MAPS: Adaptive Path Selection for Multipath Transport Protocols in the Internet

Yu Chen  Xin Wu  Xiaowei Yang
Department of Computer Science, Duke University
{yuchen, xinwu, xwy}@cs.duke.edu

ABSTRACT
As the access links of the Internet get faster, there is an incentive for the end hosts to demand higher throughputs. To meet their demands with minimum change at the Internet core, it is necessary to explore the existing core capacity efficiently. Multipath transport protocols try to address this problem by allocating traffic efficiently and fairly over some preselected paths, thus utilizing the path diversity of the network. However, because the Internet is heterogeneous, i.e., different parts of the Internet have different capacities, it is critical to select those paths with great care. Without efficient path selection, some of the flows might have to pass the more congested part of the network. As a result, their throughput might be significantly lower than using another set of paths. Adaptive path selection for multipath transport layer (MAPS) solves the problem by actively probing the residue capacity of unused paths for each flow. Moreover, MAPS only initiates the probes sparingly, depending on the real-time quality of current subflows. Our simulation shows that with the efficient path selection scheme by MAPS, the underlying transport layer is able to improve the average throughput of bulk transfer by as much as 40%; moreover, it is also able to improve the fairness index significantly when the traffic demand is high.

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

General Terms
Adaptive Path Selection, Multipath Transport Protocols

1. INTRODUCTION
The access links of the Internet are faster than ever before. For example, in 2012, homes in Kansas City will have access to the 1Gbps Fiber Network provided by Google [2]. This trend poses new challenges to the protocol design. With the existing transport protocols like TCP, there will be more shared bottlenecks and congestion at the Internet core. Since it is going to be expensive to upgrade the core, it is natural to ask whether there exists a solution at the protocol level.

DETOUR [10] shows the possibility of better congestion control by routing through an alternate path. The intuition behind it is simple. The Internet is heterogeneous. Different parts of the Internet have different congestion levels. The more path diversity we have, the less likely we are going to stuck at one particular congested path.

Recently, there have been attempts to apply multipath TCP to datacenter networks. Given a flow with a small set of subflows, each of them using a preselected path, MPTCP tries to allocate traffic to those subflows. Compared with TCP, MPTCP is able to use the available bandwidth in more than one paths. Moreover, it also improves the fairness over TCP.

Because there is also a lot of path diversity in the Internet, there is an incentive to ask how to apply multipath TCP to the Internet. Our design of MAPS is tailored for the Internet topology. Instead of having a few preselected paths like MPTCP, it is possible to generate hundreds of paths with algorithms like Path deflection [11] and path splicing [8]. Both of them are essentially source routing algorithms. They generate different paths by varying the tags or the pathlets at the source.

Similar to MPTCP, we want to use a small set of paths for the subflows. Thus we need to select a subset of paths from the paths generated by those algorithms. The question is whether the path selection algorithm makes a difference. If so, it is also important to have an algorithm to select a set of good paths for each flow.

We answer those two questions in our design. We first show that because of the underlying heterogeneity of the Internet, not all the paths are equally good. Second, because of the bias of the path generation algorithms towards the links with more paths connected to them, a random path selection algorithm is not going to select all the links with equal chance. Even if all the links have the same capacity, some of them might not be selected at all, thus waste the capacity.

With a simple dumbbell topology, we also show that the underlying transport protocol is not always able to offset the inherent deficiency of a set of paths, so as to achieve the fairness and efficiency of traffic allocation to
the flows. While there is a chance for us to get a set of bad paths with a random selection, it is necessary for us to have an algorithm to select a set of good paths for each flow.

In our design, we make 3 choices. First, we use either path deflection or path splicing as our path generation protocol. Second, because of the possible large bandwidth-delay product, we enable the explicit congestion control at the routers, to give faster numerical feedback to the sources. Third, since a set of bad paths might give a poor performance for the underlying transport protocol, we have a separate adaptive path selection layer (MAPS).

MAPS is based on informed probes. Each flow makes a periodic decision on whether to probe, depending on its estimation on the real-time quality of its own set of paths. With informed probes, we do not have to initiate probes every round, thus reduce the interference of the probing traffic on the normal traffic. Moreover, MAPS use the data packets as the probes without incurring any network overhead. The probes enable us to discover the capacity of some previously unused paths, and switch to those paths when they are significantly better than one of the paths in use. While multipath transport protocols like MPTCP try to exploit the capacity of existing paths, MAPS aims to explore the path diversity of the network.

By comparing it with the random path selection, we conclude that MAPS is able to shift flows from more congested paths to less congested ones. This prevents the congestion from building up. The experiments indicate that MAPS is especially helpful in removing the traffic from the hotspots when the traffic demand is heavy. Our results also show that MAPS improves both the overall throughput and the fairness of the traffic.

2. MAPS

Because MPTCP is the first practical multipath transport protocol in datacenters, comparing the difference between the datacenter network and the Internet will give us valuable insights. With those insights, we will have a better understanding of the criteria for a good Internet architecture, so that the multipath transport protocols can work in it. Similar to the datacenter architecture in MPTCP [9], we need to consider two important components in our design, the underlying topology and traffic pattern. In this paper, we only compare the topology in this section.

Datacenter networks are hierarchical. There are usually one or very few paths from an endhost to a core switch, with exactly the same length. Hence for two hosts, there are not many paths connecting them. Moreover, the links at the same layer usually have the same capacities. For two endhosts, because the layers along all the paths are the same, those paths always have exactly the same link characteristics. The propagation delay for the links are usually short, because the switches and the endhosts usually resides in the same data center.

Compared with the datacenter networks, the Internet topology is irregular and heterogeneous. Given a source-destination pair, there are many more paths between them than the datacenter networks, each of them having different hop counts and costs. The capacities for two different paths might not be the same at the same hop. Furthermore, the delay for the links is also significantly larger than those in the datacenter networks.

With those differences in mind, our design is different from the data center architecture in the following ways:

1. A path generation layer In datacenter architecture, the paths can be identified by the cores. In our design, a path generation layer is used to generate the paths and map the path identifiers to different paths.

2. A path selection layer In datacenter architecture, the paths between two endhosts are homogeneous, so a random selection of paths as subflows suffices for the multipath transport layer. In our design, we need a separate path selection algorithm. It enables us to select the better paths, thus make better use of the network capacity.

3. MPXCP as the underlying transport protocol The Internet has a much higher propagation delay than the datacenter networks. We borrow the idea from Explicit Congestion Control (XCP) [4]. It works better than TCP in high delay-bandwidth networks. Thus, to combat the increasing demand at the endhost, an explicit path control is going to help. Section 2.2 is going to give more detail about MPXCP.

2.1 Path Generation

Define a path identifier as a number or vector specifying the choices at each intermediate hop. The path generation layer maps the path identifiers to different paths. Routing deflection and path splicing are two different path generation algorithms. Both of them are essentially source routing algorithms, i.e., the path identifier is specified by the source and transmitted inside the network. Each router is going to decipher the path identifiers and choose the next hop from a set of candidates. Both of them ensure the loop-free property by requiring the next hop to have lower cost than the previous hops. Both of them are able to generate enough number of paths, with relatively small overhead.

Despite the advantages above, the paths generated by those algorithms are biased towards certain links. For example, define two links as link-disjoint for a source-destination pair if they never share a path from the source to the destination. Suppose \( l_1 \) and \( l_2 \) are two link-disjoint
bottlenecks for two different set of paths, as shown in Figure 1. It is highly likely that the number of paths connected to \( l_1 \) and \( l_2 \) are different. Without loss of generality, assume \( l_1 \) has more paths connected to it than \( l_2 \). In this case, if the path selection is completely random, i.e., the algorithm randomly chooses a path identifier and uses the path corresponding to the identifier, it is more likely to choose \( l_1 \) than \( l_2 \).

As a result, with a small set of randomly selected paths for a flow, it is quite likely many subflows are sharing the same bottleneck, at the same time leaving the other links underutilized. In fact, in the example above, it is highly likely that the path containing \( l_2 \) is not selected at all. Inclusion of paths with \( l_2 \) as well, however, might give us much better throughput, depending on the real-time quality of the path containing \( l_2 \). In order to fully utilize the capacity of the network, it is necessary to have an additional path selection layer in our architecture.

![Figure 1: Example of possible bias of the path generation algorithms towards \( l_1 \) over \( l_2 \)](image)

### 2.2 Congestion Control

Of all the multipath transport protocols, MPTCP is the first practical solution to effective path selection and congestion control in datacenter networks. Compared with running separate TCPs for individual subflows, the key difference of MPTCP is to apply the “linked increases” instead; that is, in the additive increase stage, all the subflows increase collectively by a constant. Subflows with large windows increase faster, thus effectively move the traffic away from the congested paths.

The traffic of the Internet, however, is very different from the traffic inside the data centers. Delay is one fundamental difference. Because the geographic distance between two routers are usually much larger than the distance between two switches inside a data center, the propagation delay is much larger in the Internet. As the demands from the endhosts increase, the bandwidth of the Internet is also going to increase. To meet the challenges in networks with higher bandwidth-delay product, we design MPXCP by combining the feedback method from XCP and the control rules from MPTCP. MPXCP moves the control rules to the routers. The routers allocates a portion of a link’s residue capacity to the sources sharing the link instead. The routers compute the amount of change in window size for each source, by applying the aggregated additive increase, and independent multiplicative decreases. When the link is close to full utilization, we shuffle a small portion of traffic from all the sources. How much traffic to shuffle is a tradeoff between fairness and convergence speed. After that, the routers sends the change back to the sources in the acknowledgements. The source applies the change on receiving them. MPXCP is stable and converges faster than MPTCP in our simulations.

Both MPTCP and MPXCP assumes that it is possible to find one good unloaded path among many randomly selected paths. However, the problem is whether they are still able to find one good unloaded path, if the set of selected paths is small in size. While wishful thinking tells us with enough flows, it is always possible for MPTCP and MPXCP to shift enough traffic from congested paths to those less congested ones, there is a limit on the possible adjustments at the transport layer. For example, consider the dumbbell topology in Figure 2. Suppose there are 100 flows from Node 1 to Node 5 at the same time. For each flow, we have 2 subflows. There are three paths from Node 1 to Node 5, they are 1-5, 1-2-3-5 and 1-2-0-4-3-5, with bottlenecks from Node 1 to Node 5 (Bottleneck 1), from Node 2 to Node 3 (Bottleneck 2) and from Node 0 to Node 4 (Bottleneck 4). Since there are 2 subflows for each flow, we can only select 2 bottlenecks for each flow. For those 100 flows, let \( r_{ij} \) denote the number of flows with Bottleneck \( i \) and Bottleneck \( j \) in their subflows. Consider an arbitrary time interval, to be efficient, every bottleneck should reach its full capacity, thus the amount of traffic transmitted through each bottleneck should be proportional to its bandwidth during any time interval. By fairness, we mean all the flows should have the same rate \( \lambda \) at any time. Thus at any time interval \( t \), the amount of traffic on Bottleneck 2 should be exactly \( \frac{155}{155 + 40 + 9} \lambda t \). Since the maximum amount of traffic on Bottleneck 2 is \( (r_{12} + r_{23})\lambda t \), we have

\[
\frac{(r_{12} + r_{23})\lambda t}{(r_{12} + r_{23} + r_{31})\lambda t} \geq \frac{155}{155 + 40 + 9} \quad (1)
\]

\[
\Rightarrow \frac{r_{31}}{r_{12} + r_{23} + r_{31}} \leq \frac{49}{204} < \frac{1}{4} \quad (2)
\]

A random selection of paths is expected to select Bottleneck 1 and Bottleneck 3 in the subflows for about one third of the time. While the gap between \( \frac{1}{3} \) and \( \frac{1}{4} \) is
small, it has an impact on the performance of the underlying transport protocol, as we will show in Section 3.

![Diagram of the dumbbell topology with heterogeneous bottlenecks](image)

**Figure 2: The dumbbell topology with heterogeneous bottlenecks**

### 2.3 Path Selection

As shown above, since the Internet is heterogeneous and the path generation layer is biased towards certain links, to achieve fairness and efficiency, the random selection of paths might not suffice for the underlying protocols. Thus in our architecture, we have an extra layer for path selection.

Our solution is based on probabilistic probes. It shifts the subflow from the path with the least throughput (the worst path) to the probing path, if the probing path has a consistently higher throughput over a long scale. In our design, we need to improve the performance of MPXCP without incurring much probing overhead. Besides, the probes should have little impact on the stability of MPXCP. To achieve these goals, our design makes careful decisions on the following three questions:

#### 2.3.1 When shall we initiate the probes for the new paths?

Periodically, a flow needs to check whether to make a new probe. The decision is based on the quality of the worst path every 4 RTT. The intuition is that the throughput of other paths might give us hints about the network capacity. If the worst path does not perform significantly worse than the others, the probe has a high chance of failure. However, if the worst path is indeed much worse, a probe is perhaps a good choice. Thus, \( Q = \frac{X_{\text{worst}}}{X} \) is a good metric to reflect the difference, because it measures the ratio between throughput of the worst path and the average throughput. Here we define the probing probability \( P \) as in Equation 3. Besides \( Q \), there is an exploration factor \( p \) in the equation. This exploration factor enables the flow to explore new paths, even if all the paths have equally bad throughput. For MAPS, we set \( p = 0.1 \).

\[
P = (1 - \frac{X_{\text{worst}}}{X})(1 - p) + p \quad (3)
\]

Since the path generation algorithm is able to generate hundreds of different paths for a large network, to explore more paths, we start the timer for a path when it initiates a probe. MAPS will not select a path before its timer expires. Since MAPS mainly probes the topology instead of the traffic pattern, we want to try as many paths as possible, thus we set \( p = 40\text{RTT} \).

#### 2.3.2 How will we design the probes?

In MAPS, we use the data packets as the probes. Thus MAPS is compatible with the endhosts without the path selection layer. By sending packets over a probing path, we continuously track the congestion window and RTT for the new path and the worst path. We compute the throughput for a path by dividing the size of its congestion window with its RTT. While this value oscillates a lot because of the sawtooth behavior of the transport protocols, if a path consistently outperforms the worst path over a long timescale, we are confident it is a better path regardless of future traffic pattern. We believe the history profile is a better measurement than a single value because it reflects the trend, thus it enables us to have a stable decision over a relatively long run.

#### 2.3.3 When shall we stop the probes and what shall we do after that?

Given the history profile of the throughput as defined in Section 2.3.2, MAPS switches to a new path only when it grows to twice the throughput of the worst path without stop and stays above that threshold for probation period of 10 consecutive RTT. If the throughput of the probing path decreases at the additive increase phase, or goes below the threshold during the probation period of 10 consecutive RTT, the source of the flow is going to switch back to the fallback mode.

Since with each failed probe, MAPS has more information about the topology. It is intuitive for MAPS to try more rounds. We immediately decide whether to start next probe by computing the probability including the throughput of the new failed probe. We repeat this process if the probes keep failing, and stop it when the probability decides so, or the number of consecutive probes in a round exceeds a threshold. In MAPS, we set this threshold to 5.

### 3. EVALUATION

In our simulation, we run our ns2 simulation in the dumbbell topology as shown in Figure 2. We use the dumbbell topology for 2 reasons. First of all, it is easy to control. With a simple traffic pattern, it is always possible to predict and choose appropriate bottlenecks. Secondly, it is a good model for us to get insights about the Internet core. To simulate the heterogeneity of the real Internet, we set the bottleneck links to different capaci-
ties in Figure 2. To simulate the path generation algorithms, since the number of paths is limited, we are able to compute the probability for the choice of each path. While it is more interesting to study the real traffic pattern on real topologies, we left it as future work.

Previous work [5] has shown that the web traffic can be modeled by the ON/OFF streams, where each file transmission is followed by an idle “thinking” period. It also shows that both the file sizes and the thinking time have a Pareto distribution, with shape parameters 1.12 and 1.50 respectively. In our simulation, we control the congestion level by the demands of ON/OFF traffic. With increasing file size and decreasing thinking time, there will be more traffic on the fly, so that we can increase the congestion level of the network.

In each simulation, we compare the performance of two versions of MPXCP. One of them uses the random selection of paths (MPXCP-RANDOM), the other runs with MAPS enabled (MPXCP-MAPS).

3.1 Traffic in Networks with Heterogeneous Paths

In our simulation, we have 2 source-destination pairs, one from Node 0 to Node 4 and the other from Node 1 to Node 5. Each source-destination pair has 100 flows between them. For each source-destination pair, it is going to choose 2 paths out of 3 available paths. We run each simulation for 7200 seconds. We define a flow as a short flow if it runs for less than 2 seconds. For short flows, we never run MAPS for them, because with MAPS, the probing paths will be at the slow start for most of the time, thus it will the overall throughput of the network.

3.1.1 Networks with Interactive Web Traffic

To evaluate MAPS on interactive web traffic, we run the simulation in a network with many small files and a long average thinking time. In our simulation, we set the average size of the files to 300KB, the size of the web pages in 2008 [1]. We set the average thinking time to 30s, according to [5].

Figure 3 shows the aggregated throughput for three different bottlenecks over a 7200 seconds’ interval.

These figure show us that for most of the time, all the three bottlenecks are not able to reach their full capacities. The spikes occur occasionally. They might occur during the transmission of large files, or bursts of small transfers.

However, there is conspicuous difference starting from 640 seconds. The sudden increase of aggregated throughput lasts for about 1500 seconds for MPXCP-RANDOM, while for MPXCP-MAPS, it lasts for only about 300 seconds. To find the reason, We examined the data file and found that MPXCP is transferring a 7GB stream. MPXCP-MAPS used Bottleneck 2-3 in one of its sub-flows, while MPXCP-RANDOM does not. We also checked the time interval for transferring such a stream, the numbers are consistent with what we observed above. Hence, in a network with very few bulk transfers, MAPS makes a better decision for some of them.

3.1.2 Inter-datacenter Networking

Nowadays, the interactive web traffic is no longer the dominating traffic in the Internet, as in the 1990s. More data is moving to the data centers. According to [6], Google alone covers about 5.2% of the Internet traffic in 2009. The service providers like Google usually have a few data centers in different parts of the world. The companies need to synchronize data between those data centers, because 1) The geographic diversity can protect the backups of data even in wars or disasters. 2) It reduces the delay for the data to reach the local customers. While the data transferred between the datacenters is already huge nowadays, the amount is still increasing.

According to [7], the demands on a data center show strong diurnal patterns. Since the diurnal pattern is generally stable over a short time scale, MAPS helps those inter-datacenter traffic to select more good paths, thus make better use of path diversity among those datacenters. Of all the inter-datacenter traffic, the client-triggered video transfer is one important class [3]. In this paper, we model the client-triggered video transfer as ON/OFF traffic with an average stream size of 10 MB and an average thinking time of 1 second.

Figure 4 plots the aggregate throughput for all the three bottlenecks.

Bottleneck 0-4 and 1-5 are both able to achieve their full capacity for MPXCP-RANDOM and MPXCP-MAPS. However, for bottleneck 2-3, the traffic for MPXCP-RANDOM is unable to fully utilize its capacity. The time graph shows a bursty behavior for its traffic. MPXCP-MAPS, however, is always able to fully utilize its bandwidth.

Figure 5 compares the Jain’s fairness for all the flows at each time interval. MPXCP-MAPS is able to achieve a fairness index of close to 1 for most of the time; The
bursty throughput in Figure 4. Since very few flows have high throughput when others starve, the fairness index of MPXCP-RANDOM, however, is very low for most of the time.

As we have discussed in Section 2.2, MPXCP-RANDOM might be inefficient and unfair in the network with heterogeneous paths. However, that section did not explain why there is such a huge difference between MPXCP-RANDOM and MPXCP-MAPS. In an attempt to explain the difference, we plot the percentage of flows choosing Bottleneck 0-4 and Bottleneck 1-5 in their subflows. The ratio for MPXCP-MAPS is always below 0.25. This ratio for MPXCP, however, is quite surprising. It starts from 1/3, the theoretical value for MPXCP-RANDOM, and increases very fast. In less than 1500 seconds, it is very close to 1. We explain the reason for the clustering effect as follows. Since Bottlenecks 0-4 and 1-5 have a significantly lower bandwidth than Bottleneck 2-3, when more long flows have them in their subflows, the amount of bandwidth shared by those flows reduces even further, thus further delays the completion time of those flows. Eventually, we observe most of the flows sharing those bottlenecks at the same time, because the little share of the slow bottlenecks makes them unlikely to finish in a short time. Since most of the flows are trapped in those slow bottlenecks, very few flows are using the much faster Bottleneck 2-3, thus resulting in its bursty throughput in Figure 4. Since very few flows have high throughput when others starve, the fairness index is pretty low in Figure 5. Because MPXCP-MAPS is able to shift traffic from Bottlenecks 0-4 and 1-5 when they are shared by too many flows, it achieves a higher aggregated throughput and better fairness.

4. CONCLUSION

This paper explores the design challenges of applying multipath transport protocols to the Internet. One of them is to select a set of paths for a flow to use. As far as we know, the heterogeneous bandwidth of the Internet and the bias of path generation algorithms toward certain links make the path selection an important choice. In many cases, random selection of paths is not a good choice for MPXCP. We designed a new path selection algorithm MAPS and showed its improvement over the random selection of paths.

While multipath transport protocols provide an attractive solution to improve throughput in the Internet, we believe we still need to solve many unexplored challenges for it to really work. In the future, we are going to explore the problem of path selection on real topologies and traffic patterns. We believe it is an important step for making the multipath transport protocols really practical in the Internet.

5. REFERENCES

