Network and End-System Support for Transparent Use of Multiple Paths

Murtaza Motiwala, Megan Elmore, Yogesh Mundada, and Nick Feamster
College of Computing, Georgia Tech

ABSTRACT

We demonstrate the implementation of a system that allows applications to use multiple paths in the network. The system consists of an end-system component, which includes a socket capture library, monitoring daemon, and a Click router; and a network component, which includes a Click router which allows traffic to be redirected along different paths. The system does not require any application modification and is independent of the multipath routing protocol that is implemented in the network. We demonstrate our system can provide fast recovery from network failures for several real-time applications and also show how applications can use the system's interface to achieve higher throughput.

1. Introduction

When communication networks experience failures or disruptions, networked applications can face downtime on the order of minutes as routing protocols re-converge to find new working paths. Multipath routing presents a way to mitigate this problem: when a failure gets detected, routers may forward traffic along an alternate path or paths. Although multipath routing has received significant attention from both research and industry, it has yet to see wide-area deployment. We believe that one significant reason for this slow adoption rate is the difficulty of implementing a multipath routing mechanism that provides a simple interface with significant control over paths to applications, while still allowing routing protocols to scale. The primary goal of this demonstration will be to show a simple, scalable multipath routing mechanism that provides fast recovery from failures and provides end systems and applications a simple mechanism for accessing multiple paths through the network.

Two of the challenges in designing a multipath routing protocol that masks network failures are defining and detecting network failures. First, the notion of failure is often application-specific. For example, instant messaging applications would want to avoid high latency, and VoIP communications become unintelligible when jitter exceeds a few tens of milliseconds. Second, requiring network devices to detect these end-to-end failures would be difficult, since these devices do not have direct access to end-to-end connectivity information. To scale and adapt to different applications' needs, failure recovery is better achieved by cooperation among end systems.

Many existing mechanisms allow end systems to select network paths. RON [1] uses an overlay network to let end-systems use non-default paths, and schemes like Routing Deflections [4] and Path Splicing [3] present a way of letting end systems access multiple paths in the network in a scalable fashion. Unfortunately, applications and end systems lack a consistent API for accessing alternate paths. To enable such access, these systems would need a simple and sufficiently powerful API for path selection, a mechanism for monitoring availability and performance along alternate paths, and a mechanism for switching paths. To design and deploy such a system in existing networks, we must address the following issues:

- *Transparent integration:* The system must be incrementally deployable and must work even if all routers in the network do not support the use of multiple paths. Also, it should be possible to run applications unmodified on the end system.
- *Non-intrusive and lightweight monitoring:* Path quality monitoring involves performing path (both, active and passive) monitoring, finding best available path and switching to a different path after weighing path advantage over switching cost. The active monitoring should not cause disruptions in the network traffic and should consume minimum network resources.
- *Stability:* Path characteristics keep changing under dynamic network conditions, which can lead to frequent modifications to the current best path. Excessive switching can nullify the benefits of the new path and must be avoided.

In this demonstration, we will show how an implementation of an end-system API for multiple paths in the network, along with the ability to monitor alternate paths in real time, can be used to allow applications to realize the benefits of multipath routing in practice. We also plan to release the implementation of the system, and during the demo, we will inform other interested researchers about how to use our implementation.

2. Design

The system has two components: support at the end-system and support at routers in the network.

2.1 End-system Support

The end-system support consists of an interface for applications to specify the networking performance metrics which are useful for them and a monitoring daemon which monitors the path quality on behalf of the applications. Figure 1 gives an overview of the extension to the end host. Routers in the network support multipath routing, in our present implementation, we have implemented Path Splicing [3], where the routers store multiple forwarding paths

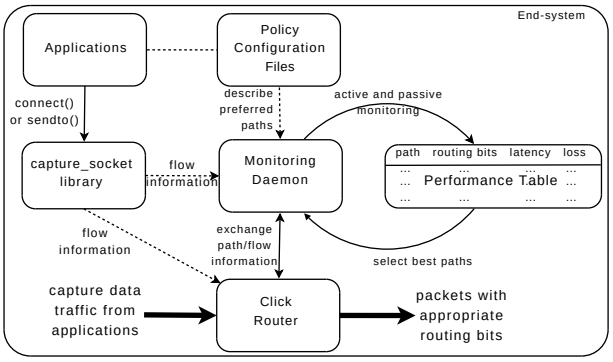

Figure 1: Design of the different components of the failure recovery system at the end system. Applications supply policy configuration files to the monitoring daemon which uses the information to monitor performance of paths. Click router adds appropriate routing bits to data traffic as specified by the monitoring daemon.

("slices") for each destination and read the packet header to decide which "slice" to forward the packet.

Socket Capture Library Applications that wish to use multiple paths may do so with no modifications. To keep track of a set of desirable paths for each application, the system provides a socket capture library that intercepts `connect()` calls for TCP and `sendto()` calls for UDP flows. The library is used to determine when new flows are being initiated from the end host and works transparent to the application as shown in Figure 1.

Monitoring Daemon The monitoring daemon runs as a separate process at the end-host. The purpose of the monitoring daemon is to monitor available paths in the network on behalf of the applications running on the end-host. The monitoring daemon receives a notification from the socket capture library whenever a new flow is initiated by an application.

2.2 Network Support

The second component of the system is the support for Path Splicing [3] in the network.

Click-based splicing enabled routers We enable multipath routing schemes by installing Click software routers ([2]) at end-hosts and in the network. End-hosts (as shown in Figure 1) are responsible for applying the desired routing bit sequence for a flow in the identification field of the IP header. For Path Splicing, this sequence consists of eight two-bit codes, each corresponding to the slice on which to forward at each hop. The in-network routers contain a Click element that can decipher the sequence in the IP-ID field; this design choice is extensible: users of our system can substitute the Click element that understands splicing for one that understands the multipath routing scheme of their choice.

Figure 2 shows an example of using a spliced path. The IP-ID field encodes the forwarding tree to be used at each of the routers on the path. Routers read the appropriate bit position in the IP-ID field (indexed by the TTL field) to know the correct forwarding table ("slice") to use to forward the packet on.

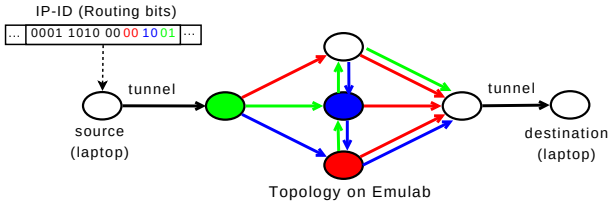

Figure 2: Small Toy topology: The source and destination nodes will be laptops connected via tunnels to a topology set up on Emulab. The different colors show the different slices in the case of splicing.

3. Demonstration Goals

The demonstration will answer the following questions:

How can researchers set up the end-system and network components? We will show how to set up the end-system components (socket capture library, monitoring daemon and Click router) at the end-host. We will run a few standard network applications and show how they can use multiple paths in the network without modifying the application binary. We will also show how to set up a network topology on Emulab with custom Click routers.

How can real-time applications benefit from multiple paths? We will run a set of real-time applications like video and VoIP between two end hosts connected via a multipath network. We will then introduce faults in the network (*e.g.*, packet loss, high latency) which affects the default shortest path between the hosts. Next, we will enable the use of multiple paths and monitoring at end hosts and show real-time recovery of communication between the end hosts.

How can applications increase their throughput by using multiple paths? We will show the increase in throughput-that multipath routing can achieve for several throughput-intensive applications. We will demonstrate this by a simple file transfer application written to exploit path diversity.

How much overhead does the system introduce? We will demonstrate the low CPU, memory, and bandwidth overheads for our implementation.

How to use a different multipath routing scheme? We will show how to replace the multipath scheme in the network with some other multipath scheme that uses bits in the packet header to indicate the use of alternate paths (*e.g.*, Routing Deflections [4]).

REFERENCES

[1] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proc. 18th ACM Symposium on Operating Systems Principles (SOSP)*, pages 131–145, Banff, Canada, Oct. 2001.

[2] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, Aug. 2000.

[3] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala. Path Splicing. In *Proc. ACM SIGCOMM*, Seattle, WA, Aug. 2008.

[4] X. Yang, D. Wetherall, and T. Anderson. Source selectable path diversity via routing deflections. In *Proc. ACM SIGCOMM*, Pisa, Italy, Aug. 2006.