LETTER

Multipath Probing and Grouping in Multihomed Networks

Jianxin LIAO[†], *Member*, Jingyu WANG[†a)], Tonghong LI[††], *and* Xiaomin ZHU[†], *Nonmembers*

SUMMARY We propose a novel probing scheme capable of discovering shared bottlenecks among multiple paths between two multihomed hosts simultaneously, without any specific help from the network routers, and a subsequent grouping approach for partitioning these paths into groups. Simulation results show that the probing and grouping have an excellent performance under different network conditions.

key words: *multihomed, network probing, path diversity, shared bottleneck*

1. Introduction

In multihomed networks, most researches about concurrent multipath transfer (CMT) and retransmission/backup path selection have the assumption that the multiple paths are independent [1], [2], but this assumption is rarely warranted. In a real multihomed network, multiple paths are likely to overlap each other and even share the same bottleneck, which can weaken path diversity drastically. So it is necessary to diminish this assumption and take into account the correlation between topologically joint paths [3], which is partly motivated by the observation that packets sent over dependent paths likely suffer from bottleneck or congestion simultaneously. *Thus, it is reasonable to differentiate and group the paths according to their correlations, which can be obtained through external end-to-end measurements, for the multipath selection and control in the design of CMT schemes.*

An end-to-end path is a virtual link directly connecting two IP addresses which come from source and destination host respectively. We let P_{ij} simply denote any one path connecting source address IP_s^i and destination addresses IP_d^j. Most researches on network measurement attempt to learn about the characteristics of a single path, but do not address directly the problem of identifying the correlations between multiple paths. Therefore it is crucial to identify bottlenecks in the large-scale network so as to evaluate the path correlation. Path bottleneck points are the most critical to impact the performance of the entire path, and their relative locations directly affect the degree of path correlation. Rubenstein et al. [4] attempted to detect whether two flows share bottleneck through end-to-end measurement. However, their goal is to exploit the relation between the flows rather than their forward paths.

The path correlation is defined as the correlation coefficient, based on Pearson's correlation function, between two sample sequences collected from P_{ij} and P_{xy}. In our scheme, we adopt the delay incurred by each probing packet to compute the correlation coefficient. We can simply compare the levels of correlation to statistically identify shared bottlenecks. The *comparison tests* learned from [4] is defined as follows: (1) Compute the *cross-measure*, M_x, between pairs of packets on two different paths, spaced apart by time $t{>}0$. (2) Compute the *auto-measure*, M_a, between packets on the same path, spaced apart by time $T{>}t$. (3) If $M_x{>}M_a$, then two paths share a common bottleneck, otherwise they do not. The intuition behind this test is that if the spacing between packets on different paths at the shared bottleneck is smaller than the spacing between packets on the same path, then the cross correlation coefficient should exceed the auto correlation coefficient.

2. Multipath Probing and Grouping

2.1 Path Topology Logicalization

To indicate the relations of any two paths, we can model the path logical topology only including source-destination hosts and crucial Intermediate points (branching point (B) and joining point (J)). We assume routing strategies use general single-path routing, which implies that there is a unique path from each source address to each destination address. These assumptions are realistic, the same as in [5], which are consistent with the routing behavior in the internet: the next hop taken by a packet is determined by a routing table lookup on the destination address. In this case, any two paths can form one of four logical topologies shown in Fig. 1. A shared bottleneck exists if congestion occurs along the top portion of the 1-by-2 component; along the bottom portion of the 2-by-1 component; or along the center portion of the 2-by-2 shared component. In the following section, we will discuss our probing process and grouping process. The notations used in our discussion are listed in Table 1.

2.2 Probing Structure

In [4], their work is only suitable to the Inverted-Y and Y topology and does not easily generalize to more than two

Manuscript received August 6, 2010.
Manuscript revised October 26, 2010.

[†] The authors are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, 100876 China.

[††] The author is with Technical University of Madrid, Madrid, 28660 Spain.

a) E-mail: wangjingyu@bupt.edu.cn

DOI: 10.1587/transinf.E94.D.710

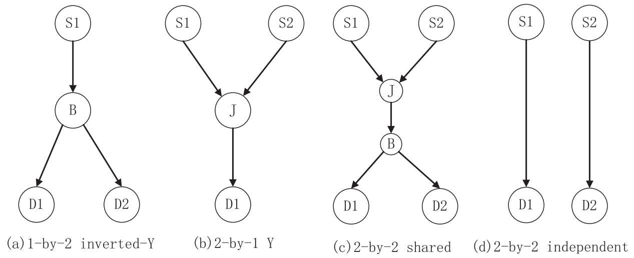

Fig. 1 Four logical topologies of any two paths.

Table 1 Notations definition.

T	spacing between packets on the same path
t	spacing between pairs of packets on two different paths
DS_n	destination address set with N addresses
λ	inter-packet spacing within a block
μ	inter-block spacing
Δ	spacing between two adjacent N-packets-pairs
x(u)[y(v)]	timestamp of sample u[v] from $P_{ij}[P_{xy}]$

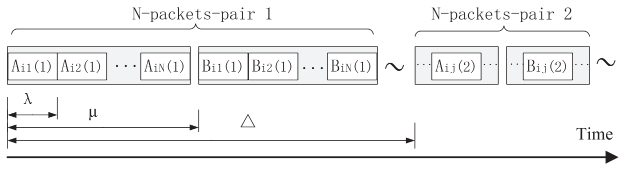

Fig. 2 N-packets-pair probe sequence from single source i.

flows in which the probing load is too heavy, so it is essential to design a general, light and fast probing scheme to infer the correlation of multiple paths simultaneously. The key problem is how to set the packet spacing and sending time of probing sequences such that multiple *comparison tests* can be performed in parallel. One of the essential techniques in our constructions is the extension of "*packet-pair*" [6].

Definition 1 An *N-packets-pair* probe sequence $S(DS_n; \lambda; \mu; \Delta)$ is a sequence of block pairs with each block including N packets of the same size as shown in Fig. 2. The consecutive N packets are transmitted respectively to different destinations in destination address set DS_n (there are N addresses), in much the same way as a packet is multicasted to multiple receivers. The inter-packet spacing within a block is λ time units, the inter-block spacing is at most μ time units and two adjacent N-packets-pairs is spaced by at least Δ time units.

The intuition behind the structure is to provide a baseline for the delay correlation over each path of the same source. The key insight is that because of their temporal proximity, as the quantity of addresses in a destination host (N) is limited to a small number, we expect packets within an *N-packets-pair* to have a high probability of experiencing a shared fate on the shared links. This well-designed structure ensures that if both paths from the same source share a bottleneck, then the spacing of packet-pair on the same path (T) at this bottleneck is larger than the spacing between packets on different paths (t). Thus, the precondition of *comparison tests* can be satisfied. The M_a of P_{ij} can be computed by the delay of A_{ij}(k) and B_{ij}(k), and the M_x is slightly complicated, which will be discussed in the next section. The values of μ and Δ in the above definitions are chosen empirically in order that the intra-pair and inter-pair packets highly experience correlated and dependent packet delays, respectively.

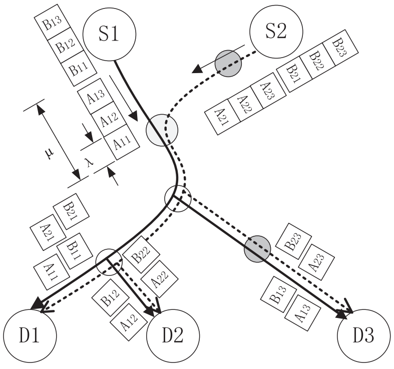

Fig. 3 Example of multi-source probing in one period for the source with two addresses and the destination with three.

2.3 Probing Process

For a probing session initiated by a single source address, the set of destination addresses are treated as probing terminal points of a probe tree, in which each two branches forms an Inverted-Y topology, and the above *N-packets-pair* sequence is used. For M source addresses, the similar probing sequence is sent from every source in parallel and simultaneously as Fig. 3. As each source constructs one probe tree, there are M trees altogether. The branches from different trees converge to a certain junction and forms an Y topology.

To compute this *cross-measure*, we should find a sequence of matched packets from two paths, which should arrive that bottleneck at roughly the same time if both paths have a shared bottleneck. A key step in this process is synchronizing and matching up the sample sets on each of the paths. For any two paths from single source corresponding to the Inverted-Y topology, we notice that the delay from each source to any shared link (shared bottleneck if it exists) is almost the same. Thus, the sending times can be used to find a sequence of matched packets.

For any two paths corresponding to the Y topology, the receiving times can be used to determine samples. As the packet loss is likely to occur, receiving times of dropped packets do not exist, which can be estimated by interpolating the receiving times of other packets. Let x(u) denote the timestamp (receiving time) of sample u from P_{ij}, and y(v) denote the timestamp of sample v from P_{xj}. We merge two sets of x(u) and y(v) and compute the mean spacing t, such that sample x(u) in each packet-pair is paired with a peer sample y(v) that minimizes |x(u)-y(v)| for all v. In this case, we may require to adjust probing spacing μ and Δ to ensure T>t. The obtained samples can be used for *comparison* *tests*.

For other case in which two paths have different sources and destinations corresponding to the 2-by-2 topology, packets may be congested far from the sources and destinations, so determining samples for this case is more complicated which is not discussed in [4]. We propose an indirect way by introducing *intermediary path* to circumvent this tough problem easily.

Definition 2: Consider two *interlaced paths* P_{ij} and P_{xy} without identical source address and destination address, that is $i \neq x$ and $j \neq y$. *Intermediary path* is the path that has the same source with one path and the same destination with the other, i.e. P_{iy} and P_{xj}.

Theorem 1: Let $P_{ij} \underset{SB}{\odot} P_{xy}$ denote that P_{ij} and P_{xy} have shared bottleneck SB. The two *interlaced paths* P_{ij} and P_{xy} have shared bottleneck SB if and only if both of them have the same bottleneck SB with the *intermediary path* P_{iy} (or P_{xj}). That is $P_{ij} \underset{SB}{\odot} P_{xy} \rightleftharpoons P_{ij} \underset{SB}{\odot} P_{iy} \wedge P_{xy} \underset{SB}{\odot} P_{iy}$.

Intuitively, each path can have one and only one bottleneck. If two *interlaced paths* P_{ij} and P_{xy} have shared bottleneck with the same (*intermediary path*) P_{iy} (or P_{xj}), then both of them are bound to have shared bottleneck; and vice versa.

Corollary 1: Let $P_{ij} \otimes P_{xy}$ denote that P_{ij} and P_{xy} do not have shared bottleneck. The two *interlaced paths* P_{ij} and P_{xy} do not have shared bottleneck if and only if one of them does not have shared bottleneck with the *intermediary path* P_{iy} (or P_{xj}). We have $P_{ij} \otimes P_{xy} \rightleftharpoons P_{ij} \otimes P_{iy} \vee P_{xy} \otimes P_{iy}$.

The corollary can be obtained by performing a negation operation of Theorem 1.

2.4 Grouping Process

The grouping process takes as input a set of target paths (with sufficient samples) to be grouped. We number P_{ij} as k_{th} path with $k = (i-1)N + j$. We group each path according to its order. First, the first path P_{11} is to be grouped, then the second path P_{12}, etc. To group a new path P_{xy}, we designate a *representative path* P_{ij} in every group which has an identical source or destination address with P_{xy}, i.e., $i = x$ or $j = y$.

A new path is only compared to the *representative path* of the group to determine whether it should join the group or not. This ensures that all paths that are grouped together are highly correlated. However, if there is no *representative path* in the group, the new path is not joined to that group, as it can not have shared bottleneck with any path of that group according to the Corollary 1. According to the order of path grouping, at least one of its *intermediary paths* have completed grouping process. In case there are more than one *representative path* in a group, any of these *representative paths* can be used for comparison. Finally, if the new path can not be joined to any existing group, a new group is created.

3. Evaluation and Numerical Results

3.1 Simulation Configuration

We use OPNET to imitate a more universal network including one source with three addresses, one destination with four addresses and several routers. A real traffic trace (“Star Wars” movie [7]) is used as a source of self-similar background traffic. The buffer size in the router is set to 100 packets in the generic case. Table 2 summarizes the simulation parameters. The capacity and propagation delay of each link are indicated in Fig. 4. The 12 paths produce five bottlenecks: SB1 and SB2 are congested by high cross-traffic load; SB3 and SB4 are congested by limited bandwidth; and the only unshared bottleneck between R3 and R7 is uniquely possessed by path P_{34}.

3.2 Grouping Accuracy Index

Measuring the accuracy of a grouping approach in a unified manner is challenging due to the possibility of simultaneous occurrence of various error types. We adopt the grouping Accuracy Index (AI) proposed in [8] to evaluate our path grouping scheme, which is computed as: Accuracy Index (AI) $= 1 - \frac{\sum_{i=1}^{|P_o|}(n_{fs})_i}{N} - \frac{\sum_{j=1}^{|P_c|}(s_j-1)}{N}$, where N denotes the total number of paths, P_c denotes the set of correct groups, P_o denotes the set of groups generated by the grouping scheme, $(n_{fs})_i$ denotes the number of paths erroneously included in the i^{th} group $\in P_o$, and s_j denotes the number of subgroups in P_o that are split from the j^{th} correct group $\in P_c$. The AI varies between a fraction (above 0) and 1. The closer to 1, the more accurate the result is.

Table 2 Simulation parameters.

TCP flows	12 infinite FTP flows
Cross traffic	2 flows, CBR (8 Mbps and 6 Mbps)
Background traffic	to all links (1 Mbps traces)
Queue size	250 packets
Drop policy	Drop-Tail and RED
Mean size of packet	500 bytes

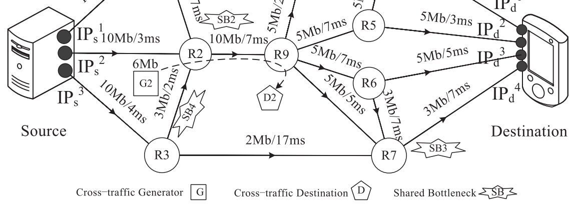

Fig. 4 In the more universal simulation configuration, there are five shared bottlenecks. Its correct grouping is $\{P_{11}, P_{21}, P_{31}\}$, $\{P_{12}, P_{13}, P_{22}, P_{23}\}$, $\{P_{14}, P_{24}\}$, $\{P_{32}, P_{33}\}$, $\{P_{34}\}$.

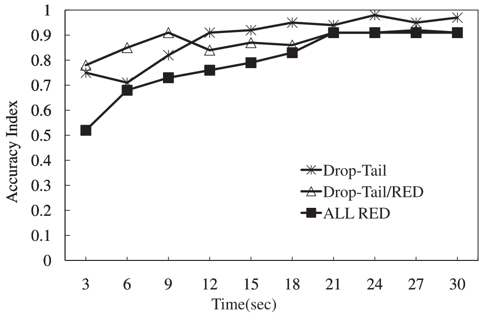

Fig. 5 AI with different packet drop policies.

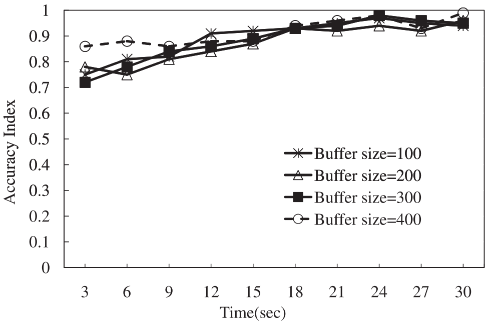

Fig. 6 AI with different router buffer sizes.

3.3 Impact of Network Conditions

In the following, we study the impact of network conditions such as the drop policy and buffer size on the accuracy of our scheme.

1) Packet Drop Policy: We use Drop-Tail policy and RED policy here. Figure 5 shows the resulting AI with three cases: (1) using the Drop-Tail queue for all queues; (2) using both Drop-Tail queues and RED queues; (3) using only RED queues. Results show that in case 3 the accuracy is reduced. The reason is that random packet drop interferes with the sample process and introduces noise to the correlation computation. This is also consistent with the results presented in [4].

2) Buffer Size: We vary buffer sizes from 100 to 400 packets in every router where we use Drop-Tail queues. Detailed results for specific buffer sizes are shown in Fig. 6. Although the delay correlation is more clearly manifested in bottlenecked routers with long queues, varying buffer sizes does not result in significant performance variation in steady state. Variation is more distinct during the transient period, because the sending windows at sender are explored too excessively at that period. We believe that having routers with larger buffers can usually enhance performance.

4. Conclusion

We propose a multipath probing and subsequent grouping strategy to partition the paths into groups which can be used for multipath selection, congestion coordination, or pricing modules. The results demonstrate that the scheme under different network conditions is accurate, even with burst background traffic. Our approach is better for the stable network where the change of path bottleneck is relatively smaller.

Acknowledgments

This paper was jointly supported by: (1) National Science Fund for Distinguished Young Scholars (No. 60525110); (2) National 973 Program (No. 2007CB307100, 2007CB307103); (3) National Natural Science Foundation of China (No. 61072057, 60902051); (4) Fundamental Research Funds for the Central Universities (BUPT2009RC0505); (5) National Key Science & Technology Specific Project of China (No. 2011ZX03002-001-01, 2011ZX03002-002-01).

References

[1] J.R. Iyengar, P. Amer, and R. Stewart, “Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths,” IEEE/ACM Trans. Netw., vol.14, no.5, pp.951–964, Oct. 2006.

[2] H. Hsieh and R. Sivakumar, “A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts,” Proc. ACM Mobicom, Atlanta, USA, Sept. 2002.

[3] J. Apostolopoulos and M. Trott, “Path diversity for enhanced media streaming,” IEEE Commun. Mag., vol.42, no.8, pp.80–87, Aug. 2004.

[4] D. Rubenstein, J. Kurose, and D. Towsley, “Detecting shared congestion of flows via end-to-end measurement,” IEEE/ACM Trans. Netw., vol.10, no.3, pp.381–395, June 2002.

[5] M. Rabbat, M. Coates, and R. Nowak, “Multiple source Internet tomography,” IEEE J. Sel. Areas Commun., vol.24, no.12, pp.2221–2234, Dec. 2006.

[6] S. Keshav, “A control-theoretic approach to flow control,” Proc. ACM SIGCOMM, Zurich, Switzerland, Sept. 1991.

[7] M.W. Garrett and W. Willinger, “Analysis, modeling and generation of self-similar VBR video traffic,” Proc. ACM SIGCOMM, London, U.K., Sept. 1994.

[8] O. Younis and S. Fahmy, “FlowMate: Scalable on-line flow clustering,” IEEE/ACM Trans. Netw., vol.13, no.2, pp.288–301, April 2005.