Routing Optimization for Ensemble Routing

Wenfei Wu, Yoshio Turner, Michael Schlansker

HP Laboratories
HPL-2011-125

Keyword(s):

Routing VLAN placement; Linear programming traffic splitting

Abstract:

The Ensemble Routing architecture (presented at ANCS 2010) implements multipath routing for data center networks. Rather than managing individual flows, ensemble routing manages flows in groups or ensembles to provide scalable responsive management using simple hardware. Ensemble Routing combines: routing VLANs that define a set of diverse paths through complex networks, and load balancing algorithms to split traffic among those VLANS and optimize traffic flow. This extended abstract describes improved algorithms for the formation of routing VLANs and traffic load-balancing for Ensemble routing. The VLAN formation algorithms are improved by incorporating traffic flow estimates into the VLAN formation heuristics. The previous load balancing algorithm using a greedy heuristic is replaced by linear programming that determines optimal traffic splitting among VLANs. Simulations show that these mechanisms significantly enhance performance.

External Posting Date: August 21, 2011 [Fulltext] Approved for External Publication
Internal Posting Date: August 21, 2011 [Fulltext]

Routing Optimization for Ensemble Routing

Wenfei Wu[1]
University of Wisconsin-Madison
wenfeiwu@cs.wisc.edu

Yoshio Turner
HP labs
yoshio_turner@hp.com

Mike Schlansker
HP labs
mike_schlansker@hp.com

ABSTRACT
The Ensemble Routing[1] architecture (presented at ANCS 2010) implements multipath routing for data center networks. Rather than managing individual flows, ensemble routing manages flows in groups or ensembles to provide scalable responsive management using simple hardware. Ensemble Routing combines: routing VLANs that define a set of diverse paths through complex networks, and load balancing algorithms to split traffic among those VLANS and optimize traffic flow. This extended abstract describes improved algorithms for the formation of routing VLANs and traffic load-balancing for Ensemble routing. The VLAN formation algorithms are improved by incorporating traffic flow estimates into the VLAN formation heuristics. The previous load balancing algorithm using a greedy heuristic is replaced by linear programming that determines optimal traffic splitting among VLANs. Simulations show that these mechanisms significantly enhance performance.

Categories and Subject Descriptors
C.2.1 Network Architecture and Design

General Terms
Algorithms, Management, Performance, Design

Keywords
Routing VLAN placement, Linear programming traffic splitting.

1. INTRODUCTION
Data center networks are increasingly growing to very large scale. Multiple paths are needed through these networks to maximize bisection bandwidth and guarantee efficient all-to-all traffic flow. A number of previous network architectures have been developed for multipath routing. For static or random routing[2] protocols such as Equal Cost Multiple Path (ECMP), potential conflicts may cause congestion on certain link. In dynamic routing approaches such as Hedera[3], maintaining per flow state for millions of flows leads to slow response to changing traffic.

Ensemble routing networks are constructed using enhanced access switches at the edge of the network that load balance ingress traffic, and core switches in the interior of the network that carry standard Ethernet traffic. Routing VLANs are defined so that each VLAN reaches all access switches. To achieve scalability, all flows are hashed into several routing classes. Access switches forward all packets in a routing class to one of the VLANs. The access switches also measure the traffic on each VLAN and routing class. A logically central controller periodically reads the measurements and determines which routing VLAN to use for each routing class in the next period, re-programming the routing class to VLAN mapping at each access switch.

In the original Ensemble Routing work[1], routing VLANs were constructed by placing seeds in the network and growing each seed into a tree until all access switches are reached by each tree. The VLAN growing algorithm favors assigning links that are used by fewer previously placed VLANs. For load balancing, the routing classes are considered one by one and assigned to VLANs using a greedy heuristic that favors VLANs using network links with the least traffic.

We significantly improve these two policies. For VLAN placement, our improved approach better takes into consideration the expected traffic inside each VLAN when placing VLANs. For load balancing, we formalize the problem as a linear programming problem, and use an LP tool to get the optimal solution. Simulation results demonstrate that these new approaches lead to lower load on network links.

2. VLAN Placement

2.1 General Topologies
The original routing VLAN placement algorithm for general topologies places several seeds in the network, and grows each seed into a tree until all access switches are reached. A VLAN counter for each link records how many VLANs use this link. VLANs are grown breadth-first along shortest hop paths from the seed to each access switch; if two links reach the same node, the link with smaller VLAN count is chosen. After a VLAN is placed, the used links will increase the VLAN counter by 1.

Our improved heuristic still places seeds in the network and uses minimum hop count to grow each VLAN (Dijkstra). The difference is that when the new algorithm encounters the choice of two paths that reach the same node, it prefers the path with less traffic instead of the lowest VLAN count.

The traffic in each VLAN is estimated as follows. We consider two routing policies, symmetric and asymmetric. Symmetric means that each access switch uses the same routing class-VLAN mapping. Asymmetric allows each access switch to use a unique routing class-VLAN mapping. For placement, we further assume uniform all-to-all traffic among all servers. Then we derive for symmetric/asymmetric the resulting traffic that will appear on each routing VLAN. That is, for symmetric, each VLAN is assumed to carry all-to-all traffic. For asymmetric, all traffic on a VLAN is assumed to originate from the seed switch with uniform destination distribution to servers not connected to the seed switch (shortest path).

We replace the VLAN counter for each link with a flow counter. After a VLAN is built, each used link adds the incremental flow count to the link flow counter. The algorithm also maintains a flow counter for each node. This flow counter records the maximum link load on the path from the VLAN seed to that node. When the algorithm can choose two paths to the same node, it uses the node flow counter to avoid the need to traverse the two entire paths to find the least loaded path.

2.2 HyperX Topology
For the HyperX topology, we propose an optimally balanced VLAN placement. The HyperX is a generalized hypercube. To grow VLANs, multiple seeds are placed at each switch. The VLANs grown from each seed at a particular switch use shortest

[1] This work is performed when Wenfei is an intern in HP labs.

hop paths, and each of these VLANs traverses the dimensions in a different order, such that all dimensions and links are uniformly covered by the resulting trees.

```
VLAN Placement
for each seed s_i
   V_i=BuildVLAN (s_i)
   AddFlowCountOnEachLink (V_i)
BuildVLAN( nodeIndex s )  //rooted at s
bool reached[n]={false};
double distance[n]={INFINITE};
int previous[n]={NULL};
int flowCount[n]={0};
//initialization
reached[s]=true; distance[s]=0;
//Dijkstra
while( exist unreached access switch)
   k=Find nearest unreached node to root
   reached[k]=true;
   UpdateNeighbor (k)
UpdateNeighbor ( nodeIndex k )
for each node i in node k's neighbor set
  int newPathFC=min(flowCount[k],
                    FlowCount(link(k,i)));
  if (distance[k] + 1 < distance[i])
    previous[i] = k;
    distance[i] = distance[k] + 1;
    flowCount[i]= newPathFC
  else if (distance[k]+1==distance[i])
    if(newPathFC<flowCount[i])
      flowCount[i]=newPathFC;
      previous[i]=k
```

Consider, for example, a 3-dimension HyperX (with dimension X, Y, Z). For each node, we could build 6 VLANs by extending the three dimensions in sequence XYZ, XZY, YXZ, YZX, ZXY, ZYX; so that all dimensions are used uniformly and all shortest paths are utilized. We can also build only 3 VLANs, like XYZ, YZX, ZXY; here each dimension and physical link is still used uniformly, but not all shortest paths are supported. For HyperX, this *extend-dimension algorithm* gives the optimum balance solution.

3. Traffic Splitting

Once the VLAN placement is done, the central controller will split the traffic among all VLANs to balance link load for uniform or non-uniform traffic. The controller reads the measured traffic matrix (not necessarily uniform traffic) from each access switch, uses linear programming to get the best traffic splitting for each VLAN, and then sends the new routing class-VLAN mapping to access switches. We plan to explore the use of this linear programming for both online and offline load-balancing.

Access switches are indexed from *0* to *n-1* and denoted by $s_0, \ldots, s_{n-1}$; links are indexed from *0* to *m-1*; an order pair $<s_i, s_j>$ is indexed by $i \times n+j$, denoted by $p_{i \times n+j}$. Then traffic matrix can be denoted by a vector $T=(t_0, t_1, \ldots, t_{n \times n-1})$, where t_i is the traffic between pair p_i. The fraction of traffic assigned to the k VLANs is denoted by $f_0, f_1 \ldots f_{k-1}$, where $\Sigma_{(i)} f_i=1$.

For each VLAN l, we use a matrix B_l to describe its path information. The columns denote links and the rows denote pairs. For each pair p_i, if its path includes the j-th link, we set $b_{ij}=1$; otherwise $b_{ij}=0$.

The traffic in VLAN i is given by $T \times f_i$. So $T \times f_i \times B_i$ is the traffic on each link introduced by VLAN i. We use $a_{ij} f_i$ to denote traffic on link j in VLAN i. So the total traffic on a link j is $l_j=\Sigma_{(i)} a_{ij} f_i$

$$T \times f_i \times B_i = (t_0 \quad \cdots \quad t_{n \times n-1}) f_i \begin{pmatrix} b_{00} & \cdots & b_{0,m-1} \\ \cdots & \cdots & \cdots \\ b_{n \times n-1,0} & \cdots & b_{n \times n-1,m-1} \end{pmatrix} = (a_{i0} \quad \cdots \quad a_{i,m-1}) f_i$$

Our goal is to minimize the load on the most-loaded link, i.e. to minimize $u=max(l_0, l_1, \ldots, l_{m-1})$. This min-max problem is equivalent to a linear programming problem:

$$\min u, \text{ s.t.}$$

$$L=AF,\ L = (l_0, l_1, \ldots, l_{m-1}),\ F = (f_0, f_1, \ldots, f_{k-1}),$$

$$u \geq l_0, u \geq l_1, \ldots, u \geq l_{m-1}$$

After the traffic portion to each VLAN is known, the central controller will map a number of routing class to that VLAN. The portion of routing classes to a certain VLAN is approximately equal to the portion given by the LP solver.

4. Simulation

4.1 VLAN placement

We compare link loads using the original VLAN placement algorithm (link-VLAN heuristic), our algorithm (link-load heuristic) and extend-dimension algorithm(Ext-Dim). Following table shows min,max,average link loads under uniform traffic (matrix of all 1's) with equal traffic splitting across VLANs.

HyperX	Symmetric			Asymmetric		
5x5x5	**Max**	**Min**	**Aver**	**Max**	**Min**	**Aver**
LVLANHeu	74.2	24.3	46.8	44.3	10.3	25
LLoadHeu	55.4	19.3	46.8	31.3	18.3	25
Ext-Dim	45.9	45.9	45.9	25	25	25
10x10	**Max**	**Min**	**Aver**	**Max**	**Min**	**Aver**
LVLANHeu	25.0	10.4	19.1	16.5	3.5	10
LLoadHeu	23.7	6.4	19.1	10	10	10
Ext-Dim	18.9	18.9	18.9	10	10	10

The results show that our new heuristic reduces the max link load by 10%. In 2-D HyperX, our algorithm matches the optimum Ext-Dim solution for asymmetric routing.

4.2 Traffic Split

We use HyperX10x10, symmetric flow pattern, and uniform traffic (matrix all 1's). VLANs are derived using our new heuristic VLAN placement. The traffic splitting algorithm leads to the following link usage:

HyperX	Symmetric		
5x5x5	**Max**	**Min**	**Aver**
Equ-Spt	55.4	19.3	46.8
LP	48.7	19.3	46.8
10x10	**Max**	**Min**	**Aver**
Equ-Spt	23.7	6.4	19.1
LP	20.54	6.3	19.1

The results show that our LP solver reduces the max load by at least 10% compared to naïve equal splitting across all VLANs.

5. Conclusion and Next Steps

Our improved VLAN placement and traffic splitting algorithms appear promising. We plan to extend this work: examining non-uniform topologies (e.g. after link/switch failure) and traffic patterns, and adding the algorithms to the Open vSwitch testbed implementation of Ensemble Routing.

6. References

[1] Schlansker, M. et al. Ensemble Routing For Datacenter Networks. ANCS 2010.

[2] Albert Greenberg, et al. VL2: A Scalable and Flexible Data Center Network. SIGCOMM 2009

[3] Mohammad Al-Fares, et al. Hedera: Dynamic Flow Scheduling for Data Center Networks. NSDI 2010