Equation 1, the network will be stable (i.e. able to deliver all the traffic) and the link utilization of all policies which keep the network stable will be the same. Our goal is not to improve link utilization, but to reduce the delay to deliver a flow (or equivalently, by Little's Theorem, the number of flows in progress at any given time).

We denote by  $z_i$  the  $i$ -th content to arrive into the network (and slightly abusing notations, its size as well).  $z_i$  is associated with a source  $s \in S$  and a destination  $d \in D$  and we assume that for each source-destination pair, there is a set of paths  $\mathcal{P}_{s,d}$  that  $z_i$  can follow. We denote by  $V$  that allocation of the sequence of  $z_i$ 's to a corresponding path, and define  $T_V(z_i)$  to be the completion time of  $z_i$  under this allocation.

Another key aspect is that of the number of flows being considered at a given time. For an extremely large amount of flows, the probabilistic splitting of the flows according to the  $\pi_{s,d}^k$  will yield a result which converges to Equation 1 by the central limit theorem. This means that the link utilization in such case will be close to optimal. Further, for a very large amount of flows, the resource allocation needs to be kept simple to keep up with the speed.

However, for smaller scales and with heavy tail flow size distribution, the probabilistic resource allocation will have worst results (as we will see in the Evaluation). Therefore, we restrict ourselves to networks at the edge, and ignore the core of the network (where minMLU will perform fine).

We define the *response time* of the network for flow  $z$  as the time between the first arrival of the first chunk of  $z$  at the ingress of the network until the last departure from the last chunk of flow  $z$  at the egress of the network.

The content-based traffic allocation problem is therefore to find an allocation  $V : z_i \rightarrow \mathcal{P}_{s,d}$  which solves:

$$\text{Content-Based TE Problem: } \min_V \sum_{i=1}^n T_V(z_i) \quad (2)$$

*Theorem 3.1:* The Content-Based TE Problem is NP-Hard **Proof:** Our problem is a dynamic flow scheduling problem. A special case of our problem formulation is where each item is of the same unit size and contains only one packet to go through the network. This can be mapped in a straightforward manner to the aircraft routing problem defined in [23], and shown there to be NP-Hard. Since it is a sub case of our formulation, our problem is therefore NP-Hard as well ■

The complexity of finding an optimal solution to the Content-Based TE problem therefore induces us to look at heuristic solutions. We now turn to describing the requirements of a practical solution and to presenting a content-based traffic allocation mechanism which meets these requirements.

## B. TE Requirements

Current networks fall short of enabling this vision. IP flows are difficult to properly define, since a sequence of packets matching an IP header filter with no interruption longer than a given time-out can incorporate multiple distinct content, applications, or even users. This yields our first requirement.

**Requirement 1: Content-based abstraction.** In order to perform a fine grained resource allocation, the network layer needs to be able to uniquely identifies each flow and to be able to distinguish different content and users.

The current IP architecture has been evolving towards a software-based control plane [12] which performs per-flow decisions. As a new flow enters the network, a rule is applied that is set by the controller. As we want to achieve a content specific traffic engineering, we need such mechanisms to apply to content, as in for instance [1].

**Requirement 2: Content-based Control Plane.** The control plane of the network extends to the edge, and is able to make routing decision once for each (or a subset of the) piece of content.

The decision can be as simple as assigning a tag (such as an MPLS label) at the ingress edge, so that the flow follows a given path through the network fabric until it reaches the egress edge.

The control plane also needs to be aware of the congestion in the network fabric to make a proper path selection. For this, a network monitoring infrastructure is required to keep track of the amount of traffic allocated to the nodes' outgoing links. Namely, when a flow  $z_{s,d}$  with size  $z$  is allocated to the  $k$ -th path  $P_{s,d}^k$ , it will add  $z$  to the backlog of the edges traversed by path  $P_{s,d}^k$  and for the nodes monitoring the congestion, a description of the flows and remaining backlogs is required.

While this is a strong requirement, we make it to demonstrate the potential of the mechanism to reduce the response time of the network. We hope that this requirement can be relaxed. [25] considered some policies which ignored all flows below a certain size threshold for resource allocation, and [2] focused on identifying only elephant flows in the context of the data center for instance. From the knowledge of the flows assigned to the paths, the traffic assignment mechanism must be able to derive some behavior of the forwarding plane.

**Requirement 3: Estimation of traffic.** The control plane needs to be aware of the behavior of the forwarding plane under given flow conditions.

TCP is an end-to-end protocol, which makes the forwarding behavior inside the network (say, reaction to congestion) contingent to a policy at the end points. In order to properly assign resource, the control plane would need to understand the impact of its decisions on the flows and have a model of the TCP behavior (either a theoretical model as in [20] and references therein, or an empirical model based upon previously observed network conditions).

**Requirement 4: Scale.** Any resource allocation policy has to scale up with the size of the network.

We have addressed this requirement earlier: for large scales, a probabilistic approach will approach the optimal. Therefore

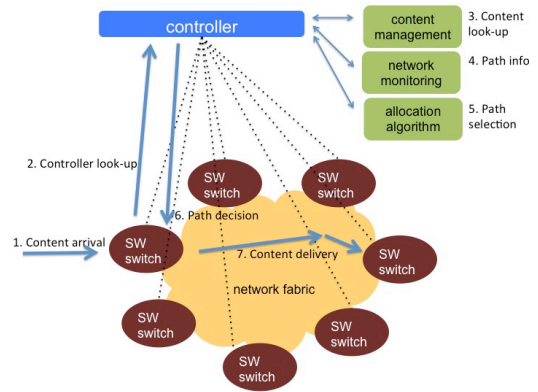


Fig. 1. Architectural View

we suggest two tiers of policy: a probabilistic minMLU mechanism in the core, and a dynamic allocation from the edge to the core, to meet this requirement.

## IV. MINIMAL RESPONSE TIME POLICY

### A. Architectural description

Figure 1 presents the elements of our resource allocation architecture. We assume an ICN protocol names the object, so that a content is uniquely identified by its name, taking care of Requirement 1. We assume a (logically) centralized controller making content-based decisions at the edge of the network, which is typically going to be a software switch connected to the controller.<sup>3</sup>

The controller handles several functions:

- It can select the path for the content, either by setting a rule at the switches in the network fabric, or by assigning a label for a pre-computed path;
- A Content Management function performs the following tasks: it maps the content to a location, either in a cache in the network, or to an egress path out of the network; it also monitors content information, such as identifying the content size and keeping a database of the mapping of the content to its size;
- A Network Monitoring function maintains a view of the conditions inside the network, either by polling the switches for network conditions, or by estimating the conditions from the input and the predicted network evolution, or a combination of both;
- Finally, based upon the input of the Content Management and Network Monitoring functions, an allocation algorithm decides which of the candidate paths for a  $(source, destination)$  pair will provide the best network performance.

<sup>3</sup>Some ICN architectures such as CCN/NDN do not specify a controller, and assume a distributed routing mechanism. However, CCN/NDN includes a strategy layer that selects how to forward interest packets, and if the upstream edge router has some knowledge of the downstream network congestion, this edge router can select the proper path for the data messages, assuming the function of the (now distributed) controller. For simplicity, we describe only a centralized controller here.

The controller could select multiple paths and allocate some amount to each path. However, our goal is to demonstrate the benefit of the idea in the simple scenario, therefore as a first step, single path is our focus. This controller can be extended as an extension of OpenFlow, as in [1]. The mapping of the content name to its size is straightforward in ICNs. The Network Monitoring and Allocation Algorithm functions need to be detailed more, in the following subsections.

### B. Allocation Algorithm

Recall  $z_i$  is the  $i^{\text{th}}$  content to be transferred in the network, as well as its size in bytes. To each point-to-point content transfer from the source  $s \in S$  to the destination  $d \in D$ , we associate a backlog function  $B_{z_i}(t)$  to denote the backlog generated by the content  $z_i$  from  $s$  to  $d$  at time  $t$ . Letting  $t_{z_i}$  be the arrival time of content  $z_i$ , then  $B_{z_i}(t)$  is a non-increasing function of  $t$  for  $t \in [t_{z_i}, +\infty)$ : it diminishes from the size of the content  $z_i$  down to 0. For instance, if we consider a flow using the full capacity  $c$  of a link, then  $B_{z_i}(t)$  can be given as following:

$$B_{z_i}(t) = [z_i - c(t - t_{z_i})]^+. \quad (3)$$

where  $[g]^+ = \max\{g, 0\}$ . In general, due to the dynamics of the flow interactions and of the transport protocol (say, TCP in IP, or an interest-shaping mechanism in CCN/NDN), it is easier to compute  $B_{z_i}(t)$  at each link by subtracting the volume of content that has traversed the link from the original content size.  $B_{z_i}(t)$  correspond to the unfinished, remaining amount for the flow  $z_i$ . Note that this backlog is not inside the network, but rather corresponds to the amount of data which has not transited through the network yet for a specific object.

We are given the bandwidth sharing function which allocates  $f(z_i)$  units of bandwidth to object  $z_i$  per unit of time. For instance, if TCP is the transport protocol, one can view  $f(z_i)$  as the rate achieved by the TCP session which carries  $z_i$ .

Given  $f$  and  $B_{z_i}(t_{z_n})$  for all  $i = 1, \dots, n-1$  (where we consider the  $n^{\text{th}}$  arrival to be scheduled), we can estimate the completion time for all files. It is an iterative process which looks at the next file to terminate, namely the object  $j$  such that  $B_{z_j}(t_{z_n})/f(z_j) \leq B_{z_i}(t_{z_n})/f(z_i)$  for all  $i = 1, \dots, n-1$ . Upon completion of  $z_j$  at time  $t^*$ , we have a different set of objects (all the objects for which  $B_{z_i}(t^*) > 0$  minus  $z_j$ ). We can iterate on all the objects such that  $B_{z_i}(t^*) > 0$  to calculate the completion time of each object. We denote by  $T_V(z_i)$  the completion time of  $z_i$  under the allocation set  $V$  describing the path allocation of the objects  $z_1, \dots, z_{n-1}$ .

For the arrival  $z_n$ , there is a subset  $\mathcal{P}_{s,d}$  of all the paths between source  $s$  and destination  $d$  that we can assign  $z_n$  to. Denote by  $K_{s,d}$  the cardinality of the candidate path subset, and by  $V_{P_i}, i = 1, \dots, K_{s,d}$  the allocation set that describes the current allocation plus the potential allocation of  $z_n$  to the  $i$ -th path  $P_i \in \mathcal{P}_{s,d}$ . For instance,  $V_{P_1}$  is the allocation of  $z_1, \dots, z_{n-1}$  to their current path with backlog  $B_{z_i}(t_{z_n})$  and of  $z_n$  to the first path in  $\mathcal{P}_{s,d}$  with backlog (or in this case, object size)  $z_n$ .

---

### Algorithm 1 Minimum Response Time Policy

---

**Require:**  $\mathcal{P}_{s,d}$  for each  $(s,d)$  traffic demand pair **and**  $B_{z_i}(t)$  for every content  $z_i$  being transferred,  $i = 1, \dots, n-1$  with  $B_{z_i}(t) > 0$

- 1) Select one path  $P \in \mathcal{P}_{s,d}$ ,  $1 \leq i \leq K_{s,d}$  from the candidate paths set and insert it to the allocation set by  $V \rightarrow V + z_n \Rightarrow P$ .
- 2) Given the bandwidth function  $f$  and remaining backlogs  $B_{z_i}(t_o)$  at time  $t_o$ , calculate the expected response time  $T_V(z_i)$  of each flow. Find the checkpoint time  $t_{check}$  which is the minimum expected response time found by  $t_{check} = \min_i T_V(z_i)$ . Update the backlogs of each flow at time  $t_{check}$  by

$$B_{z_i}(t_{check}) = (B_{z_i}(t_{check}) - t_{check} \times f(z_i))^+.$$

- 3) If all flows are completely transmitted then go to next step. Otherwise, recursively go back to step 2 and calculate the response time of the non-terminated flows after the checkpoint. Update the response time of the flows by

$$T_V(z_i) = T_V(z_i) + t_{check}$$

where  $T_V(z_i)$  is the aggregate response time of flow  $z_i$  for candidate allocation  $V$ .

- 4) Calculate the total response time of all flows as  $T_V = \sum_{i=1}^n T_{V_P}(z_i)$ .
  - 5) Iteratively go back so step 1 and select the next candidate path until all candidate paths are selected in series.
  - 6) Given the total response times of each candidate path scenario  $T_V$ 's, select the one which will give the minimum total response time. Add this path to the existing set of paths.
- 

We attempt to find the path  $P \in \mathcal{P}_{s,d}$  such that:

$$\underset{P \in \mathcal{P}_{s,d}}{\text{minimize}} \sum_{i=1}^n T_{V_P}(z_i) \quad (4)$$

that is, to find the path with the minimal total completion time for all objects in the system.

To keep this policy tractable, subsequently in our simulations, we specifically consider  $\mathcal{P}_{s,d} = \{P_{s,d}^k, k = 1, \dots, K_{s,d}\}$  as the set of  $K_{s,d}$  shortest paths given by the output of Yen's k-shortest path algorithm [28] modified to increase path diversity.

Our algorithm, denoted by Minimum Response Time Policy (MRTP), for path selection for incoming content  $z_n$ , originated at node  $s$  and destined for node  $d$ , is summarized in Algorithm 1.

### C. Network Monitoring and Bandwidth Estimation

Algorithm 1 requires to know the backlogs  $B(z_i)$  and the bandwidth sharing function  $f(z_i)$ . The backlog can be monitored at the edge, or can be computed if the function  $f(z_i)$  is known at all time. The function  $f(z_i), \forall i$ , can be estimated

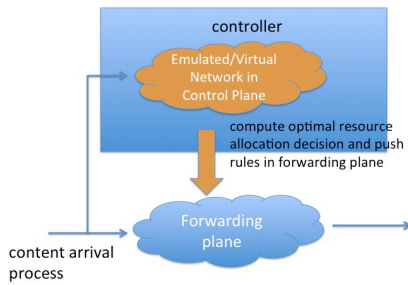


Fig. 2. Computing the MRTP decisions using a Mirror Virtual Network

from a model of the transport layer or from an empirical measurement of the network conditions. In most systems, it is quite complex to estimate  $f(z_i)$  as the dynamics of the system are very intricate. Therefore, the MRTP policy with perfect knowledge of  $f(\cdot)$  is an idealized policy. We present now a scheduling policy which can be realized in practice.

The policy functions in two phases: a *training phase*, where the network acquires the path decision based upon an *off-line learning*; and an *exploitation phase*, where the policy discovered in the training phase is implemented. The two phases alternate periodically, so as to keep the decisions in the exploitation phase up to date.

The learning phase require knowing the best decision. We propose the use of a mirror VNet [26] or a network emulation environment that takes as input the packet arrivals into the real network, and then computes the completion times of the existing flow for each possible path selection to learn the MRTP decision. Therefore, the forwarding happens in the data plane according to the existing policy (say, the decisions learned during the previous training phase) while the resource allocation mechanism acquires the policy for the next exploitation phase.

We propose an iterative optimal path selection algorithm for the training period.

- 1) We start with the first flow  $f_1$  with source  $s$  and destination  $d$  and add it to the active traffic input set  $T$ .
- 2) We select the first candidate path  $p_i \in P_{s,d}$ ,  $i = 1$  of  $f_1$ .  $P_{s,d}$  is the set of candidate paths of flow between  $s$  and  $d$ .
- 3) We run the emulator (in our experiments, NS-2) and calculate the actual total response time of this simulation  $RT_i$ ,  $i = 1$ .
- 4) We select the next  $p_i$  in the set  $P_{s,d}$  and go back to step 2).
- 5) We find the index of the smallest element  $r_1$  of the set  $\{RT_1, RT_2, RT_3\}$  and store it as the optimal path selection of flow  $f_1$ . We add the optimal path selection to set  $PS$ .
- 6) We add the next flow to the active traffic input set  $T$ .
- 7) Except for the last element of  $T$ , we take the path selections of the flows from the set  $PS$ .
- 8) We repeat the steps through 2) to 5) and find the optimal path selection of the last element of set  $T$ .
- 9) Go back to 6) until the size of set  $T$  is equal to the

predefined size of the training set.

Once we find the optimal path selection of the flows in the training set, we use this knowledge to derive the path selection policy for further flows.

- 1) For each source-destination pair, extract the path selection statistics.
- 2) For a specific source-destination pair, if an index is dominantly popular, set that index for the future flows of that source-destination pair.
- 3) For a specific source-destination pair, if more than one index is closely seen in the training set, make a random selection by taking the weights of paths as their popularity in the training set.
- 4) For a specific source-destination pair, if the sample size is 0, use MTRP for future flows of that kind.

During our simulations, we observed that there is a single path index in the training set for most of the source-destination pairs. For the rest, we applied the third and fourth item of the previous list.

Figure 2 shows the traffic being duplicated to be fed both to a network emulator which computes the optimal scheduling decision by exhaustively trying the multiple potential paths and to the actual network for transmission. The emulator is only used during the training periods, and is turned off otherwise. In our evaluations, we compute the optimal weights based upon the input of 1,000 flows, then apply the learned decisions to the next 10,000 flows.

## V. EVALUATIONS

We developed a Java simulator and an NS-2 simulator to evaluate the response time performance of the proposed MRTP against minMLU, MBP, and RRF because we wanted to see the improvement using existing transport protocols. In a CCN-based simulator with a proper interest shaping mechanism, we would observe a higher gain.

### A. Methodology

In the Java simulations, the bandwidth sharing function  $f(\cdot)$  used for estimating the response times is the same bandwidth sharing function used in the simulation itself. Therefore, idealized MRTP is implemented. For each simulation setup, we implemented these algorithms on two different networks and several different traffic models and parameters. The first network is the Abilene (WAN) network, a backbone network established by Internet2 group. It has 11 nodes and 14 edges as depicted on the left of Figure 3. The capacity for each link is 9920Mbps except the link between the nodes in Indianapolis and Atlanta where the capacity is 2480Mbps. In NS-2 simulations, latency of each link is set to 10ms. Three candidate paths for each source-destination node pair are computed based on OSPF link weights given by [18] and Yen's algorithm [28]. We used different traffic demand matrices to characterize the traffic inputs. They are uniform, gravity-based and a realistic model from [30]. The realistic matrix was built by measuring the end to end traffic for each source-destination pair in the Abilene network, for a period of 5 minutes. The

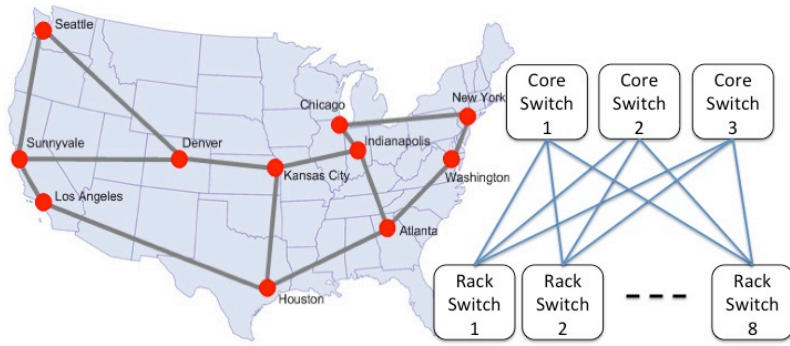


Fig. 3. Abilene Network (left) and Data Center topology (right)

traffic input is created by using Pareto distribution and Poisson process for the object sizes and arrival times, respectively. Pareto parameters  $x_m$  and  $\alpha$  and Poisson parameter  $\lambda$  are varied in different simulations to measure the performance in various traffic scenarios.

The second network is a data center (DC) network which is shown on the right of Figure 3. It consists of 8 rack switches and 3 core switches, which make a combined 11 nodes. The network is fully bipartite connected meaning there is a link between each rack and core switch. The links have 9920Mbps capacity and 10ms latency for NS-2 considerations. We assumed a uniform traffic demand matrix between each rack switch in DC network. There are three disjoint and equal cost paths between each rack switch pair. Those paths have a length of two hops going over one of the core switches.

### B. WAN

The first Java simulation scenario in the Abilene (WAN) network is based on the traffic demand taken from [30]. To create different traffic inputs, we set  $x_m$  to three different values 3, 30 and 300. The flow arrival rate ( $\lambda$ ) is varied between 12 and 18 for  $x_m = 300$ , between 120 and 180 for  $x_m = 30$  and between 1200 and 1800 for  $x_m = 3$ . The traffic load is calculated using the formula: Traffic load  $= \lambda \times x_m \times \frac{\alpha}{\alpha-1}$ , which corresponds to between 60% and 100% link utilization with these specific values. The 100% link utilization in this network for this traffic matrix is satisfied with a traffic load of 25084Mbps. The response time results of MRTP and minMLU are given in Figure 4. Note that the response time measurements of the cases in which  $x_m = 30$  and  $x_m = 3$  are scaled by 10 and 100 to fit all three simulation scenarios into the same graph. According to the results, minMLU increases the mean response time by approximately 66% compare to MRTP.

The second Java simulation is also setup in the Abilene network but with a gravity-based traffic demand. Under this model, it requires a traffic load of 65140Mbps to achieve 100% link utilization. Therefore we updated the arrival rate parameter  $\lambda$  to achieve link utilization between 60% and 100% link utilization.  $\lambda$  is set between 3000 and 4800 for  $x_m = 3$ ,

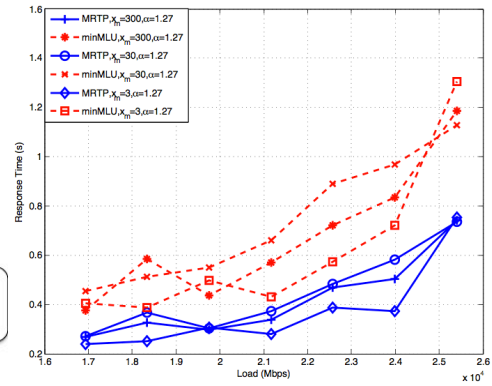


Fig. 4. Mean Response Time Comparison with Abilene Traffic Matrix

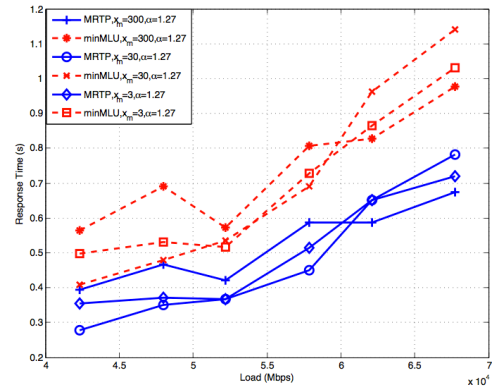
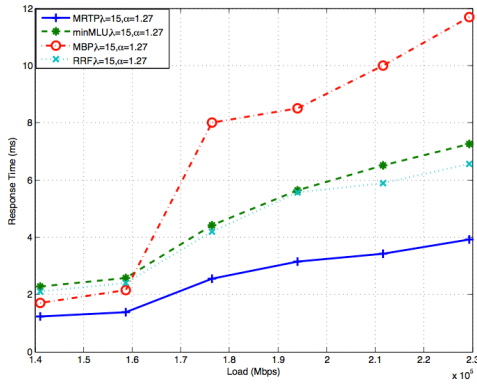
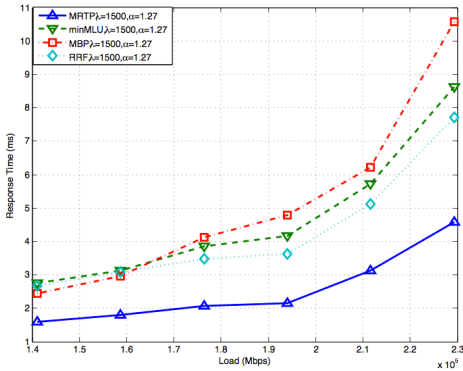


Fig. 5. Mean Response Time Comparison with Gravity-Based Traffic

between 300 and 480 for  $x_m = 30$  and between 30 and 48 for  $x_m = 300$ . The response time results of MRTP and minMLU are given in Figure 5. Note that, as before, the response time measurements of the cases in which  $x_m = 30$  and  $x_m = 3$  are scaled by 10 and 100 to fit all three simulation scenarios into the same graph. The simulation results show an approximately 42% increase in response time by minMLU compared to MRTP.

### C. Data Center (DC)

In the DC network, we carried out a Java simulation which had uniform traffic demand matrix. In addition to MRTP and minMLU, MBP and RRF are also included in the simulation. As in the WAN simulations, the traffic input parameters are arranged such that they correspond to between 60% and 100% link utilization. The DC network requires a traffic load of 229151Mbps to achieve 100% link utilization under uniform traffic model. For  $\lambda = 15$ ,  $x_m$  is set between 2000 and 3250 and for  $\lambda = 1500$ ,  $x_m$  is set between 20 and 32.5. In Figure 6 and Figure 7, the response time calculations of four different techniques for the traffic input with  $\lambda = 15$  and  $\lambda = 1500$  are presented. In both graphs, MRTP significantly outperforms the other techniques. It reduces the response time approximately 40% and 45% compared to minMLU and RRF. MBP policy performs second best in the low utilization regime but its performance deteriorates in the high utilization regime.

Fig. 6. Mean Response Time Comparison with Uniform Traffic,  $\lambda = 15$ Fig. 7. Mean Response Time Comparison with Uniform Traffic,  $\lambda = 1500$ 

#### D. NS-2 simulations

We carried out simulations in NS-2 on the WAN network. The setup of the NS-2 simulations differ from the setup of Java simulations in the nature of bandwidth sharing knowledge. In the NS-2 simulations, we follow the procedure explained in IV-C. We start with a training period where we have the perfect knowledge about the bandwidth and the response time of the flows. After the training period, the derived optimal path selections for each type of flow are used in the rest of the simulation. We call this Trained MRTP (T-MRTP). The trained optimal path selections are updated by new training periods after a certain time<sup>4</sup>. We call this method Recalibrated TM RTP (RT-MRTP).

The first set of simulations are carried out with gravity-based traffic input which requires approximately 93% MLU. We run the simulations with 20,000 flows. There are two training sets, the first one is the first 1000 flows and the second one is the flows indexed between 6001 and 6650. In Fig. 8, we compare the average response time of MRTP with minMLU, T-MRTP and RT-MRTP for traffic input with incremental sizes. For the first 1000 flows, minMLU performs approximately 72% worse than MRTP since MRTP has the perfect knowledge of the bandwidth. These results are consistent with the Java simulations. T-MRTP and RT-MRTP have a similar gain over

<sup>4</sup>As a further direction, we plan to carry out online exploitation instead of relying on periodical training sequences.

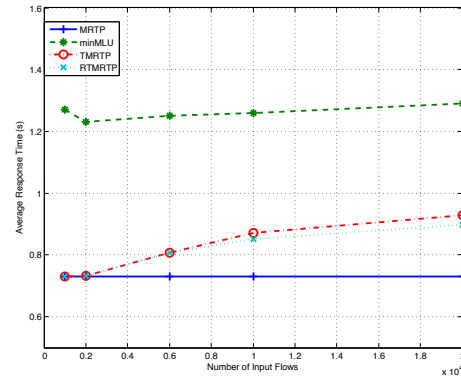


Fig. 8. Response Time Comparison of NS-2 Simulations

Traffic Set	TM RTP	minMLU	Degradation
1-350	0.650s	1.013s	55.8%
1-1000	0.658s	0.949s	44.2%
1-5000	0.710s	0.943s	32.8%
1-10000	0.764s	0.977s	27.8%

TABLE I  
AVERAGE RESPONSE TIME: WAN NETWORK/UNIFORM TRAFFIC

minMLU in the beginning of the traffic set and they manage to preserve this advantage in the rest of the traffic set. The performance degradation due to minMLU is as low as 41%. We observe that the quality of the trained optimal path selection is very high after the training set and they gradually drop as more flows come in to the network. Therefore, we see the benefit of the re-calibration of these trained path selections between flows 6001 and 6650. RT-MRTP reduces the average response time approximately 5% compared to T-MRTP. In addition, it is seen that the second training set is not as effective as the first training set because the size of the training set affects its influence on the overall performance. In addition, the fact that the second training period starts from a sub-optimal state might be a reason for the reduction in the effect of the training set. As a results, we can say the bigger and more fresh training sets improve the performance. In that case, there is a trade off between the complexity and the performance.

The second set NS-2 simulations are also carried in WAN but with uniform traffic input. There are 10,000 flows and the first 350 of them are training flows. MLU is approximately 90%. The comparative response time results of T-MRTP and minMLU are presented in Table I.

According to the results, during the training period, minMLU induces a higher 55.8% response time compared to T-MRTP. In that period, T-MRTP is the same as MRTP. The quality of the path selection parameters of T-MRTP drops from the level of MRTP as more flows come in. However, there is still a significant gain for traffic input which is 30 times bigger than the training period. The fact that the performance gain in this scenario is not as high as the gravity-based traffic can be attributed to the smaller size of the training period, which is a reiteration of the trade off between complexity and performance.



## VI. DISCUSSION

Our results point towards a significant gain in response time, but at the cost of a greater complexity. We argue that, since it is mostly software complexity at the controller, it will be eventually possible to reap the gains of fine grained resource allocation.

As a future work, the scalability could be improved by using a threshold to monitor only elephant flows and let short flows be mis-allocated without significant penalty. That would reduce the number of scheduling decisions.

The algorithm measures the total response time, but this could be modified to include other objectives. For instance, one could normalize the response time by the size of the flow, so as not to penalize short flows. We can also compute other performance objectives over the path selection.

We improve the performance significantly over minMLU, which is not however optimized for response time. However, load balancing is a proxy for reducing the congestion in the network and therefore improving the response time. Due to the current IP flow abstractions, no traffic engineering mechanism can really optimize for response time. This is where ICN abstractions offer a significant advantage.

The bandwidth estimation mechanism can be optimized as well. For a network that operates in a steady state, we take advantage of the fact that learning methods allow the network controller to assess the performance of specific flows based upon previous operations. However, we believe that a proper resource allocation should also allocate the bandwidth, and that the transport layer should be evolved to support this.

## VII. CONCLUSION

We have presented a traffic engineering architecture and an allocation algorithm that takes advantage of the abstractions of Information-Centric Networks to perform a fine grained resource allocation at the content level. Unlike IP, ICN offer the proper abstraction to increase the efficiency of the network resources and to reduce the response time of the network. We argue that, more than ubiquitous caching, it is the most significant feature of ICN.

We have seen that a controller can compute an estimated response time for a set of candidate paths and that even a relatively small set of such candidate paths can produce significant gains in response time.

We have presented the MRTP policy which allocates traffic to paths such that the total response time of the flows in the network is minimized. This is based upon an estimation of the bandwidth for each flow, which can either be estimated, or controlled by the controller. We argue that the latter is the most efficient way to allocate the resources, and proposed a bandwidth sharing function that achieved objectives of fairness and performance.

We have evaluated the MRTP over multiple network topologies and network conditions, and it significantly improves the network's response time for a wide range of network utilization and for different traffic size distributions. We have

shown reduced delay in all cases, and up to 50% improvement over min-MLU or RRF in some evaluation scenarios.

## REFERENCES

- [1] Abhishek Chanda, Cedric Westphal, and Dipankar Raychaudhuri. Content based traffic engineering in software defined information centric networks. In *Proc. IEEE Infocom NOMEN workshop*, April 2013.
- [2] A.R. Curtis, Wonho Kim, and P. Yalagandula. Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. In *INFOCOM, 2011 Proceedings IEEE*, pages 1629–1637, April 2011.
- [3] D. DiPalantino and R. Johari. Traffic engineering vs. content distribution: A game theoretic perspective. In *IEEE INFOCOM 2009*, pages 540–548, April 2009.
- [4] A. Sharma et. al. Distributing content simplifies ISP traffic engineering. *SIGMETRICS Perform. Eval. Rev.*, 41(1):229–242, June 2013.
- [5] B. Ahlgren et. al. A survey of information-centric networking. *Communications Magazine, IEEE*, 50(7):26–36, 2012.
- [6] C.-Y. Hong et. al. Achieving high utilization with software-driven wan. In *ACM SIGCOMM'13*, pages 15–26, 2013.
- [7] G. Xylomenos et. al. A survey of information-centric networking research. *IEEE Communications Surveys & Tutorials*, (99):1–26, 2013.
- [8] H. Wang et. al. COPE: traffic engineering in dynamic networks. *SIGCOMM Comput. Commun. Rev.*, 36(4):99–110, August 2006.
- [9] J. G. Dai et. al. Stability of join-the-shortest-queue networks. *Queueing Syst. Theory Appl.*, 57(4):129–145, December 2007.
- [10] M. Al-Fares et. al. Hedera: Dynamic flow scheduling for data center networks. In *Usenix NSDI'10*, pages 19–19, 2010.
- [11] M. Yu et. al. Tradeoffs in CDN designs for throughput oriented traffic. In *ACM CoNEXT'12*, pages 145–156, 2012.
- [12] N. McKeown et. al. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008.
- [13] R. Clegg et. al. A critical look at power law modelling of the internet. *Comput. Commun.*, 33(3):259–268, February 2010.
- [14] S. Jain et. al. B4: experience with a globally-deployed software defined WAN. In *ACM SIGCOMM'13*, pages 3–14, 2013.
- [15] S. Y. Fayyazbakhsh et. al. Less pain, most of the gain: incrementally deployable ICN. In *ACM SIGCOMM'13*, pages 147–158, 2013.
- [16] V. Gupta et. al. Analysis of join-the-shortest-queue routing for web server farms. *Perform. Eval.*, 64(9-12):1062–1081, October 2007.
- [17] W. Jiang et. al. Cooperative content distribution and traffic engineering in an ISP network. In *ACM SIGMETRICS'09*, pages 239–250, 2009.
- [18] B. Fortz and M. Thorup. Internet traffic engineering by optimizing ospf weights. In *IEEE INFOCOM'00*, volume 2, pages 519–528, 2000.
- [19] Van et. al. Jacobson. Networking named content. In *ACM CoNEXT'09*, pages 1–12, 2009.
- [20] Wolfram Lautenschlaeger. A deterministic tcp bandwidth sharing model. *CoRR*, abs/1404.4173, 2014.
- [21] Ingmar et. al. Poese. Enabling content-aware traffic engineering. *SIGCOMM Comput. Commun. Rev.*, 42(5), September 2012.
- [22] M.J. Reed. Traffic engineering for information-centric networks. In *IEEE International Conference on Communications (ICC)*, June 2012.
- [23] K. Roy and C. J. Tomlin. Solving the aircraft routing problem using network flow algorithms. In *American Control Conference*, volume 1B, pages 1077–1083, 2007.
- [24] A. Sharma, A. Mishra, V. Kumar, and A. Venkataramani. Beyond MLU: An application-centric comparison of traffic engineering schemes. In *INFOCOM, 2011 Proceedings IEEE*, pages 721–729. IEEE, April 2011.
- [25] Kai Su and Cedric Westphal. On the benefit of information centric networks for traffic engineering. In *IEEE ICC Conference*, June 2014.
- [26] A. et. al. Wundsam. Network troubleshooting with mirror vnets. In *IEEE GLOBECOM Workshops*, pages 283–287, Dec 2010.
- [27] Haiyong Xie, Guangyu Shi, and Pengwei Wang. TECC: Towards collaborative in-network caching guided by traffic engineering. In *IEEE INFOCOM'12*, pages 2546–2550. IEEE, March 2012.
- [28] Jin Y. Yen. Finding the k-shortest loopless paths in a network. *Management Science*, 17(11):712–716, Jul. 1971.
- [29] Ossama Younis and Sonia Fahmy. Constraint-based routing in the internet: Basic principles and recent research. *Communications Surveys & Tutorials, IEEE*, 5(1):2–13, third quarter 2003.
- [30] Yin Zhang. Abilene traffic matrices, <http://www.cs.utexas.edu/~yzhang/research/AbileneTM/>.