Using the Last-Mile Model as a Distributed Scheme for Available Bandwidth Prediction

Olivier Beaumont[1], Lionel Eyraud-Dubois[1], and Young J. Won[1,2]

[1] INRIA Bordeaux Sud-Ouest
LaBRI, University of Bordeaux 1, Talence, France
{Olivier.Beaumont,Lionel.Eyraud-Dubois}@labri.fr
[2] IIJ Research Lab., Tokyo, Japan
young@iijlab.net

Abstract. Several Network Coordinate Systems have been proposed to predict unknown network distances between a large number of Internet nodes by using only a small number of measurements. These systems focus on predicting latency, and they are not adapted to the prediction of available bandwidth. But end-to-end path available bandwidth is an important metric for the performance optimisation in many high throughput distributed applications, such as video streaming and file sharing networks. In this paper, we propose to perform available bandwidth prediction with the last-mile model, in which each node is characterised by its incoming and outgoing capacities. This model has been used in several theoretical works for distributed applications. We design decentralised heuristics to compute the capacities of each node so as to minimise the prediction error. We show that our algorithms can achieve a competitive accuracy even with asymmetric and erroneous end-to-end measurement datasets. A comparison with existing models (Vivaldi, Sequoia, PathGuru, DMF) is provided. Simulation results also show that our heuristics can provide good quality predictions even when using a very small number of measurements.

Keywords: Network Coordinate System, Last-Mile, Network Measurement, Available Bandwidth Prediction, Labeling Scheme.

1 Introduction

Predicting network performance (latency or available bandwidth) is important for many Internet applications. For video on demand [18] and peer-assisted streaming [14] for example, estimations of available bandwidth allow the construction of an efficient overlay topology.

A number of measurement tools have been developed [9] which measure the available bandwidth on the path between two given Internet nodes. However, in a large scale system, performing measurements between all pairs of nodes would incur too large of an overhead. Thus, there is a need for the possibility to infer

E. Jeannot, R. Namyst, and J. Roman (Eds.): Euro-Par 2011, LNCS 6852, Part I, pp. 103–116, 2011.

(in this paper we also use the term predict[1]) the unmeasured bandwidth values from a limited number of actual available measurements.

For latency estimation, several solutions have been successfully proposed, under the global terminology of Network Coordinate Systems. Most of these solutions embed network nodes into a metric space (not necessarily Euclidean) and approximate the latency between nodes by the distance between their embeddings, which can easily be computed from their coordinates. GNP [16] is an example of such a system, in which each node is positioned in an Euclidean space with respect to a number of landmarks whose positions are already established. Vivaldi [6] is a decentralised extension of GNP, which avoids the need for landmark nodes. However, the efficient counterparts of these coordinate systems for available bandwidth prediction are still to be proposed.

Recent literature about overlay design for peer-to-peer data dissemination [3] has generalised the use of a "last-mile" approximation, in which the rates of simultaneous communications are only limited by the upload and download capacities of each node. This simplifying assumption is quite natural in the context of the Internet, and allows to derive provably efficient overlay designs. In this paper, we analyse the validity of this approximation, and how it can be used to develop a technique for predicting available bandwidth from a limited number of measurements.

More precisely, we propose a decentralised heuristic to compute the capacities of each node from a relatively small number of measurements. We analyse this heuristic by using a dataset of available bandwidth measurements performed on PlanetLab [5]. The accuracy of the predicted values with our solution compares favourably with existing solutions, while requiring significantly fewer measurements.

The organisation of the paper is as follows. We first describe the related works in Section 2. Section 3 presents the rationale behind the last-mile model and our proposed heuristic to compute the capacities of the nodes. In Section 4, we present the evaluation of our solution and compare it to other existing solutions. Concluding remarks and future works are given in Section 5.

2 Related Works

2.1 Latency Estimation

In the context of latency estimation, a number of network coordinate systems have been proposed, based on the following idea: embedding the nodes of the network into a multi-dimensional space and using the distance between two

[1] The design of techniques for efficient and reliable available bandwidth measurements is an interesting research question, but it is not in the scope of this paper. Instead, we assume in this work that a (limited) number of measurements are available, and our goal is to use these measurements to provide estimations for the unmeasured bandwidth values.

points in this space as an estimation of the latency between the corresponding nodes. We can make a distinction [11] between landmark-based and decentralised approaches. In the landmark-based approach (*e.g.* GNP [16], PIC, etc.), a fixed number of landmark nodes are selected and positioned in the space. Non-landmark nodes measure their latency to these landmark nodes and compute their coordinates so as to minimise the resulting prediction error. On the other hand, in the decentralised approach (*e.g.* Vivaldi [6], Big Bang Simulation, etc.), all participating nodes have the same role, and the coordinates of the nodes are computed in a decentralised way by direct measurements between participating hosts.

Vivaldi [6] is one of the most well-known decentralised coordinate system. It relies on the simulation of a system of springs, in which the interaction between two nodes is modeled by a spring whose force represents the estimation error. This simulation procedure allows to adapt the computed coordinates to changing network conditions.

Recent works have also studied embedding into a hyperbolic structure [7]. An example of such a system is the Sequoia algorithm [17], which embeds the nodes as the leaves of a weighted tree, and approximates the distance between two nodes by the length of the path between their respective positions in the tree. The Sequoia algorithm comes with a theoretically proven performance guarantee, and can be applied to both latency and bandwidth estimation. However, the algorithm is quite sensitive to violations of the triangular inequalities, and there is for the moment no decentralised version of Sequoia: computing the embedding requires the measurements between all pairs of nodes.

An important problem with metric-based embeddings comes from the violations of triangle inequalities, which are often observed in Internet measurements. Several studies have thus considered non-metric embeddings. IDES [15] is based on matrix factorisation, which consists of approximating a large matrix by the product of two smaller matrices. Each node is thus assigned two vectors (an incoming and outgoing vector), which correspond respectively to one row and one column of the two smaller matrices. The distance between two nodes A and B is computed as the scalar product of the outgoing vector of A and the incoming vector of B. The IDES system is based on a set of landmark nodes, and recently a decentralised version has been proposed, called DMF [13] for decentralised matrix factorisation. DMF is an iterative procedure in which each node locally minimizes the prediction error by solving a least square problem.

2.2 Bandwidth Estimation

There is a relatively small number of studies focusing on bandwidth estimation. The authors of Sequoia [17] studied the applicability of their algorithm to available bandwidth, and the works based on matrix factorisation [13] can be applied to bandwidth estimation as well. PathGuru [19] is a landmark-based system specifically designed for available bandwidth estimation, which relies on the observation that, in certain circumstances, Internet available bandwidth forms an ultra metric space. In PathGuru, each node measures the available bandwidth to

and from every landmark, and the estimation of bandwidth between two given nodes A and B is performed using the pair of landmarks which most closely forms an ultra metric space with A and B.

BRoute [10] is another system for available bandwidth estimation, which is based on the observations that most bottleneck links are on the path edges, and that relatively few routes exist near the source and destination. Unlike the previous solutions which only require end-to-end measurements, BRoute uses landmarks and network management tools (such as traceroute and BGP routing information) to identify the bottleneck links near each source and destination, and to infer which links are used by packets between A and B.

A number of works in the literature of communication optimization in large scale systems assume that each participating node is characterised by its upload and/or download bandwidth. This applies to a variety of topics, such as video-on-demand [18,4], peer-assisted streaming [14,3,2] or multi-port divisible load scheduling [1]. Thanks to its simplicity, this rather natural assumption allows to derive provably efficient algorithms. In this paper, we analyse how well this model can approximate the actual available bandwidth in a large scale distributed platform.

3 Last-Mile Bandwidth Prediction Model

3.1 Last-Mile Model

Throughout the paper, we denote by $\mathcal{M}_{A,B}$ the measured available bandwidth from node A to node B, and by $\mathcal{P}_{A,B}$ the corresponding predicted value.

The previous research on the properties of the Internet indicate that the bandwidth at the edge of the network, the so called last-mile (end-host) bandwidth, reflects the overall performance of the complete end-to-end path. Hu et al. [12] show that 60% of wide-area Internet paths between end-hosts have their bottleneck in the first or second hop. A recent study also shows a similar property in broadband access networks [8]. As an insight, we have observed the dataset obtained from measurements on the PlanetLab platform [5] which is described more precisely in Section 4 where it is used to evaluate our heuristics.

As a representative example, Figure 1 is the plot of the outgoing bandwidth measurements from hosts `planetlab3.hiit.fi` and `planet-lab7.millenium.berkeley.edu`, which will be denoted `hiit` and `berkeley` in what follows, to all the other hosts in the platform. The bandwidth values are sorted for increased readability. The plot for `berkeley` (Figure 1(b)) shows what can be expected for a host with a low outgoing capacity: the bandwidth to the first 50 hosts is limited by their respective ingoing capacity, and then the bandwidth to the rest of the nodes is limited by the outgoing capacity of `berkeley`, and thus the plot remains quite flat. On the other hand, the plot for `hiit` (Figure 1(a)) shows the result for a host with a large outgoing capacity: the bandwidth increases quite smoothly. In both cases however, a small number of larger bandwidth measurements can be noticed by a sharp increase around node 300, which can be interpreted as bogus measurements and will be discussed later.

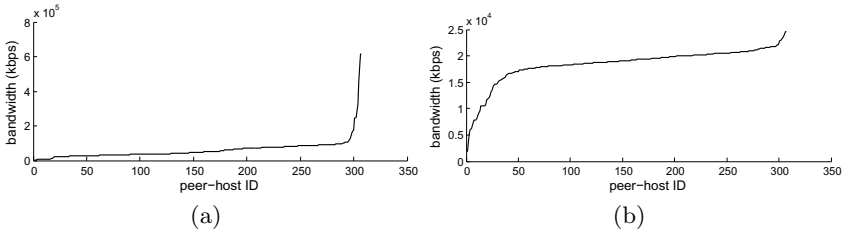


Fig. 1. Outgoing bandwidth distributions for two PlanetLab hosts; the values are sorted in increasing order: (a) planetlab3.hiit.fi; (b) planetlab7.millennium.berkeley.edu

This observation has led to propose a last-mile modelisation of available bandwidth [14]. In this model, each participating node x is represented by two different bandwidth capacities, one for upload (β_x^{out}), and one for download (β_x^{in}). Based on these values, and assuming that they are the only limiting factors for the end-to-end performance, the predicted bandwidth $\mathcal{P}_{x,y}^{\text{LM}}$ between two nodes x and y is given by $\min(\beta_x^{\text{out}}, \beta_y^{\text{in}})$.

In this paper, in order to analyse the validity of this model, we propose heuristics to compute values for β_x^{out} and β_x^{in} so as to minimise the prediction error. We first propose a simple way to compute reasonable initial values, and then describe an iteration procedure following the one used in the context of DMF [13].

3.2 Initial Values

The initial observation may identify that the upload capacity of a node β_x^{out} has to be at least as large as the measured value between node x and any other node y (otherwise it would have been impossible to measure such a high value for this particular node y). However, setting $\beta_x^{\text{out}} = \max_y \mathcal{M}_{x,y}$ is potentially dangerous: only one bogus measurement is enough to obtain wrong predictions for x. Furthermore, the last-mile assumption is not always satisfied in practice, and it may happen that some nodes share a bottleneck link. A typical example is the case of two nodes A and B on a common local area network, which is connected to the Internet through a DSL connection. The bandwidth from A to any other node X on the Internet is then limited by this DSL connection, while the bandwidth from A to B is not. Hence, setting $\beta_A^{\text{out}} = \max_y \mathcal{M}_{A,y} = \mathcal{M}_{A,B}$ would result in largely over-estimated predictions for bandwidth from A to any X on the Internet, since it would ignore the limiting DSL connection.

This situation can be observed on our dataset. For instance, on Figure 1(a), most values lie below 2.10^5 kbps, except for a couple of outliers with very high measured bandwidth, which can be either erroneous measurements or hosts with a local, direct high-capacity connection to `hiit`.

This observation motivates the removal of a few outliers hosts before computing β^{in} and β^{out} values. The solution we propose in this paper is to define β_A^{out} as a given percentile $1 - \alpha$ of all measured values $\mathcal{M}_{A,y}$. It is a generalisation of the previous straightforward answer of taking the maximum value, which corresponds to $\alpha = 0$; larger choices of α ignore more and more measurement values.

This solution yields a very simple way of computing β^{out} and β^{in} values, and is very resilient to missing and erroneous measurements and "too high" bandwidths due to nodes in the same local network, since corresponding measurements are ignored.

Algorithm 1. Computing initial values for the last-mile

Input: $\mathcal{M}_{,}$: measurement matrix
k: number of neighbours for each node
α: percentile parameter
Output: β^{out} and β^{in}
for all node A **do**
select a random set S of k neighbours
sort $up = (\mathcal{M}_{A,y})_{y \in S}$ and $down = (\mathcal{M}_{y,A})_{y \in S}$
$\beta_A^{\text{out}} = (1-\alpha) -$ percentile of up
$\beta_A^{\text{in}} = (1-\alpha) -$ percentile of $down$
end for

With this procedure, each host can compute its own β^{out} and β^{in} values independently, assuming that it has access to the measurements of all pairs it is involved in. Furthermore, a standard technique to reduce the measurement overhead (at the cost of accuracy) is the *random sampling* of the hosts, in which each host selects a random subset of neighbours and performs available bandwidth measurements to and from this subset. By computing the $(1-\alpha)$-th percentile of these measurements to use as β^{out} and β^{in}, the result is expected to be a reasonable approximation of the real β^{out} and β^{in} values obtained if all measurements were available. This results in Algorithm 1.

In practice, many overlay networks provide the ability of choosing a random node, either by construction (*e.g.* Distributed Hash Tables) or using gossiping algorithms, so that random sampling can easily be implemented in a distributed way.

3.3 Iterative Procedure

In order to improve on this initial calculation, following Vivaldi [6] and DMF [13], we propose a procedure in which nodes iteratively update their β^{in} and β^{out} values. To update its β^{out} value, each node A obtains the values of β^{in} from its neighbours, and sets β_A^{out} to the value x that minimises the prediction error:

$$E(x) = \sum_{y \in S} \left(\mathcal{M}_{A,y} - \min(x, \beta_y^{\text{in}})\right)^2 \tag{1}$$

The value of β_A^{out} which minimises this quantity can be easily computed, since each y such that $\beta_y^{\text{in}} \leq \beta_A^{\text{out}}$ contributes a constant factor to the error. Hence for x between two consecutive values $\beta_{y_1}^{\text{in}}$ and $\beta_{y_2}^{\text{in}}$, the error can be rewritten as:

$$\sum_{\beta_y^{\text{in}} \leq \beta_{y_1}^{\text{in}}} \left(\mathcal{M}_{A,y} - \beta_y^{\text{in}}\right)^2 + \sum_{\beta_y^{\text{in}} > \beta_{y_1}^{\text{in}}} \left(\mathcal{M}_{A,y} - x\right)^2 \tag{2}$$

And this expression is minimised for x equal to the average of the $\mathcal{M}_{A,y}$ values for $\beta_y^{\text{in}} > \beta_{y_1}^{\text{in}}$. Of course, if this value is above or below the prescribed interval, x is set to the corresponding bound of the interval. By sorting the β_y^{in} values and testing all the k possible intervals, it is possible to compute the value of x that minimises equation (1). The resulting iterative procedure is described in algorithm 2.

Algorithm 2. Iterative procedure.

Input: $\mathcal{M}_{,:}$ measurement matrix
k: number of neighbours for each node
α: percentile parameter
i: number of iterations
Output: β^{out} and β^{in}
Initialise β^{out} and β^{in} with Algorithm 1
for i iterations **do**
  for all node A **do**
    Sort $(\beta_y^{\text{in}})_{y \in S_A}$
    for all interval l $(\beta_{y_l}^{\text{in}} \leq \beta_{y_{l+1}}^{\text{in}})$ **do**
      Compute x_l which minimises eq (2)
    end for
    Select l so that $E(x_l)$ is smallest (eq (1))
    Update $\beta_A^{\text{out}} = x_l$
    Update β_A^{in} similarly
  end for
end for

4 Evaluation

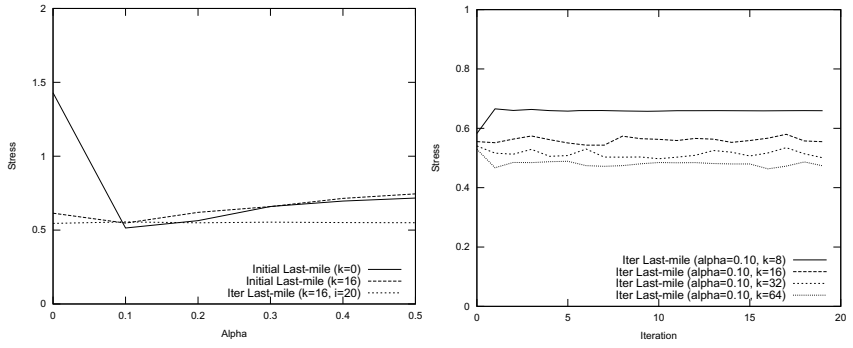
4.1 Methodology

The experimental results described in this paper are based on a dataset from the S-cube project [20]. This project aims at monitoring the large scale distributed platform PlanetLab [5]. Available bandwidth is measured between almost all pairs of nodes of PlanetLab, and results are made available as regular snapshots of the platform. For space reasons, we only present here results obtained from the snapshot of April 20th, 2010; however other snapshots yield the same conclusions. This snapshot contains 426 hosts, with some missing measurements, and we extracted a set of 308 hosts for which the complete measurement matrix is available[2].

The quality of the prediction algorithms is given by the precision of the predictions compared to the original values. In this paper, we use the *modified relative error* as defined by the authors of IDES [15]:

$$e_{x,y} = \frac{|\mathcal{M}_{x,y} - \mathcal{P}_{x,y}|}{\min(\mathcal{M}_{x,y}, \mathcal{P}_{x,y})}$$

[2] The code and dataset used to obtain the results of this section will be made publicly available upon acceptance of the paper.



(a) (b)

Fig. 2. (a) Stress of the last mile embeddings for different values of α; (b) Stress of the last mile embeddings after each iteration

where the min-operation serves to increase the penalty for underestimated values. Most plots in this paper depict the cumulative distribution function (CDF) of the modified relative error for all pairs of hosts. Therefore, if algorithm $\mathcal{A}$ provides better estimations than algorithm $\mathcal{B}$, then the plot corresponding to $\mathcal{A}$ will be above the plot corresponding to $\mathcal{B}$ in the graph.

Sometimes it is more convenient to represent the fitness of the embedding with a single value. In that case we will consider the 80-th percentile of the modified error ratios, *i.e.* the error e such that 80% of the node pairs have their available bandwidth estimated with error at most e. We also measure the stress, to represent the global error of the prediction, defined by:

$$stress = \sqrt{\frac{\sum_{x,y}(\mathcal{M}_{x,y} - \mathcal{P}_{x,y})^2}{\sum_{x,y}\mathcal{M}_{x,y}^2}}$$

4.2 Parameter Tuning

We first analyse the effect of the parameter α on the accuracy of the predictions, both for the initial values obtained with algorithm 1 and for the result of the iterative procedure after 20 iterations (our observations show 20 iterations are enough to reach convergence, see figure 2(b)). The number of neighbours k is fixed to $k = 16$, and we also compare to the special case in which there is no random neighbour selection (all measurements are used), which we denote as $k = 0$. The resulting stress values are depicted in Figure 2(a). For the cases which involve random selection, we report average, minimum and maximum values over 10 runs.

The first observation is that using a non zero value of α is very important when considering all measurements, which is expected as discussed in section 3: a small number of invalid measurements have a very bad impact on the accuracy of the predictions. With random selection ($k = 16$), the effect of the parameter α is not

as big, and it is even lower for the iterative procedure, which effectively improves the fitness of the embedding and gives a result which does not depend on this parameter. This hints that the result of the iterative procedure is independent of the initial values. In the rest of the evaluation, we will use the value $\alpha = 0.1$.

The effect of the parameter k (number of neighbours for each node) is studied in details in the next section, together with the comparison with other prediction heuristics.

We also study the convergence of the iterative procedure by measuring the stress of the fitness obtained after each iteration. The result is shown on Figure 2(b) and shows that the total stress remains stable across the iterations, and that the convergence is fast. We can also see that the initial values computed by algorithm 1 are actually quite precise.

4.3 Comparison Methods

We compare our results with several other solutions from the literature:

- The Vivaldi [6] algorithm provides a basis for comparison even though it was originally designed for latency estimation.
- The Sequoia [17] algorithm, based on tree embeddings, is advertised as being usable for both latency and bandwidth estimation.
- PathGuru [19] is a landmark-based solution explicitly designed for bandwidth estimation.
- DMF [13] is an algorithm which was proposed in the context of latency estimation, but it can be used for bandwidth estimation as well since it does not make any assumption on the structure of the input measurement matrix.

Public implementations of Vivaldi[3] and DMF[4] are available and have been used for this evaluation. However, no implementation seems to be available for PathGuru and Sequoia, so we implemented them based on their description in the corresponding paper.

This dataset is used as input to different prediction algorithms. However, Sequoia and Vivaldi are originally designed for latency prediction, for which smaller values mean that nodes are closer. Hence, these algorithms are fed with the inverse of the available bandwidth measurements[5], and their resulting distance predictions are inversed as well before comparing to the original measurements.

For all algorithms, we used the default values of the parameters as they are described in the corresponding paper (15 prediction trees for Sequoia and $l = 10$ dimensions for DMF). However, we changed the number of neighbours in DMF and landmarks in PathGuru to explore the compromise between accuracy and number of measurements used.

[3] `http://www.eecs.harvard.edu/~syrah/nc/`

[4] `http://www.run.montefiore.ulg.ac.be/~liao/DMF`

[5] This choice is different from the one made in the evaluation of Sequoia [17], in which the authors subtract the bandwidth values from a large constant. Using the inverse as we are doing actually yields better results for Sequoia.

4.4 Evaluation Results

We first analyse the variability of the results with respects to the random choices involved: choice of the levers for Sequoia, of the landmarks for PathGuru, and of the neighbours of each node for DMF and last-mile. We provide in Table 4.4 the average and standard deviation of the stress and of the 80-th percentile of the modified error ratio for 30 runs for each heuristic. For visual comparison, the CDFs of modified relative error for a selection of parameters are given on Figure 3. For a given heuristic and parameter value, the CDFs corresponding to the 30 runs are depicted together on the plot to visualise the variability.

In addition, Figure 4 provides a direct comparison of the most relevant heuristics. On this figure the CDFs of one run for each heuristic are plotted together. The low variability exhibited by table 4.4 ensures that these particular plots are relevant enough.

Table 1. Average and standard deviation of stress and 80-th percentile error

Algorithm	80-th perc. error		stress	
	avg	std	avg	std
Vivaldi ($k = 32$)	3.93	0.98	9800	$7.6\text{x}10^7$
Vivaldi ($k = 128$)	4.68	2.6	7400	$1.3\text{x}10^8$
Sequoia	1.5	0.097	0.73	0.0012
PathGuru ($k = 32$)	2.00	0.55	0.77	0.0013
PathGuru ($k = 64$)	2.58	0.33	0.78	0.00069
PathGuru ($k = 128$)	2.54	0.099	0.79	0.00081
LM ($k = 8$)	0.76	0.0027	0.64	0.00012
LM ($k = 16$)	0.64	0.0012	0.56	0.00024
LM ($k = 32$)	0.64	0.00083	0.51	0.00017
LM ($k = 64$)	0.65	0.0007	0.47	0.00016
LM ($k = 128$)	0.65	0.00019	0.42	0.000043
DMF ($k = 8$)	2.12	0.0079	3.16	2.5
DMF ($k = 16$)	1.33	0.0019	1.14	0.024
DMF ($k = 32$)	0.64	0.00025	0.51	0.00055
DMF ($k = 64$)	0.47	0.000073	0.35	0.00011
DMF ($k = 128$)	0.39	0.000043	0.26	0.000074

The results for Vivaldi show as expected that this algorithm is not appropriate for bandwidth estimation. We can also see that the prediction of last-mile and DMF (for large enough values of k) are much more accurate and stable than the predictions of PathGuru and Sequoia. PathGuru in particular is very sensitive to the choice of the landmarks, and its performance does not really increase with the number of landmarks (however it gets more stable). The predictions of Sequoia are better than those of PathGuru, but remember that Sequoia needs to access the measurements between all pairs of nodes. Sequoia is also (together with Vivaldi) the only heuristic which produces symmetric estimations, and this is a big disadvantage because available bandwidth between two nodes is often asymmetric.

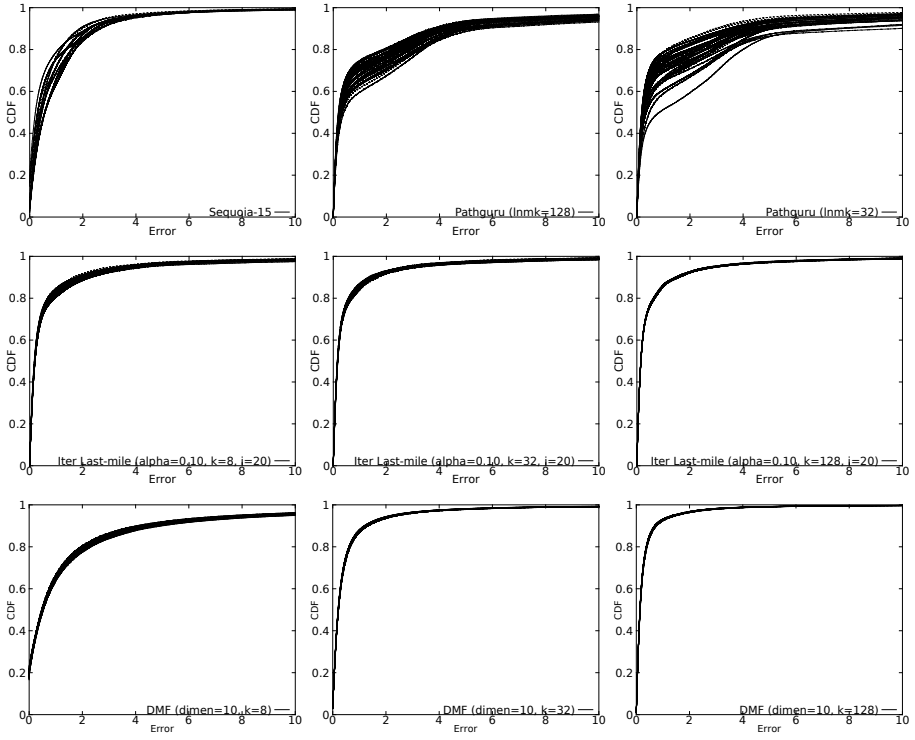
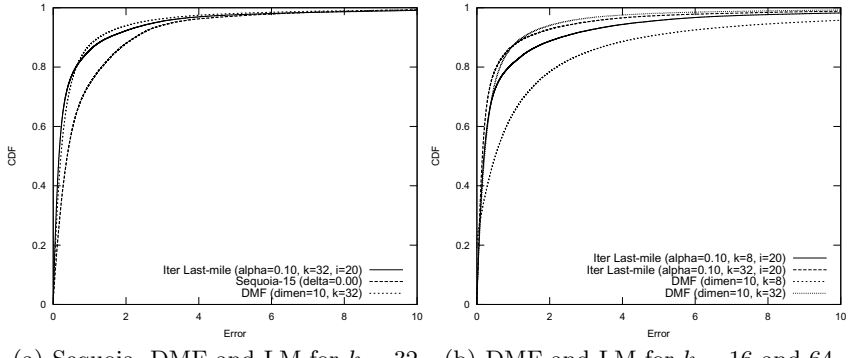


Fig. 3. CDFs of modified relative error: 30 runs of Sequoia and PathGuru; last-mile for different values of k; DMF for different values of k



(a) Sequoia, DMF and LM for $k = 32$ (b) DMF and LM for $k = 16$ and 64

Fig. 4. Direct comparison of modified relative error

We can also see that while DMF is able to make a better use of a larger number of measurements, last-mile achieves a reasonably good accuracy even for low values of k. Actually, increasing k does not increase much the accuracy of the predictions of last-mile, but it makes them more stable. In particular, last-mile with 16 neighbours per node is about as accurate as DMF with 32 neighbours per node, which is the default value proposed by the authors of DMF [13] for latency estimation. It is worth pointing out that measuring available bandwidth incurs a larger overhead than measuring latency; hence, being able to use a smaller number of measurements is an attractive feature.

These results show that the last-mile model is able to explain a large part of the structure of the available bandwidth on the Internet, with a very low number of parameters (each node is characterised by only 2 values, to be compared with 20 for DMF with 10 dimensions) and accessing a small number of measurements. The last-mile model is thus a promising approach for the prediction of available bandwidth on the Internet.

5 Concluding Remarks

Estimating the available bandwidth between nodes in a large scale distributed platform is a crucial issue in many distributed applications. On the other hand, it is impossible to rely on complete measurement sets, because of the intrinsic cost of these measurements, and because many measures may be inaccurate due to varying external conditions. Therefore, as it has been done successfully for latency estimations, several labeling schemes have been proposed, such as Sequoia and PathGuru, that enable to predict at low cost the bandwidth between any pair of hosts.

In this paper, we propose simple decentralised heuristics to use the last-mile model as a prediction mechanism for available bandwidth, by characterising each node by an incoming and an outgoing capacity. Based on real-world PlanetLab bandwidth measurements, we show that this model, although simple, achieves better prediction accuracy than the current available solutions, in particular when the number of available measurements is low. The prediction results of PathGuru depend heavily on the choice of landmarks, and Sequoia suffers from its inability to provide asymmetric predictions. When more measurements are available, decentralised matrix factorisation provides more precise predictions than our last-mile heuristic, probably because each node is described with a larger number of parameters.

In the future work, we are planning to investigate the possibility to increase the number of parameters in the last-mile model for a better accuracy, and also to make a combined use of latency and available bandwidth measurements in order to improve the predictions of the model.

References

1. Beaumont, O., Bonichon, N., Eyraud-Dubois, L.: Scheduling divisibleworkloads on heterogeneous platforms under bounded multi-port model. In: IEEE IPDPS 2008, pp. 1–7 (April 2008)
2. Beaumont, O., Eyraud-Dubois, L., Agrawal, S.K.: Broadcasting on large scale heterogeneous platforms under the bounded multi-port model. In: IEEE IPDPS 2010, pp. 1–10 (April 2010)
3. Bonald, T., Massoulié, L., Mathieu, F., Perino, D., Twigg, A.: Epidemic live streaming: optimal performance trade-offs. ACM SIGMETRICS Perform. Eval. Rev. 36, 325–336 (2008)
4. Boufkhad, Y., Mathieu, F., de Montgolfier, F., Perino, D., Viennot, L.: An upload bandwidth threshold for peer-to-peer video-on-demand scalability. In: IEEE IPDPS 2009, pp. 1–10 (May 2009)
5. Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., Bowman, M.: Planetlab: an overlay testbed for broad-coverage services. ACM SIGCOMM Comput. Commun. Rev. 33, 3–12 (2003)
6. Dabek, F., Cox, R., Kaashoek, F., Morris, R.: Vivaldi: a decentralized network coordinate system. In: ACM SIGCOMM 2004, Portland, OR, USA, pp. 15–26 (September 2004)
7. Dinitz, M.: Online, dynamic, and distributed embeddings of approximate ultrametrics. In: Taubenfeld, G. (ed.) DISC 2008. LNCS, vol. 5218, pp. 152–166. Springer, Heidelberg (2008)
8. Dischinger, M., Haeberlen, A., Gummadi, K.P., Saroiu, S.: Characterizing residential broadband networks. In: IMC 2007, San Diego, CA, USA, pp. 43–56 (October 2007)
9. Goldoni, E., Schivi, M.: End-to-end available bandwidth estimation tools, an experimental comparison. In: Ricciato, F., Mellia, M., Biersack, E. (eds.) TMA 2010. LNCS, vol. 6003, pp. 171–182. Springer, Heidelberg (2010)
10. Hu, N., Steenkiste, P.: Exploiting internet route sharing for large scale available bandwidth estimation. In: IMC 2005, pp. 16–16 (October 2005)
11. Ledlie, J., Gardner, P., Seltzer, M.: Network coordinates in the wild. In: USENIX NSDI 2007, pp. 299–311 (April 2007)
12. Li, N.H., Li, L.E., Mao, Z.M., Steenkiste, P., Wang, J.: A measurement study of internet bottlenecks. In: IEEE INFOCOM 2005 (March 2005)
13. Liao, Y., Geurts, P., Leduc, G.: Network distance prediction based on decentralized matrix factorization. In: IFIP NETWORKING 2010, pp. 15–26 (May 2010)
14. Liu, S., Zhang-Shen, R., Jiang, W., Rexford, J., Chiang, M.: Performance bounds for peer-assisted live streaming. ACM SIGMETRICS Perform. Eval. Rev. 36, 313–324 (2008)
15. Mao, L.K.Y., Saul, Smith, J.M.: Ides: An internet distance estimation service for large networks. IEEE JSAC 24(12), 2273 (2006)
16. E., T.S., Ng, H.Z.: Predicting internet network distance with coordinates-based approaches. In: IEEE INFOCOM 2002, pp. 170–179 (June 2002)
17. Ramasubramanian, V., Malkhi, D., Kuhn, F., Balakrishnan, M., Gupta, A., Akella, A.: On the treeness of internet latency and bandwidth. In: ACM SIGMETRICS 2009, Seattle, WA, USA, pp. 61–72 (June 2009)

18. Suh, K., Diot, C., Kurose, J., Massoulie, L., Neumann, C., Towsley, D., Varvello, M.: Push-to-peer video-on-demand system: Design and evaluation. IEEE JSAC 25(9), 1706–1716 (2007)
19. Xing, C., Chen, M., Yan, L.: Predicting available bandwidth of internet path with ultra metric space-based approaches. In: IEEE GLOBECOM 2009 (December2009)
20. Yalagandula, P., Sharma, P., Banerjee, S., Basu, S., Lee, S.-J.: S3: a scalable sensing service for monitoring large networked systems. In: ACM SIGCOMM Workshop on Internet Network Management, Pisa, Italy, pp. 71–76 (September 2006)