Network Working Group F. Devetak
Internet-Draft S. Kapoor
Expires: March 16, 2018 Illinois Institute of Technology
September 12, 2017

Dynamic MultiPath Routing
draft-kapoor-rtgwg-dynamic-multipath-routing-01

Abstract

In this draft we consider dynamic multipath routing and introduce two methods that use additive increase and multiplicative decrease for flow control, similar to TCP. Our first method allows for congestion control and re-routing flows as users join in or leave the network. As the number of applications and services supported by the Internet grows, bandwidth requirements increase dramatically so it is imperative to design methods to ensure not only that network throughput is maximized but also to ensure a level of fairness in network resource allocation. Our second method provides fairness over multiple streams of traffic. We drive the multiplicative decrease part of the algorithm with link queue occupancy data provided by an enhanced routing protocol.

Status of This Memo

Copyright Notice

1. Introduction

Internet packet traffic keeps growing as the number of applications and services it supports as well as their bandwidth requirements explode. It has then become necessary to find ways to ensure that network throughput is maximized. In this draft we propose dynamic multi-path routing to improve network throughput.

Multipath routing is important, not only for throughput but also for reliability and security. In multipath routing, improvements in performance are achieved by utilizing more than one feasible path [M75]. This approach to routing makes for more effective network resource utilization. Various research on multipath routing have addressed network redundancy, congestion, and QoS issues [CRS99] [ST92]. Prior work on multipath routing includes work on bounding delays as well as delay variance [DSAK11] [SKDA13].

The prior work is primarily from the viewpoint of static network design but, in practice, congestion control is necessary to prevent some user flows from being choked due to link bottlenecks. Single path routing implementations of TCP achieve that by rate control on specified paths. TCP is able to handle elastic traffic from applications and establishes a degree of fairness by reducing the rate of transmission rapidly upon detecting congestion. Regular TCP has been shown to provide Pareto-optimal allocation of resources [PU12]. However, unlike the single path approach of TCP, we consider multipath routing with associated issues of path selection and congestion. We may note that multipath TCP (MPTCP) has been studied extensively [RG10] [GWH11] [RH10] [AP13] with a number of IETF proposals [M05] [M06] [M07] [M08]. Prior work on multipath TCP is defined over a specific set of paths and the choice of paths or the routing is independent of congestion control; determining the right number of paths thus becomes a problem. The variation of throughput with the number of paths has been illustrated in [RG10] [GWH11]

Along with consideration of congestion, we also need to ensure a level of fairness in network resource allocation. Factoring fairness into the protocol is important in order to prevent some user’s flows from suffering due to bottlenecks in some links. Based on mathematical optimization formulations, we consider route determination methods that ensure fairness where all users can achieve at least a minimum percentage of their demand. We introduce an algorithm that uses additive increase and multiplicative decrease for flow control and we have experiments to illustrate its stability and convergence. The algorithm may be considered as a generalization of TCP.

We have performed an extensive set of simulations using the NS-3 simulation environment. In our implementation we drive the multiplicative decrease part of the algorithm using queue occupancy data at each outgoing network link, with that data provided by an enhanced routing protocol. For a more in-depth evaluation of the algorithm’s performance we simulated not only the fairness algorithm but also a version of the same without the fairness component. We also performed and compared simulations using standard TCP and TCP with ECMP enabled.

2. Joint Routing and Congestion Control: Preliminaries

The Joint Routing and Congestion Control utilizes the Link State of the network. The algorithm utilizes a price variable that models congestion at each link and a variable that models the fairness coefficient. The fairness coefficient is used to establish the same percentage of traffic is being routed for multiple source-sink pairs.

2.1. The Price function

Each edge of the network has a price function associated with it, referred to as P(e). The price function measures the congestion on the link. The price function lies in the interval [0,1] and is 0 if the edge occupancy is low and 1 if the edge occupancy is high.

In theory the edge occupancy is given by f(e)/C(e) where f(e) is the amount of traffic on the link and C(e) is the capacity of the link. In practice, the edge occupancy is measured by the congestion in the queue serving the link.

The price function increases as the congestion grows. This function’s values will be referred to by a price variable on link e which is denoted by PBQ(e)

2.2. The Fairness function

The price function is complemented by an "increase" function, i.e. a variable that regulates the amount of traffic changes based on the fraction of traffic of the commodity that has been routed. This function, th values represented by the variable PBF(s-t), is used to model the fairness. This variable is related to the source-destination pair, denoted by s-t, whose requirements are being satisfied.

The variable PBF(s-t) starts with an initial value and goes down to zero as the requirement of s-t is being increasingly met. The increase in PBF is dictated by a fairness co-efficient, Gamma.

The formula for PBF(s-t) is 1- y(s-t)/[Gamma *d(s-t)] where Gamma is the fairness co-efficient, d(s-t) is the demand and y(s-t) is the amount of requirement that is being met.

3. Joint Routing and Congestion Control: Algorithm

We present the details of the algorithms below

3.1. Preliminaries

Let T be the time interval used to increment or modify routing.

We use two coefficients for each path Pi

1. Additive increase coefficient: A positive value ai by which we increment the flow on a path at each iteration xi(t) = xi(t-T) + ai

2. Multiplicative decrease coefficient: The coefficient bi that we apply to decrement flows: xi(t) = (1-bi)xi(t-T) where x, t, T are the same as above.

We utilize multiple methods for calculating bi:

METHOD 1: bi may be computed as follows:

1. bi =0 if no edge on the path Pi is congested.

2. bi =0.5 if one edge on the path Pi is congested.

3. bi= 1.0 if more than one edge on the path Pi is congested

METHOD 2: bi may be computed as follows:

1. bi = 1-1/2^c where c is the number of congested edges.

3.2. The Basic Multi-path Algorithm

We propose two routing mechanisms. The first, presented here,is a basic mechanism that is primarily based on multiplicative decrease and additive increase

In the first routing method, for each commodity c, let P(c) be the set of paths being used. If any of the paths Pi is congested, the flow on the path is reduced using the multiplier (1-bi). Next, the shortest path is found to push additional flow requirements if they exist. An additive flow of value ai, which is chosen to be a constant independent of i, is pushed on that path.

```
 At the end of each time interval T
        FOR each  commodity c:
        Calculate commodity flow
                FOR each flow path, i, of commodity c
                        Calculate bi
                        IF {demand is met and bi = 0}
                                No change in path flow
                        ELSE
                                Apply coefficient bi to
                                        decrease path flow
                        ENDIF
                ENDFOR
                IF {demand not met}
                        Find shortest path for commodity c
                        IF{shortest Path is new}
                                Add new path to list
                        ENDIF
                        Increment shortest path flow by a
                ENDIF
        ENDFOR
```

If the number of paths used is excessive then no new paths need be generated.

3.3. The Multi-path Algorithm with Fairness

In order to ensure that different source-sink pairs are treated fairly, the coefficient bi for path Pi is chosen as PBQ - PBF with two components: a congestion component PBQ and a fairness component PBF .

1. PBQ is calculated as bi before;

2. PBF is calculated using the formula PBF(s-t) = 1-Total_current_FLOW(S-t)/(Gamma* Demand(s-t))

where Gamma is the fairness parameter and DEMAND(s-t) is the demand.

In the second routing method, again, for each commodity c, let P(c) be the set of paths being used. If any of the paths is congested, the flow on the path Pi is reduced using the multiplier (1-bi). The key difference is in the computation of bi. bi is not uniform across various user requests but is dependent on the fraction of flow of that commodity that is already being serviced by the network. Having reduced congestion, if it exists, a shortest path is found to push additional flow requirements if they exist. Again an additive flow of value ai, which in our current implementation is chosen to be a constant independent of i, is pushed on that path.

```
At the end of each time interval T
FOR {each  commodity}
        Calculate commodity c flow
        FOR {each path i}
                Calculate PBQ and PBF
                IF {demand is met}
                        IF {NO Congestion}
                                 No change in path flow
                        ELSE
                                bi = max (0, PBQ-PBF)
                                flow on path i,
                                xi(t) = (1-bi)*xi(t-T))
                        ENDIF
                ELSE
                        IF {NO Congestion}
                                bi = -PBF
                        ELSE
                                bi = max (0, PBQ-PBF)
                        ENDIF
                        xi(t) = a + (1-bi)*xi(t-T))
                ENDIF
        ENDFOR
        Recalculate Commodity flow
        IF {flow change in any path AND demand not met}
                Find shortest path
                IF {shortest path is new}
                         Add new path to list
                ENDIF
                Increment shortest path flow by a
        ENDIF
ENDFOR
```

If the number of paths become excessive then they can be curtailed. At that stage no additional flow is pushed until congestion is relieved.

4. Experiments

We implemented [1] the two algorithms using the NS-3 simulation environment. We modeled the network topology on the network of a large service provider, with link capacities proportional to capacities in the actual physical network. In our implementation we used, for routing, a combination of link-state routing protocol and source routing. For the link-state part we augmented the NS-3 implementation of the OSPF routing protocol by adding link queue occupancy to the data exchanged by nodes in Link State Advertisement (LSA) messages, a minimal increase in LSA data. That allows for more sophisticated monitoring of network status: if the queue occupancy for one of more links of a path exceeds a given threshold we conclude that the path is experiencing congestion and that the multiplicative decrease has to be applied to adjust the allocation of flow to the paths. The additive increase is applied at each iteration, if demand is not met, to augment the sending rate. Details of congestion measurement are as follows: In a node-to-node connection (data link) both the source (originating) node and the sink (destination) node have a PointToPointNetDevice. The PointToPointNetDevice at the originating node is associated with a queue function of type DropTailQueue. The queue function stores the number of packets in the queue (waiting to be sent to the destination node). A queueOccupancy parameter is calculated: numPacketsInQueue/ queueMaxSize and compared to a queueThreshold parameter. If queueOccupancy> queueThreshold the path that uses the data link is flagged as congested. queueThreshold is an input parameter declared at the start of the simulation. At the same time path delay is also calculated. The source node uses OSPF to find the shortest path to the destination and, based on available network data, builds a source-routing vector that is inserted in the packet and used by intermediate nodes to route the packet to the destination. To implement the source-routing function we augmented the NS-3 Nix-Vector protocol that builds the source-routing vector from the list of nodes to be traversed, list that is obtained from OSPF. The main process is iterative as we refresh LSA at a fixed interval: for our simulations we experimented with updating LSAs every 50 ms and 500 ms.

Our results for the throughput improvement are presented below and compared with throughput results for the standard TCP and TCP using ECMP. We show the average throughput over multiple runs.

No of Commodities	TCP	TCPw ECMP	DMP	DMP/Fairness
5	4008	3330	7949	4139
10	3275	2966	6017	5612
15	2849	2715	5373	5090
20	2912	2582	4633	3703

5. Conclusion

In conclusion we found that our algorithm with fairness provides throughput improvement over both regular TCP and TCP with ECMP. In addition, its ability to discover additional path dynamically eliminates the need to set a preselected set of paths, allowing the spreading of the traffic load amongst a wider but still reasonable set of paths. Further results may be found at www.cs.iit.edu/˜kapoor/papers/reducerate.pdf .

6. References

6.1. References

[AP13] Amer, P. and F. Pang, "Non-renegable selective acknowledgments (nr-sacks) for mptcp.", Proceedings of the 2013 27th International Conference on Advanced Information Networking and Applications Workshops , 2013.

[CRS99] Cidon, I., Rom, R., and Y. Shavitt, "Analysis of multi-path routing", IEEE/ACM Trans on Networking pages 885-896, 1999.

[DSAK11] Devetak, F., Shin, J., Anjali, T., and S. Kapoor, "Minimizing path delay in multipath networks", IEEE ICC, 2011.

[GWH11] Greenhalgh, A., Wischik, D., Handley, M., and C. Raiciu, "Design, implementation and evaluation of congestion control for multipath tcp.", 8th USENIX conference on Networked systems design and implementation , 2011.

[M05] Internet Engineering Task Force, "Coupled congestion control for multipath transport protocols", RFC 6356 , 2011.

[M06] Internet Engineering Task Force, "Multipath tcp (mptcp) application interface considerations", RFC 6897 , 2013.

[M07] Internet Engineering Task Force, "Tcp extensions for multipath operation with multiple addresses", RFC 6824 , 2013.

[M08] Internet Engineering Task Force, "Architectural guidelines for multipath tcp development", RFC 6182 , 2011.

[M75] Maxemchuk, N., "Dispersity Routing", IEEE ICC, 1975.

[PU12] Popovic, M., Upadhyay, U., Le Boudec, J., Khalili, R., and N. Gast, "Mptcp is not pareto-optimal: performance issues and a possible solution", Proceedings of the 8th international conference on Emerging networking experiments and technologies , 2012.

[RG10] Barre, S., Greenhalgh, A., Wischik, D., Handley, M., Raiciu, C., and C. Pluntke, "Data center networking with multipath tcp.", 9th ACM SIGCOMM , 2010.

[RH10] Raiciu, C., Handley, M., Barre, S., and O. Bonaventure, "Experimenting with multipath tcp.", Proceedings of the ACM SIGCOMM 2010 conference , 2010.

[SKDA13] Devetak, F., Anjali, T., Shin, J., and S. Kapoor, "Concurrent multipath routing over bounded paths: Minimizing delay variance", Globecom 2013 , 2013.

[ST92] Suzuki, H. and F. Tobagi, "Fast bandwidth reservation with multiline and multipath routing in atm networks", Proceedings of IEEE Infocom pages 2233-2240, 1992.

6.2. URIs

[1] http:www.cs.iit.edu/˜kapoor/papers/reducerate.pdf

Authors' Addresses

Fabrizio Devetak
Illinois Institute of Technology
10W 31 Street
Stuart Building
Chicago, IL 60565
US

Sanjiv Kapoor
Illinois Institute of Technology
10W 31 Street
Stuart Building
Chicago, IL 60565
US

Phone: +1 312 567 5329
Email: kapoor@iit.edu
URI: http:www.cs.iit.edu/˜kapoor