

Accurate Relay Selection for Improved Multipath Throughput in Internet Overlays

Andrei Agapi

Vrije Universiteit, Amsterdam
The Netherlands
Email: aagapi@few.vu.nl

Thilo Kielmann

Vrije Universiteit, Amsterdam
The Netherlands
Email: kielmann@cs.vu.nl

Abstract—Distributed applications deployed over the Internet, such as in grids, clouds, or P2P networks, crucially depend on the speed of their Internet connections. A common solution is to exploit underlying network parallelism via multipath routing within an application-level overlay network. Accurately selecting appropriate relay hosts within an overlay, however, remains a challenging problem, frequently addressed by simple heuristics.

We propose a modeling-based approach to solving the relay selection problem for multipath overlay routing. Our system enables formula-based, dynamic, flexible and overlay-specific model building and prediction of end-to-end metrics such as TCP throughput on multiple paths based on any number of underlying predictor metrics. For evaluation, we use Planetlab and, as prediction data, we leverage 9 months worth of available bandwidth traces collected from the S3 project, in addition to our own measurements. Comparing with existing heuristic methods, we show that relay selection using our modeling-based approach enables consistent performance gains across the Internet.

Index Terms—application level overlays, multipath routing, multivariate regression, path disjointness, TCP throughput

I. INTRODUCTION

Making most use of their Internet connections, whether with respect to average throughput, latency or QoS predictability has always been a desirable goal for distributed applications. With nowadays networks supporting increasingly high data rates, many application domains have emerged, enabling increasingly data-intensive, QoS-sensitive applications. While network link capacities have been known to grow at a pace superior to that of Moore’s law, and advances in router capabilities enable ever higher performance and reliability, the Internet as a whole still remains to date highly heterogenous and best-effort based. Moreover, it appears to exhibit high market elasticity, meaning that demand for bandwidth quickly soars to match and exceed provided capacities.

Novel applications such as production-scale cloud computing, cloud-based storage and virtualization, P2P computing or enterprise grids have lately emerged. These add to the strain already put on the network by other well-known bandwidth-hungry applications such as file sharing, VoIP and lately, HD video streaming or telepresence. From P2P systems to CDN’s, from enterprise grids and clouds to routing overlays [1] [12], many applications attempt to provide QoS via techniques such as downloading from multiple peers or replicas, collaborative multicasting or relaying. All these techniques essentially employ multipath streaming over the Internet. Oftentimes,

such systems act adaptively, by dynamically choosing best-performing paths. For instance, a BitTorrent client periodically tries new random peers to exchange content with, to replace worst-performing ones. While simple and adaptive, such “greedy” strategies may be sub-optimal w.r.t. achieved throughput, since a limited number of peers can be tried during a download. Other times, overlays use heuristics e.g. “widest path” as in Bandwidth Aware Routing [14] or compound heuristics as in iPlane [16], latency-wise “closest nodes”, as in many CDN’s or DHT’s, or random relay picking [7].

However, picking a good relay, datacenter cluster or peer to download from is a complex problem when considering the Internet, since many variables are involved. Underlying native topology, link capacities, cross traffic, RTT’s, loss rates, but also end host system capabilities [10][11] play an important role in path performance.

Our software can be used to dynamically build models based on a variety of measured prediction variables, relevant to each specific overlay, and dynamically predict best relays to use for various transfers. We quantify and model the influence of various factors on TCP throughput (TCPT) achieved when streaming on multiple paths and propose a generic, extensible, statistical modeling-based approach, which works by unveiling which underlying network properties mostly affect multipath TCPT and quantifying their interactions. While our approach can handle arbitrary numbers of parameters to be explored, the current paper focuses on a limited subset, for the sake of brevity. We demonstrate our approach using a number of commonly used prediction metrics. We focus on multipath streaming within the data relaying setup, i.e. relays are used to forward traffic between source and destination. This is generally the approach taken by routing overlays, such as RON [1]. While many systems rely on “disjoint paths” or advise their use as “best practice”, they seldomly rely on hard evidence regarding interactions between disjointness and other path metrics, and indeed we find that disjointness alone is not enough. By building interpretable models, our approach can also provide insight into metric interactions and generate or quantify heuristics. As an initial step, we limit our discussion to 1-relay alternate paths, used in conjunction with the direct source-destination path for multipath streaming. While further TCPT improvement may be possible by using more relays, or other multipath strategies (e.g. gathering from multiple replicas

or multicast) we found the 1-relay model instrumental as an initial step to quantify the effects of path disjointness on TCPT. The 1-relay setup has been used (e.g. [1] [7]), in general for improving path reliability, or, as in Skype, to detour around firewall and NAT filtering. We use paths in parallel to improve throughput, as per dispersity routing [18][2].

The paper is structured as follows. Section 2 outlines issues of TCP throughput modeling and of existing approaches to relay selection. Section 3 describes our approach, consisting of the PathInfo database and the msocket multipath streaming software, as well as the statistic method we use for model building. Internet measurements and evaluation results are presented in Section 4. Conclusions are drawn in Section 5.

II. BACKGROUND AND RELATED WORK

Extensive research has found that TCP throughput (TCPT) is affected by a multitude of factors and consequently quite hard to predict, even on a single path and in absence of parallel connections. While in principle TCP should fill the available bandwidth (a.b.) of a path if properly configured, this was very often shown not to be the case. Authors of pathload [9] observed TCPT to be 20%-30% higher than estimated a.b., due to TCP grabbing bandwidth from competing cross traffic (XT), whereas nettimer [13] experienced the opposite, with TCPT consistently lower than estimated a.b. on a wide range of access technologies.

Reasons why a.b. or any other metric alone cannot predict TCPT are multiple: lossless versus lossy paths (loss rate becoming a bigger factor in conjunction with RTT, e.g. per the PFTK model) [8], considerable difference between RTT, a.b. and loss as observed by TCP stream vs. as probed by estimation tools [8], high Bandwidth Delay Product (BDP) paths which TCP biases against [21], actual nature of cross traffic (TCP is known to give away a.b. to competing streams, whereas UDP has no congestion avoidance), parameter tuning (buffer sizes, congestion window, TCP implementation), but also fundamental approximations of what is actually measured by average a.b. estimation. For instance, work in [10][11] introduces the Maximum Burst Size (MBS) metric, as the effective byte count that the bottleneck router can process before dropping packets; MBS depends on router queues and instantaneous XT (can be estimated via e.g. *pipechar -Q*) and many times differs from a path's average BDP. This can be due to router queues being configured too small, traffic shaping, asymmetric paths or incorrect auto-negotiation, but also XT variability. As a result, if $MBS < BDP$ and TCP max window is configured as usual $> BDP$, packet loss will occur and the window will end up wiggling around the MBS in a see-saw pattern, causing sub-optimal TCPT.

Further, using multiple TCP streams in parallel has been shown to drastically improve TCPT far beyond predictions based on a.b. [22] and a strong correlation has been found between transfer size and TCPT [24][15], which lowers the accuracy of simple History Based (HB) prediction as by NWS or [8]. Finally, end system capability in terms of CPU, PCI chipset, IO bus bandwidth, interrupt frequency, syscall time

etc. have been shown [10][11] to affect end-to-end TCPT, to the point that end system bandwidth, as opposed to the network, could soon become the prevalent bottleneck if growth trends are maintained [10]. This is especially relevant for overlays, and makes TCPT highly overlay-dependent, thinking of systems like PlanetLab (PL), where virtualization causes very high system utilization.

In [7], Gummadi et al. classify network reliability-enhancing techniques as either CDN-style replication, proper multi-homing (as per buying multiple links from various ISPs and using BGP failover to switch away from bad paths) or RON-style [1] overlay routing. Authors there note disadvantages to all these approaches, e.g. the fact that CDN replication is costly, multi-homing can be affected by long BGP fail-over times, or that path monitoring does not scale to large overlays. We contribute towards bridging the latter gap, namely achieving scalable performance prediction by leveraging historical distributions of the desired result property (e.g. TCPT) for only a subset of sampled overlay paths, much like landmark nodes in synthetic coordinate systems. This is useful because, in large overlays, not all nodes will have previously communicated so TCPT traces may be unavailable for most node pairs, or inaccurate due to correlations with transfer size [15]. We employ statistical modeling to predict good relays for the rest, based on less complex prediction metrics, more likely to be available for all nodes, such as RTT estimates, end-to-end a.b. distributions or router-level topology, for which extensive research efforts have been made to scale measurements or approximate results (e.g. S3 [23] or Vivaldi [3]). Our work relates to various Internet tomography or scalable monitoring systems that could be used for prediction, such as S3 [23], Network Weather Service, iPlane [16] or iNano [17], but, compared to all, we provide per-overlay dynamic modeling of TCPT from an arbitrary number of prediction metrics, thus we fill a much needed gap between topology inference or path property prediction ("rich" or not), to scalable prediction of complex parameters useful for overlays, such as multipath, multistream TCPT. In fact, we show by evaluation how our modeling approach outperforms direct prediction achieved by heuristics on multiple path attributes, quite similar to those suggested in iPlane [16] or BARON [14].

Overlay routing has been extensively studied with routing overlay networks, such as [1] [12]. Other work that previously proposed multipath routing or studied path diversity includes seminal work on dispersity routing [18][2], PDF [20], focused on fault tolerance, or p2p-related work like [4], mainly focused on AS-level paths and employing routing heuristics. Within the TCPT prediction classification in [8], we do Formula Based (FB), rather than History Based (HB) prediction, we focus on multipaths, and propose online, dynamic FB modeling as opposed to one-size-fits-all formulas.

III. MEASUREMENT AND MODELING APPROACH

A. The PathInfo Database

We implement a per-path information database named PathInfo, indexed on (src,dst) pairs, which unifies vari-

ous e2e measured information and enables queries, modeling and prediction of some variables based on others. It currently includes information about router-level topology, historical distributions of: a.b. (sampled by periodic active probes), end-to-end RTTs, per-router RTTs (derived from `traceroute`). The database is currently centralized, relational and SQL compliant, supporting queries such as: `"SELECT src,dst,a_b_dist from PathInfo ORDER BY dist_max(a_b_dist)"`. As future work, we plan to distribute it and have it support laxer data models for increased scalability. In the example query, `a_b_dist` represents the distribution of historically observed a.b. on the path and the `dist_max` call returns its max value. Historically observed a.b. for that overlay, e.g. as published by S3[23] can be fed into the DB. In the current paper, we used about 9 months worth of measurements we have been downloading from S3. These are e-2-e measurements published every 4 hours and for getting them scalably in the first place, S3 takes periodic active probes using tools such as `pathchirp` or `spruce` and applies inference techniques, such as aggregating results for nearby nodes[23]. To use PathInfo, an overlay needs a similar monitoring infrastructure to run beneath. As a first step, we use historical distributions to derive likely expected values for a.b. on links, taking into account cross-traffic. Various strategies are possible, but in our approach, we use a certain (dynamically tuned) percentile of the distribution as an expected value. For this, we characterize each distribution by a Complementary Cumulative Distribution Function (CCDF), which gives us the probability $P(X)$ that a.b. will be greater than a certain value X , then, during the model building phase explained later, we explore various values of p and dynamically pick best performing models based on several statistical indicators. In the context of previously observed self-similar nature of cross traffic over several timescales (e.g. day-night timeframes) and of other factors, such as native layer path flopping, the CCDF approach is obviously a statistic approximation of a.b. We extend PathInfo with a second DB, `1RelayPathInfo`, which is a self-join of PathInfo that can be used for relay querying in 1-relay multipath streaming, indexed on `(src,dst,relay)` to increase query performance.

Definition 1: We define the router-level disjointness of a `(src,dst,relay)` tuple as the number of routers that are present in the direct Internet path `src-dst`, but are *not* present in the alternate path `src-relay-dst`.

This would mean that, by application-layer relaying, routers in the disjoint set are not being used by the alternate path, as also illustrated in Figure 1. Higher disjointness is in principle desirable, since congestion or cross-traffic occurring at routers in the common set (R_c) will affect both the direct and alternate paths, whereas congestion anywhere in R_d (the disjoint set) will affect only the direct path. So ideally the more disjoint the path provided by a relay is, the higher the likelihood that congestion is relieved from the direct path by multipath streaming. Our DB allows for instance simple querying for the most disjoint relays, e.g. via a simple `"ORDER BY disjointness(sd_router_IPs,srd_router_IPs) LIMIT 5"`, where

`sd_router_IPs` holds IPs of the routers as returned by `traceroute` for the direct path, and `srd_router_IPs` is the respective value for the alternate path, obtained as a set union of the `src-relay` and `relay-dst` paths. Router-level disjointness as defined has known limitations related to `traceroute`, e.g. inability to detect layer 2 shared resources, problems with native layer path flopping, MPLS topologies, or router aliases attaching different IPs and DNSs to interfaces on same router. Regarding path flopping, we try to alleviate it by picking the historically most stable path for a given `(src,dst)` pair, and plan to extend our system to employ de-aliasing software such as Rocketfuel’s `ally` to resolve router aliases. However, for now we approximate router disjointness using `traceroute`-returned sets of IPs and rely on statistical modeling at higher level to achieve useful prediction.

PathInfo is extensible with other properties, e.g. BGP-based AS-level path disjointness or `pathrate`-estimated link capacities. We currently evaluated PathInfo with cca 9.500 direct Internet paths, among 100 randomly picked PL nodes. For each path, we generate all 1-relay intermediate paths available, stored in `1RelayPathInfo`. The latter has about 400.000 entries, since we focused on those `(src,dst)` pairs that had path data available for at least 70 relays to reliably evaluate relay picking algorithms (S3 traces do not contain a full mesh due to missing measurements). At this volume, the centralized PathInfo still performs well, with representative query response times still sub-second (server was a dedicated, fairly old Dual Core AMD Opteron 2.4 GhZ, 4GB RAM). For the overlay described above, PathInfo has 33 MB and `1RelayPathInfo` 1.4 GB, uncompressed. For correlating TCPT response, we use a third smaller database, with TCPT responses per path, for both direct- and multi-path, measured by `myperf`, the tool described next. We perform the heavier `myperf` measurements for about 800 paths to build model, which then is applied and predicts for the full `1RelayPathInfo`.

B. Multipath streaming implementation

We implement `myperf`, an `iperf`-lookalike allowing optimization of TCPT over multiple Internet paths. `myperf` works as illustrated in Figure 1. TCPT, even on single paths, depends on window size, MTU, congestion avoidance implementation, number of parallel threads etc. While MARS modeling as described in section III-C allows adding any number of parameters to models, in the current paper we build models based on a few commonly used ones (for both alternate paths), and separately use classical tuning values to optimize others. `myperf`, just like `iperf`, allows tweaking TCP parameters. However, since our measurement platform was PL, some parameters cannot be tweaked beyond certain values. PL doesn’t allow overwriting certain settings, e.g. max window sizes (`wmem_max` and `wmem`), congestion control is pinned to `reno` and other parameters e.g. `net.core.netdev_max_backlog` are fixed, probably to limit per-slice memory and bandwidth usage. To bypass settings, one option is using a custom TCP stack implementation. However, we choose a different approach, based on tweaking

the number of parallel TCP threads per path, which helps portability, improves the slow start phase and allows multipath optimization via dynamically load balancing threads over alternate paths. As in Fig. 1, each of the alternate paths has a number of TCP threads allocated. Payload data is divided in chunks, with a customizable chunk size. Threads all compete for network resources they use in common, such as router queues. Note that all "compete" for the shared part of the path and only part of them compete for the disjoint parts. If a certain path performs better than the other at a certain point, e.g. due to cross traffic, its threads will "steal" chunks from the others. Chunk size can be tweaked to tradeoff between load balancing (too big a size causes bad-performing threads to "lock" chunks that could otherwise be sent) and overhead (small size can cause thread synchronization and system call overhead). We found good chunk sizes to be between 32K and 64K, also depending on payload size. We tune the number of threads to be assigned to each path based on the path's Bandwidth Delay Product (BDP) and the file (payload) size. The formula we use is $N = \min((BL)/W, F/c_t/K)$. N is the number of threads, B and L are the bandwidth and latency as estimated below, W is the max window size *per thread*, F is the payload size, c_t is a load balancing parameter representing the minimum number of chunks per thread (tuned to 8) and K is the chunk size. The reason for the second term is that the BDP/W formula can produce threads in excess if the payload is too small. We estimate B and L as the 0.7 percentile of the pathchirp-measured a.b. distribution multiplied by a tuned scaling factor (0.8) and the median ping-measured latency, respectively. We found these constants by tuning on PL nodes. In general, we found that underestimating the number of threads is much worse than overestimating, and the overhead of more threads is preferable to not filling the pipe.

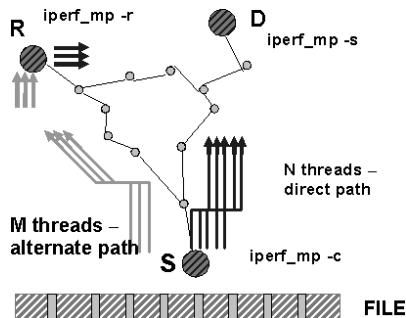


Fig. 1. Myperf: an number of threads are used on the alternate paths and "compete" for file chunks.

Unlike `iperf`, `myperf` has three components: a "server" (D), a "client" (S) and a relay (R), forwarding traffic from S to D. In the current implementation, the number of threads used on the S-R and R-D paths is equal and computed with bandwidth $B_{srd} = \min(B_{sr}, B_{rd})$ and latency $L_{srd} = L_{sr} + L_{rd}$. This is because a relay shouldn't receive data faster than it can forward it, because that will affect the bandwidth of the competing threads on the path portion shared by SD and SR. If RD cannot sustain the rate of SR, we are better off

leaving the threads on SD to perform the transfer. Therefore, the relay actually limits its download rate to its upload rate. If RD is faster than SR, it does not matter anyway, since R cannot forward traffic that has not reached it yet.

C. The statistical model

To quantify the influence of various Internet properties on TCPT, as well as how they interact, we use Multivariate Adaptive Regression Splines (MARS) [6] [5] to build a statistical model of multipath TCPT. Such a model can be dynamically built by individual Internet overlays installing a PathInfo database. An advantage of this approach is that the model will better fit the particular overlay topology and thus better predict TCPT on that overlay, by capturing typical cross-traffic, link capacities, topology, end-host characteristics, transfer data volumes specific to the respective overlay. These may also vary in time, so dynamically updating the model can be a good idea. Alternatively, insofar as our PL measurements are representative for today's Internet, our results may be used as a guidelines giving insights on how to better perform multipath routing. MARS is a non-parametric regression technique that can be used to model how various input variables (in our study we look at a.b., router-level disjointness and RTT) interact to produce one or more response variables, in our case TCPT.

We chose MARS over other regression techniques because it is a relatively advanced method that has several advantages. Firstly, it tends to build easily interpretable models, much more so than competing approaches such as neural networks or random forests. A result MARS model is a simple polynomial function, of controllable degree, in the form of a weighted sum of so-called hinge functions or products of these. Please see the TCPTI formula in subsection IV-B for an example. The hinge functions are terms or factors in the formula, such as $\max(0, \text{router_disjointness_IPs} - 4)$. They are automatically produced by the MARS algorithm to model kinks in the data dependency. In our case, the kink at point 4 shows that a router-level disjointness > 4 has a different impact on TCPT than a smaller one, thus is multiplied by a different coefficient. Terms in a MARS formula sum (also called basis functions) can be either constants (an intercept), *additive* (e.g. the term $3.8 * \max(0, \text{srd_available_bwidth} - 48408)$) or *multiplicative*, e.g. the product of two hinge functions of disjointness and bandwidth, as in the TCPTI formula. The latter model an interaction between variables, in our case that throughput is benefited more by a.b. as disjointness increases. The max degree of interaction, i.e. the maximum number of hinge functions that can be multiplied to form a term, can be tuned in MARS, e.g. to trade off formula simplicity for accuracy. By automatically discovering the above-mentioned kinks in graphs, MARS is much more flexible than linear models in modeling and fitting non-linear dependencies and better contains effects of outliers.

Secondly, MARS is designed to avoid *overfitting*, i.e. building models that fit a particular set of measurements very well, but do not generalize to other conditions. Avoiding overfitting in the presence of noise, as routinely encountered in Internet

measurements due to cross traffic, transient events etc is crucial. A very large model could be built to optimally fit a training set, e.g. by minimizing the RSS (Residual Sum of Squares). However, the model might then capture noise and characteristics of the training set that will not generalize, e.g. for an updated overlay or some time later on the same overlay. MARS has a two-step approach, whereby it first does a *forward pass*, attempting to best fit the model (incrementally adding basis functions s.t. the RSS is minimized), and then a *pruning pass*, whereby the model is shrunk by dropping terms while maximizing a different metric capturing generalization power, called Generalized Cross Validation (GCV). GCV is rooted in leave-one-out validation techniques (cross validation), a standard modeling technique whereby the training data is randomly split in a proper training set (used to build the model) and a separate test set, used to calculate the error. Models with best prediction capability on new data are thus picked from many such splits (also called "folds"). We find the ability to prevent overfitting especially important for deriving noisy and outlier-prone Internet properties.

MARS is additionally fast and scales to many variables and measurements, routinely being used for hundreds of parameters and tens of thousands of measurements. For our training sets, in the order of thousands of measurements, building a model takes seconds to at most minutes, depending especially on the chosen degree of interaction. We find this cost totally acceptable for dynamic model updating, even when scaling to large or dynamic overlays. Further details on how MARS produces regression functions can be found in [6][5] and a detailed description is out of scope within current space constraints. We briefly overview how the algorithm works to give some insight. The dependency is estimated in the form of "splines". A spline is obtained by dividing the range of the x variables into $K+1$ regions, separated by K knots and fitting a separate function in each region using mean square error minimization. MARS is basically a search algorithm that does several things automatically: it detects variables (including combinations of variables, modeled as products) that most influence accuracy, coefficients, as well as knots (breakpoints in the graph) that best fit the data. This would in principle very heavyweight, involving an extensive search over the space of all variables (and combinations thereof) and all possible values of knots (each interior data point) to consider as candidates. However, MARS manages to achieve this surprisingly fast. It consists of an initial "forward" pass, whereby at each iteration, a pair of symmetric basis functions (of opposite sign) are added to the model; choosing which variable(s) and knot values to add next is done by picking a basis function previously added (a greedy search basically) and multiplying it by a new basis function, picked to minimize the mean squared error of the result; this goes on until a max number of terms or an improvement threshold is reached. Normally the model is allowed to grow very big at this stage, in a deliberate attempt to overfit it. Then, in the backward selection phase, terms are dropped based on the GCV metric mentioned above until the latter is optimal, which yields a more general (and smaller)

model. While the "proper way" to increase generalization power is in principle a full cross validation search (building many MARS models for slices of training data and calculating RSSs on the leftout data, then picking models that generalized best, this would be computationally expensive, thus MARS develops GCV as a simplified estimator, allowing this to be done faster via one single MARS model build pass [6]. GCV is based on both squared error, sample size and a penalty for increased model complexity that each term incurs, and was shown to fare well in practice. As we will explain later, we also employ full-blown cross validation (F-fold) in our final TCP model picking.

IV. INTERNET MEASUREMENTS

A. Measurement setup

We perform PL measurements of multipath TCPT using `mypperf`. We first pick a random training set of 100 paths. For each, we choose multiple nodes as relays. About 40 relays are picked for each of the 100 paths. Relay choice is not totally random, but we also attempt to properly cover the parameter space. For disjointness for instance, since random choice would bias towards low disjointness, we use topology traces to properly cover the range of existing disjointness levels. For each (S,D,R) tuple we perform periodic `mypperf` measurements of both multipath and direct path TCPT. The 2 measurements are performed back-to-back, so as to reduce the effect of noise due to cross-traffic and PL node load. All matching measurements for a path are aggregated over time via median, for robustness. `mypperf` is tuned as described and we use transfer sizes between 20 and 80MB. These moderate sizes (still bigger than e.g. [16]) are due to the need to reduce PL traffic, as PL imposes a daily limit on both inbound and outbound traffic per node of 4.8 GB. As we had many measurements to perform (about 3000 tuples and 100 S-D pairs common to many such tuples), we chose the above sizes, which are still big enough for TCP to spend considerable time outside slow start. Result aggregates are used as response values in building MARS models.

The MARS method has in fact quite many parameters that can drastically influence the outcome, thus it needs to be carefully tuned for the specific problem, as we found out. Tweaks to MARS include: setting penalty values for knots and variables added to model (influencing model size and generalization power), search termination conditions, degree of interaction, pruning method [5] etc. We also variate the quantile levels used for a.b. CCDFs (we explore 0.1, 0.3, 0.5, 0.7 and 0.9). For tuning all these parameters, we took an automatic approach whereby we build a "bag" of MARS models that we evaluate. As quality measure for models we use cross validation. Models are generated on a subset of the training data and the error is calculated on the left-out data, using the R-Squared coefficient (normalized RSS). This is repeated a number of times and a mean of R-Squared results calculated to represent a model quality metric (also called CV-RSq, for Cross Validated R-Squared), which is maximized to pick final models. We apply this extra step for automatic tuning

and to increase generalization power, thus capture generally applicable Internet properties. Within model bags, we also variate predictor variables, involving: a.b. distributions, e2e RTT latencies and topology traces for the disjointness metric defined. We try to predict both multipath TCPT and the improvement TCPTI. The latter can be used directly to solve the relay picking problem, i.e. which relay to use, if at all. The former can be used indirectly if an alternate prediction method for single path TCP is preferred.

We illustrate by presenting two models in the next sections. The first predicts TCPTI achievable by multipathing from all of: S-D a.b., S-R-D a.b. and router disjointness. The second predicts raw multipath TCPT from all afore-mentioned parameters plus S-R-D RTT. The former is tuned to be simple and less accurate (would not pass our model bag selection), shown to illustrate improvement for the second and to exemplify the kinds of heuristics that can be derived and visualised. The latter is more accurate, suitable for real-time prediction. As final validation (besides R-Sq and CV-RSq), we employ MARS models within PathInfo to predict good relays for totally new paths, not present in any set, and run transfers at a later time than training set traces (months after the first training data was collected). We then compare achieved TCPT with 4 heuristic approaches and show benefits.

First, a description of the general benefits of multipathing we observed: 35.7% of all relays caused negative TCPTI, while the rest improved TCPT. We omit the CDF due to lack of space, but quantiles at 0(min value),0.3,0.5(median),0.7,0.9,1(max) of TCPTI are: -39.85,-0.2,0.73,3.6,11.3,38.53 Mbps respectively. Negative values suggest that many overlay transfers can do better running multiple TCP threads on the direct path as opposed to overlay routing. We note that multipathing can be quite useful, but accurate prediction is key. The distribution of raw multipath TCPT for paths we evaluated, on the same quantile points was: 0.9,13.3,21.7,35.9,60,91.3 Mbps.

B. A simple model of TCP throughput improvement

We use the estimated a.b. values on both the S-D link AB_{sd} and alternate link AB_{srd} , latter computed as min of estimated a.b. of the 2 compound links S-R and R-D, and the router disjointness $disj$. We predict absolute TCPTI *improvement TCPTI*. The MARS formula regressed is relatively simple, having only 6 terms, and using all 3 predictors:

$$\begin{aligned} TCPTI = & 4.03 - 0.43 * \max(0, 26.7 - AB_{srd}) \\ & - 0.11 * \max(0, AB_{sd} - 54.08) \\ & + 0.03 * \max(0, disj - 4) * \max(0, AB_{srd} - 60.91) \\ & + 0.01 * \max(0, 10 - disj) * \max(0, AB_{sd} - 54.08) \\ & + 1.17e-05 * \max(0, 26.70 - AB_{srd}) * \max(0, 32.88 - AB_{sd}) \end{aligned}$$

TCPTI and bandwidths are expressed in Mbps and a.b. is estimated at the 0.3 quantile (i.e. value < 70% of values in the a.b. distribution), which we generally found to be a source of inaccuracy, as more conservative values yield more accurate models. We see disjointness interacting with both

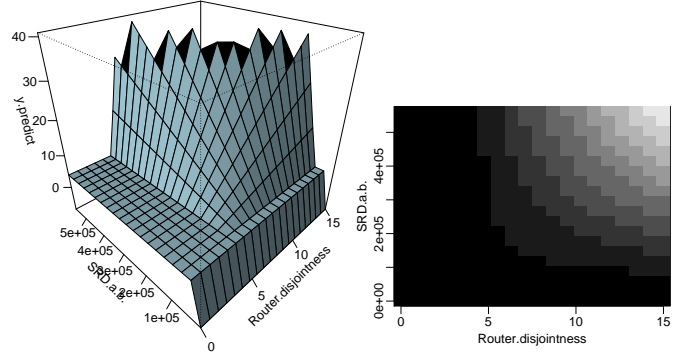


Fig. 2. Disjointness - alternate path bandwidth interaction (improvement model)

Fig. 3. Plane section of Fig. 2

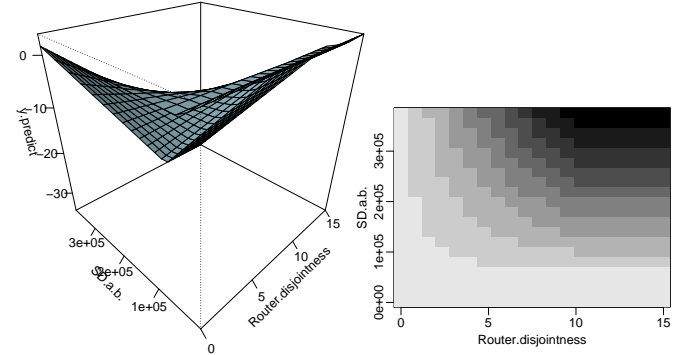


Fig. 4. Disjointness - direct path bandwidth interaction (improvement model)

Fig. 5. Plane section of Fig. 4

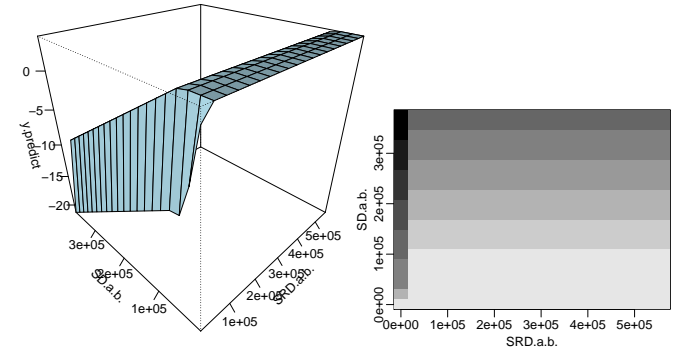


Fig. 6. Interaction between bandwidths on direct and alternate paths (improvement model)

Fig. 7. Plane section of Fig. 6

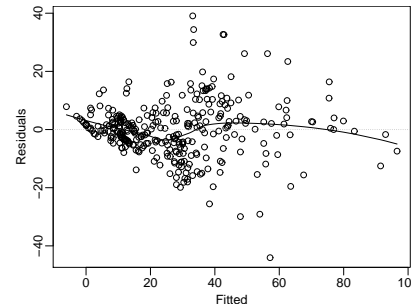


Fig. 8. Distribution of residuals per model response (throughput model)

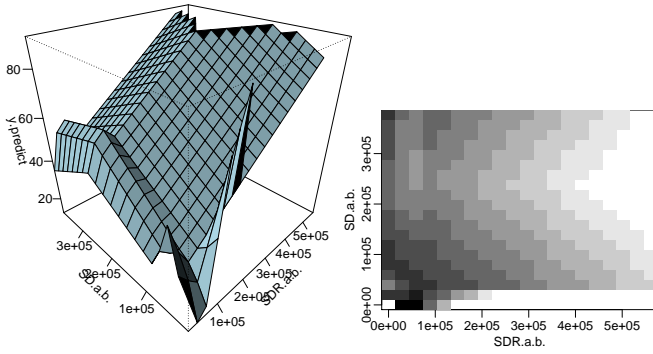


Fig. 9. Interaction between bandwidths on direct and alternate paths (throughput model)

Fig. 10. Plane section of Fig. 9

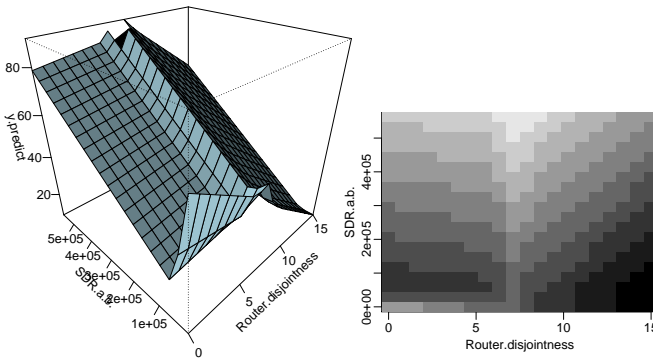


Fig. 11. Disjointness - alternate path bandwidth interaction (throughput model)

Fig. 12. Plane section of Fig. 11

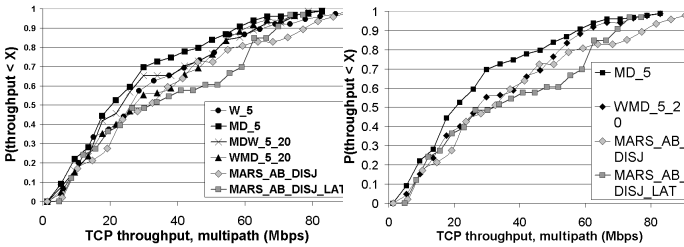


Fig. 13. CDFs of throughputs Fig. 14. Same CDFs, but with the achieved by regression methods vs. middle 2 heuristics removed, to avoid over-cluttering

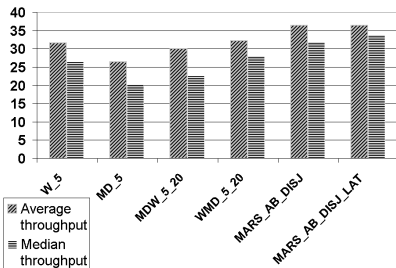


Fig. 15. Averages and medians of TCPTs achieved by various algorithms

direct and indirect path bandwidths (the multiplicative terms). Tight space constraints do not allow us to present a CDF of absolute residuals (i.e. differences between observed and predicted values) or a graph of where residuals lie relative to predicted values (similar to Fig. 8 shown for the next, more accurate model). To report the CDF, we see 50% of predictions off by < 3 Mbps and 75% by < 9 Mbps, so not particularly accurate. However, residual dispersion (analogue to Fig. 8), shows the model relatively accurate for TCPTI values in the 0-3 Mbps interval (where most predictions lie), and quite inaccurate outside it. The R-Sq metric indicates how well the model fits the training data, where 1 means perfect fit, 0 no fit, values > 0.3 a mild fit and > 0.6 a significant fit. For current model, R-Sq is 0.30522 (mild), and GRSq (Generalized R-Sq, indicating generalization power via cross validation as discussed above) is 0.25702. We mainly show the present simple model to show how the next one improves, and to illustrate kinds of heuristics that our regression produces. Typical models we generate have more terms and higher quality, with correlations > 0.7 . Figures 2-7 show interactions between variables, which even at current accuracy show interesting trends. Interaction graphs between any 2 variables are built by pinning remaining vars at median values. From Fig. 2-3, we see TCPTI levelling off for disjointnesses < 5 , even as AB_{srd} grows. Above this disjointness, TCPTI indeed grows with AB_{srd} (heuristic H1). Some 3D graphs are hard to view because of perspective angle, so we also show a projection (e.g. Fig. 3), where lighter gray-shades mean higher values and black is lowest. Fig. 4-5 is somewhat symmetric to Fig. 2-3, showing TCPTI declining as AB_{sd} grows, and this gets worse with high disjointness (H2). This suggests that as single path TCPT on the direct path grows with AB_{sd} , improvement achieved decreases. Correlation with high disjointness may reflect that bottlenecks on high a.b. paths cannot benefit by alleviation by the alternate path, in which case the penalty of detouring via a highly disjoint relay (which is probably increasingly further RTT-wise) dominates. Fig. 6-7 shows that, at median disjointness, the dominant metric influencing TCPTI is AB_{sd} , not AB_{srd} (H3), which concurs with H2, i.e. paths with high a.b. benefit less from multipathing. Reasons for inexactness of the current model stem from: use of the more inaccurate 0.3 quantile, not including latency (or other variables, e.g. loss rate or end host capabilities) and limiting interaction degree to 2 to keep the formula easily interpretable.

C. A model of multipath TCP throughput

We present next a model of TCPT based on the previous metrics plus alternate path (SRD) RTT. The quantile for a.b. is 0.7, as returned optimal by our automated model selection and interaction degree is 3. It has an R-Squared of 0.78927 (strong fit), GRSq of 0.6515 (generalization power). It has 30 terms, using all 4 predictors. We describe the CDF due to space constraints and show in Fig. 8, the distribution of residuals w.r.t. the range of values predicted. From the CDF, 75% of values are off by < 10 Mbps, and 50% by < 5 Mbps, a high improvement to the previous model, since predicted

value is TCPT itself, not TCPTI; looking at TCPT and TCPTI distributions we outlined in section IV-A, the former has much higher values, e.g. 0.7 quantile at 35.9 as opposed to 3.6 Mbps. Fig. 8 shows the model is quite accurate (within 5Mbps, except outliers) for TCPT <25 Mbps (where most values fall), loosing accuracy and slightly underestimating for bigger values. Fig. 9-12 quantifying variable interactions, are again built by fixing all other variables at median values. Fig. 9-10 plots interaction between bandwidths on the 2 paths. We notice a very interesting effect: a wider alternate path always helps, but a wider direct path only seems to help in conjunction with SRD being wider as well and mostly up to 300 Kbps. Beyond that, we may actually observe decrease unless AB_{srd} increases to a similar value, which suggests rerouting through a lower bandwidth relay for wider paths is actually bad in this overlay. Fig. 11-12 show the dependency between AB_{srd} and disjointness, where we see another interesting effect. At median values for the other parameters, disjointness exhibits a local maximum around 6 routers, beyond which increasing disjointness only helps if a.b. of the respective relay is high as well. To save space, we do not include the other 2 interactions, but in short, the AB_{sd} -disjointness interaction shows that disjointness >7 routers, enables TCPT to grow sharply with AB_{sd} . Main effect of the disjointness- RTT_{srd} interaction shows TCPT drop sharply as RTT of the alternate path grows in the interval <100ms, which seems to suggest the presence of lossy paths, where RTT matters.

D. Model validation

To further validate our models through transfer experiments, we compare results achieved by relays returned by the regression functions with those achieved by 4 heuristics: Widest paths (W), Most Disjoint paths (MD), Widest Most Disjoint (WMD) and Most Disjoint Widest (MDW). W means best provisioned w.r.t a.b., heuristic similar to that proposed in BARON [14]. MD means picking relays providing most router-level disjointness. For W and MD the 5 top relays are picked for each of cca 100 paths and their achieved TCPT results averaged. WMD means that first the top K most disjoint relays are picked (currently 20) and out of these we pick the 5 widest. MDW is symmetric to WMD. MDW and WMD are similar to heuristics suggested in iPlane [16] (there the RTT-closest path out of top K lowest loss paths are picked). As models, we use a function very similar to the one in section IV-C, including latency (MARS_AB_DISJ_LAT), as well as a function that excludes latency, similar to section IV-B, but much more accurate (it has degree of interaction 3 and more terms). It is called MARS_AB_DISJ. Results are shown below, as CDFs, averages and medians. We see that both regression functions outperform the heuristics and that the one disregarding latency performs quite close on average to the complete one. However, it seems to do this by achieving very high throughputs for a few relays, whereas the complete one performs better for most relays, as shown by the CDFs and median. We hypothesize this is due to the RTT-enabled one doing better on lossy paths, where RTT matters. Regarding the

4 heuristics, we note that $MD < MDW < W < WMD$.

V. CONCLUSIONS

We propose a novel approach to improving overlay routing via dynamic modeling. Unfortunately, optimizing underlying WAN performance for applications such as grids and p2p computing has proved a non-trivial problem, involving many variables, much denoising, thus it is notoriously hard to get right. Our approach enables such overlays to dynamically model complex, application-relevant QoS parameters, based on an arbitrarily large number of measurable, less complex predictor metrics, possibly specific to that overlay. We evaluated our method by predicting TCPT on 1-relay multipaths over Internet and gained interesting insights on ways application-level overlays should be used. Despite previous "best practice", we find that disjointness of alternate paths alone is not a sufficient heuristic, as nor are available bandwidth or RTT distance. In fact, we show that our approach of precisely modeling metric interactions outperforms simpler heuristics, even those taking into account multiple such metrics. We expect the issue to be even more critical if other metrics, e.g. relay end-host capabilities come into play and our approach is designed to scale to such settings. We plan to extend a similar study to other multipath streaming setups used by distributed applications, e.g. application-layer multicast or gathering from several replicas. We feel that our system bridges a much-needed gap between underlying network measurement and effective use of modeling to predict relevant performance.

REFERENCES

- [1] D.G. Andersen, H. Balakrishnan, M.F. Kaashoek, and R. Morris. Resilient overlay networks. In *SOSP*, 2001.
- [2] A. Banerjee. Simulation study of the capacity effects of dispersity routing for fault tolerant realtime channels. *SIGCOMM*, 1996.
- [3] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. In *SIGCOMM*, 2004.
- [4] T. Fei, S. Tao, L. Gao, and R. Guerin. How to select a good alternate path in large peer-to-peer systems. *INFOCOM*, April 2006.
- [5] Friedman. Fast mars. In *Stanford University Dept. of Statistics, Technical Report 110*.
- [6] Friedman. Multivariate adaptive regression splines (with discussion). In *Annals of Statistics 19/1, 1-141*, 1991.
- [7] K. P. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, and D. Wetherall. Improving the reliability of internet paths with one-hop source routing. In *OSDI*, 2004.
- [8] Q. He, C. Dovrolis, and M. Ammar. On the predictability of large transfer tcp throughput. *SIGCOMM CCR*, 35(4):145-156, 2005.
- [9] M. Jain and C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput. *SIGCOMM Comput. Commun. Rev.*, 32(4):295-308, 2002.
- [10] G. Jin and B. Tierney. Netest: A tool to measure the maximum burst size, available bandwidth and achievable throughput. *Lawrence Berkeley National Laboratory TR. LBNL-48350*, 2003.
- [11] G. Jin and B. Tierney. System capability effects on algorithms for network bandwidth measurement. In *IMC*, 2003.
- [12] P. Karbhari, M. Ammar, and E. Zegura. Optimizing end-to-end throughput for data transfers on an overlay-tcp path. In *Networking*, 2005.
- [13] K. Lai and M. Baker. Nettimer: a tool for measuring bottleneck link, bandwidth. In *USITS*, 2001.
- [14] S. Lee, S. Banerjee, P. Sharma, P. Yalagandula, and S. Basu. Bandwidth-aware routing in overlay networks. In *INFOCOM*, 2008.
- [15] D. Lu, Y. Qiao, P. Dinda, and F. Bustamante. Characterizing and predicting tcp throughput on the wide area network. In *ICDCS*, 2005.
- [16] H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iplane: an information plane for distributed services. In *OSDI*, 2006.

- [17] H. Madhyastha, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iplane nano: path prediction for peer-to-peer applications. In *NSDI*, 2009.
- [18] NF Maxemchuk. Dispersity routing. *Proceedings of ICC*, 1975.
- [19] A. Nakao, L. Peterson, and A. Bavier. A routing underlay for overlay networks. *SIGCOMM*, 2003.
- [20] T. Nguyen and A. Zakhor. Path diversity with forward error correction (pdf) system for packet switched networks. *INFOCOM*, 2003.
- [21] L. Qiu, Y. Zhang, and S. Keshav. On individual and aggregate tcp performance. In *ICNP*, 1999.
- [22] S. Vazhkudai, J. Schopf, and I. Foster. Predicting the performance of wide area data transfers. In *IPDPS*, 2002.
- [23] P. Yalagandula, P. Sharma, S. Banerjee, S. Basu, and S.-J. Lee. S3: a scalable sensing service for monitoring large networked systems. In *SIGCOMM workshop on Internet network management*, 2006.
- [24] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the characteristics and origins of internet flow rates. In *SIGCOMM*, 2002.