Eurasian Journal of Analytical Chemistry
ISSN: 1306-3057 OPEN ACCESS 2017 12 (4): 149-160

Peer-to-Peer and Distributed System Measurement Request Patterns

G. Michael, R. Kavitha

Received: 08 December 2016 ▪ Revised: 11 January 2017 ▪ Accepted: 10 February 2017

Abstract: Measurement service has knowledge of a larger number of network measurements than individual applications, it is in a position to determine when inference can be used to reduce the total number of measurements required to satisfy a particular demand from applications. To accomplish this, the service must quantify the measurement load required to operate a network inference mechanism, and compare this load to that of direct measurement of the requested properties. In this paper, we predict the network traffic injected by inference mechanisms, and use this knowledge to replace direct measurement traffic by inference when the cost of direct measurement exceeds that of inference. After setting up an inference mechanism, continuing measurements typically require O log(n) probes to estimate the properties of $O(n^2)$ paths.

Keywords: Measurement Request Patterns, Distributed System, Peer-to-Peer, Sensing Service.

INTRODUCTION

An important class of network inference mechanisms estimates the properties of a large number of end-to-end network paths by measuring some subset thereof. This class of mechanisms is designed to reduce the amount of injected active measurement probe traffic and the effort required to collect a large number of measurements, typically at the expense of measurement accuracy. For example, the Azureus BitTorrent client can use inferred network delay information to select peers from which to transfer data. The question of how much of a reduction in measurements the existing inference mechanisms achieve for different measurement request patterns has not been adequately studied.

A network measurement service, which provides measurement results to applications on request, is uniquely suited for utilizing network inference mechanisms. Examples of network measurement services include Script Route, the Scalable Sensing Service, iPlane, and the system by Calyam et al. Because a measurement service has knowledge of a larger number of network measurements had individual applications, it is in a position to determine when inference can be used to reduce the total number of measurements required to satisfy a particular demand from applications. To accomplish this, the service must quantify he measurement load required to set up and operate a network inference mechanism, and compare this load to that introduced by direct measurement of requested properties.

In this paper, we predict the network traffic injected by inference mechanisms, and compare it to the traffic injected by requested direct measurements. we present an efficient method for identifying opportunities when inference induces traffic than a given pattern of direct measurements. Method is not an inference mechanism, but a tool or deploying existing inference mechanisms dynamically in the hosts where their use is advantageous. We note that setting up an inference mechanism may incur a non-eligible cost. After startup, continuing measurements typically require O (n) probes to estimate properties of (n^2) paths. Our work hinges on the two observations that (1) there is a hidden constant for the O(n) robes, which can belarge, and that (2) oftentimes, not all he $O(n^2)$ path properties are requested.

This paper discuss measurements, it generally mean active network measurements that require injecting probe packets into the network, e.g., ping packets to measure end-to-end delay. A measurement service is a network service that accepts requests for the results of active measurements from applications on-demand, schedules measurement tools to be run to service these requests, and ultimately

G. Michael, Associate Professor, Department of Computer Science and Engineering, BIST, BIHER, Bharath Institute of Higher Education & Research, Selaiyur, Chennai. E-mail: michaelcse@gmail.com

R. Kavitha, Associate Professor, Department of Computer Science and Engineering, BIST, BIHER, Bharath Institute of Higher Education & Research, Selaiyur, Chennai.

returns the results from the tools to the application. A measurement host, or simply host, is an infrastructure host in the measurement service that invokes measurement tools and records their output, to be delivered to applications. Measurement endpoints are the hosts which source and/or sink traffic in performing a given measurement. The inference mechanisms consider can be broadly divided into two categories. The first category includes mechanisms that use knowledge of link-level or autonomous system (AS)-level paths to choose a subset of paths to be probed. Call these path-based inference mechanisms and term these mechanisms reference-based inference mechanisms. Mechanisms in this category include Vivaldi, and Theilmann et al.'s Dynamic Distance Maps. Our primary focus in this paper is on mechanisms in this latter category. Identify the pattern of direct measurement workload required to equal or exceed the network load of the inference mechanism, and specify advantageous replacements of direct measurements.

The reference-based inference mechanisms, many require a number of measurements that scales linearly with the number of hosts participating in the inference. Some mechanisms perform a constant number of measurements per host. Others perform varying numbers of measurements per host, but average a constant number per host. In this latter group, typically the majority of hosts participate in a constant number of measurements, and a constant number of hosts (such as the so-called landmarks from [8]) participate in a large number of measurements (linear in the number of hosts). Some of the algorithms have an additional cost for initial construction, which we will not consider in this paper. Assume that let us identify or approximate the constant in a network inference algorithm that requires, on average, a constant number of network measurements per host to infer all-pairs measurements among participating hosts.

Call this constant k, call the number of hosts participating in the inference n, and call such an inference mechanism a kn-cost inference. The authors of the delay inference mechanism, for example, recommend that hosts take measurements to 15 landmarks, and the authors of the Vivaldi delay inference mechanism recommend a selection of 32 neighbors [9]. For these two systems under their recommended configurations, k would be 15 and 32, respectively. Observe that the constant k is dictated by the workings of the inference mechanism under consideration, and is not a tunable parameter in our work. Given a set of measurements requested from a measurement service and the constant k, can determine the tipping point at which the total number of measurements requested becomes greater than or equal to the number of measurements required to perform inference. In this case, inference can reduce the total load on the network, only use inference when the total number of requested measurements exceeds kn, miss key opportunities when inference is beneficial. This is because some hosts may be participating in measurements to a large number of hosts, while others may be involved in very few measurements. Consider a situation where n hosts are interested in performing delay measurement to at least one endpoint.

Assume that m out of the n hosts are performing a complete all-pairs measurement mesh, where each of the m hosts measures delay between it and the other m rk, at the cost of reduced accuracy. Some of the algorithms have an additional cost for initial construction, which is not considered in this paper. Assume that let us identify or approximate the constant in a network inference algorithm that requires, on average, a constant number of network measurements per host to infer all-pairs measurements among participating hosts.

Identify the pattern of direct measurement workload required to equal or exceed the network load of the inference mechanism, and specify advantageous replacements of direct measurements. Represent each host requesting measurement as a node in a graph. A requested measurement between two hosts is represented as an undirected edge in that graph. When k = 3, this graph (which has 32 edges) superficially appears to see no benefit from inference, as 32 < 3n = 36. However, by performing inference among the eight nodes marked in black, we reduce the number of measurements taken to 3 8 + 4 = 28, realizing savings of four measurements. If, given a set of requested measurements, we wish to determine whether or not inference can save effort over any subset of the participating hosts, we have to answer a slightly different question. For any subset of hosts of size Graph benefiting it from partial-graph inference when k = 3. n performing measurements among themselves, if the total number of measurements being performed is greater than kn, then a kn-cost inference mechanism requires fewer total measurements than direct measurement. Using our measurement request graph above, determining whether inference can reduce the total number of measurements is a matter of finding subgraphs for which the number of measurements within each subgraph is greater than k multiplied by the number of nodes in the subgraph. Replacing the direct measurements in these subgraphs with inference will yield a smaller total number of

measurements performed. In other words, given a graph G = (V;E), our goal is to transform it into another graph G0 = (V;E0) such that we minimize jE0 j, where 0 jE0 j jEj and 0 jE0 j kjV j.

The only transformation operations allowed on G are replacement of all edges among subsets Vi of V by kjV ij edges (i.e., employing one of the inference mechanisms in the literature on subsets Vi of vertices, while using direct measurements for remaining edges). Note that a vertex in one of the Vi subsets can still be an endpoint in a direct measurement, as long as the other endpoint does not belong to a subset Vi.

TERMINOLOGY

A. UUSEE Peer to Peer Streaming

Backed by venture capital funding from recognized investors, UUSee Inc.is one of the leading peer to peer streaming solution providers in mainland China, featuring exclusive contractual rights to most of the channels of , the official Chinese television network. With a large collection of streaming servers around the world, it simultaneously broadcasts over 800 channels to millions of peers, mostly encoded to high quality streams around 400 Kbps. Similar to all current-generation streaming protocols, UUSee's streaming protocol design is based on the principle of allowing peers to serve each other by exchanging blocks of data in a sliding window of the media channel. After a new peer joins a channel in UUSee, the initial set of a number of partners (up to 50) is supplied by one of its tracking servers. The peer establishes connections with these partners, and buffer maps are exchanged periodically. During this process, it measures the round-trip delay and throughput of the connection, and then selects a number of most suitable peers (around 30) from which it actually requests media blocks.

In addition, the UUSee peer selection protocol incorporates a number of strategies to maximally utilize peer upload bandwidth. Each peer estimates its maximum upload capacity, and continuously monitors its aggregate instantaneous sending throughput to its partners during streaming. If its aggregate sending throughput is lower than its upload capacity for a sustained period of time, it will inform one of the tracking servers that it is able to receive new connections. Each tracking server keeps a list of such peers, and bootstraps new peers with peers randomly selected from this set. During streaming, neighboring peers also recommend known partners to each other, based on estimated availability for them to assist each other. As a last resort, a peer will contact a tracking server again to obtain additional partners, if its playback rate is not sustained for a certain period of time.

B. Server Front-End

Each Script route server runs an ordinary Web server on port 3355, which provides a gateway for script submission and administrative tasks. There are three main "pages" on the server: job submission, trace back, and informational. The job submission page provides an interface for measurement script submission, then replies with the output of the measurement. Again, the handshake demonstrates that the source IP address is valid to provide a measure of accountability. A convenient feature of the httpd is that it limits the execution time, size, and output of the script. We also limit the number of concurrent requests per client (1) and the number of concurrent requests overall (10).

If the interpreter fails due to resource limits, the connection is closed signaling an error to the client. Unhandled exceptions in the measurement script itself are handled by the interpreter and returned to the client as text. In addition, the UUSee peer selection protocol incorporates a number of strategies to maximally utilize peer upload bandwidth. Each peer estimates its maximum upload capacity, and continuously monitors its aggregate instantaneous sending throughput to its partners during streaming. The following algorithm explains about the routing technique:

```
#! /usr/local/bin/srinterpreter
probe=ScriptRoute::Udp.new(12)
probe.ip_dat=ARGV[0]
unreach=false
puts "(TraceRoute to #{ARGV[0]} {#probe,ip_dst})"
catch(!unreachable) do
(1..64).each {|tt1|
 (1..3).each {|rep|
 probe.ip_tt1=tt1
packets=ScriptRoute::send_train()([Struct::DelayPacket.new(0,probe)])
 if(response) then
 puts '%d %s %5.3f ms' % {tt1,response.ip_src,(packet[0].rtt*1000.0)}
      if(response.is_a?ScriptRoute::ICMP))then
```

```
        unreach=true if(response.icmp_type==ScriptRoute::ICMP_UNREACH)
       end else puts tt1.to_s + '*' end $stdout.flush }
  throw :unreachable if(unreach) }
end
```

Specifically, it provides the tcp dump formatted packets sent to particular IP addresses along with the address of the corresponding client. Finally, the informational page provides information about the measurement traffic supported, how to contact the administrator of the server, how to learn more about Scriptroute, and how to add destination filters to block unwanted measurement traffic. So that administrators know where to look to when their systems receive unexpected measurement traffic, we encourage Scriptroute servers that also have a port 80 Web server to link this page, to direct concerns to the central management site.

C. Scalable Inference Engines

Scalable Inference engines leverage configuration interfaces of sensor pods to perform periodic measurements at the nodes in the system and leverage the scalable sensing backplane to aggregate the measured data. The task of collecting the complete information about network metrics is an immense task both in terms of the infrastructure requirements as well as the measurement traffic. Scalable inference engines estimate complete information about the relevant network metrics based on partial information measured using the sensing pods. The main idea behind the inference algorithms is to measure various metrics on a small number of network paths and use the information to infer the properties of all the paths.

While scalable inference of all network properties is a challenge, a large body of research efforts successfully tackled latency estimation. Though these efforts take different approaches, they all involve periodic measurements from each node to few other nodes in the system to answer for proximity or latency queries accurately reflecting the current status of the network. In S3, inference engines use sensor pods to perform these periodic measurements and the sensing backplane to gather the data in an efficient manner using thresholding and in-network aggregation. Below we briefly describe how Netvigator and its distributed version, a scalable proximity/latency estimation algorithm, can be plugged into our architecture.

Landmark clustering is a popular scheme used for network distance estimation that uses a node's distances to a set of special nodes (referred to as landmark nodes) to estimate the node position. In Netvigator, the sensing pod at each node measures distances to a given set of landmarks, similar to other landmark clustering techniques. Netvigator additionally records the distances to the milestones that are encountered while probing the landmarks. Instead of attempting to embed all the nodes in a global Cartesian-space based on measurements, Netvigator performs local clustering for proximity estimation. Particularly, one scheme Netvigator uses to estimate latency from a node X to another node Y is based on MIN-SUM formulation:

$$\text{latency}(X, Y) = \min_{l \in L,\ \{d(X, l) + d(Y, l)\}},$$

where L is the set of landmarks and milestones and d(X, l) denotes the measured latency from node X to l.

In the S3architecture, distributed Netvigator uses web service interfaces of sensor pods to configure periodic invocations of traceroute sensor from each machine to chosen landmark nodes. These measurements are fed into the sensing backplane and distance information to different landmark nodes or milestones are aggregated along different aggregation trees exposed by the middleware using an aggregation function that tracks Top-k minimum distant nodes from a given landmark or milestone. To answer the proximity queries quickly, nodes subscribe to global aggregate values in the aggregation trees corresponding to their Top K nearest landmarks or milestones. Nodes use the publish subscribe feature of the sensing backplane to filter out most of the changes in the latency values that does not affect their proximity information.

D. Prototype and Deployment

Built a prototype of the S3 modules that is deployed on the Planet-Lab test bed. Currently, deploy and ensure the liveness of our service on Planet-Lab nodes using vxargs script run from a central manager. In near future, we plan to switch to one of the distributed frameworks like AppManager to deploy and run our service. Sensor pods are implemented as cgi scripts accessible through any web-server that supports cgi. use a light-weight open source web server.

This framework enables third party measurements, that is, measurements between two nodes can be initiated by a third node. Our current implementation has a wide variety of sensors, some of which are listed, that leverage several open source network monitoring tools for measuring various network path metrics (latency, number of hops, available bandwidth, bottleneck capacity, and loss rate). To enable large scale concurrent measurements, had to modify some of the tools. We are currently measuring all-pair network metrics periodically. Leverage the web-services based sensing pod architecture to deploy various sensors measuring different metrics and also to configure the periodic measurements. Show an example of accessing a web-enabled sensing-pod deployed on the Planet lab. The sensing backplane is not yet integrated with the sensing pods or analysis engines.

Pull the measurement data from the sensor pods on all nodes to a central node to provide the global views to other researchers by making this data available online, and also to archive the data for Internet behavior analysis. Provide estimated latencies between all planet lab nodes as estimated by Netvigator. For every snapshot of data collected, compute the estimation error over a small number of paths for which we have the actual measured latency. Plots the estimated delay versus the actual measured delay for Netvigator for a single snapshot of Planet-Lab data. The units for the axes in these plots are in microseconds. In these scatter plots, the closer the points plotted are to the diagonal, the better is the estimation.

In the Planet- lab experiments, the delay estimation with Netvigator was the best, with a mean absolute estimation error of 23 msec, followed by Vivaldi and GNP in this order. They present Navigator latency estimation results on the snapshots generated over a 7-day period between March 23 15:49:37 PST 2006 and March 30 19:12:05 PST 2006 and compute various statistics of the absolute estimation error. These statistics are the mean, the 25th, 50th, 75th and 90th percentile of the absolute error. The main observation is that the 25th, 50th and 75th percentile absolute error is fairly low and stable across the entire time period, with 75% of the measured latency having estimation error less than 25 msec [10].

E. Mapping the Internet Topology

IPlane requires geographically distributed vantage points to map the Internet topology and obtain a collection of observed paths. Planet Lab servers, located at over 300 sites around the world, serve as the primary vantage points. We also enlist the use of public Looking Glass/Trace route servers for low-intensity probing. Further, we are currently exploring the option of using data from, a system for aggregating low intensity measurements from normal PCs. Our primary tool for determining the Internet topology is trace route, which allows us to identify the network interfaces on the forward path from the probing entity to the destination. Determining what destinations to probe and how to convert the raw output of trace route to a structured topology is nontrivial, an issue we address next.

F. Probe Target Selection

BGP snapshots, such as those collected by Route Views, are a good source of probe targets. iPlane achieves wide coverage for the topology mapping process by obtaining the list of all globally routable prefixes in snapshots, and choosing within each prefix a target one address that responds to either probes. Address is typically a router and is hence more likely to respond to probes than arbitrary end-hosts. To reduce measurement load, iPlane clusters internet protocol prefixes into atoms for generating the target list. A atom is a set of prefixes, each of which has the same AS path to it from any given vantage point. atoms can be regarded as representing the knee of the curve with respect to measurement efficiency—probing within an atom might find new routes, but it is less likely to do so.

This task of determining a representative set of IP addresses is performed relatively infrequently, typically once every two weeks. iPlane uses the Planet Lab nodes to perform exhaustive and periodic probing of the representative targets. In addition, iPlane schedules probes from public trace route servers to a small random set of atoms, typically making a few tens of measurements during the course of a day. The public trace route servers serve as a valuable source of information regarding local routing policies. Note that in the long run, a functioning iPlane may actually serve to decrease the load on the public trace route servers as iPlane, rather than the trace route servers themselves, can be consulted for information on the Internet topology.

G. Measurement of Link Attributes

Next outline the details of the loss rate, bottleneck capacity and available bandwidth measurements performed from each vantage point. Previous research efforts have proposed septic ways to measure each of these properties; our goal is to integrate these techniques into a useful prediction system.

Latencies of path segments can be derived directly from the trace route data gathered while mapping the topology, and therefore do not need to be measured explicitly.

H. Loss Rate Measurements

Perform loss rate measurements along path segments from vantage points to routers in the core by sending out probes and determining the fraction of probes for which we get responses. We currently use the simple method of sending limited singleton probes with a 1000-byte payload. When the probe's value expires at the target router, it responds with a error message, typically with a small payload. When a response is not received, one cannot determine whether the probe or the response was lost, but there is some evidence from previous studies that small packets are more likely to be preserved even when routers are congested. We therefore currently attribute all of the packet loss to the forward path; the development of more accurate techniques is part of ongoing work.

I. Capacity Measurements

Perform capacity measurements using algorithms initially proposed by that vary the packet size and determine the delay induced by increased packet sizes. For each packet size, a number of probes (typically 30–40) of that size are sent to an intermediate router and the minimum round-trip time is noted. The minimum round-trip time observed over many probes can be regarded as a baseline path latency measurement with minimal queuing delays. By performing this experiment for different packet sizes, one can determine the increased transmission cost per byte. When this experiment is performed for a sequence of network links in succession, the capacity of each link can be determined. Note that our capacity measurements may underestimate a cluster link if it consists of multiple parallel physical links.

J. Available Bandwidth Measurements

After obtaining link capacities, try to probe for available bandwidth along path segments using packet dispersion techniques such as Spruce. A simple measurement is performed by sending a few, equally spaced, short probes at the believed bottleneck capacity of the path segment, and then measuring how much delay they induce. The slope of the delay increase will indicate how much background traffic arrived during the same time period as the probe. For instance, if the probes are generated with a gap of Δin through a path segment of capacity C and if the measured gap between between the probe replies is Δout, one can estimate the available bandwidth as C · (1− Δout−Δin Δin). it is possible to realize the desired scheduling most of the time.

Algorithm

A. Algorithm

Our algorithm for determining the cores hierarchy is based on the following property If from a given graph G = (V; L) we recursively delete all vertices, and lines incident with them, of degree less than k, the remaining graph is the k-core. The outline of the algorithm is as follows:

```
INPUT: graph G = (V; L) represented by lists of neighbors'
OUTPUT: table core with core number for each vertex
1.1 compute the degrees of vertices;
1.2 order the set of vertices V in increasing order of their degrees;
2 for each v 2 V in the order do begin
    2.1 core[v] := degree[v];
    2.2 for each u 2 Neighbours(v) do
                    2.2.1 if degree[u] > degree[v] then begin
                        2.2.1.1 degree[u] := degree[u] 1;
                        2.2.1.2 reorder V accordingly
                                    end
                                    end;
```

The block of statements describes the effect of deletion of the vertex v and all lines incident with it. Note that the order used in the line 2 is changed at each step by the line 2.2.1.2. In the measurements of the algorithm we have to provide efficient implementations of steps.

B. Detailed Algorithms

In the Algorithm describe an implementation of the algorithm in a Pascal like language for the case of simple undirected graph G = (V;E), E is the set of edges. The structure graph is used to represent a given graph G = (V; L). Do not describe the structure into details, because there are several possibilities, how to

implement it. Assume that the vertices of G are numbered from 1 to n. The user has also to provide function size, which returns the number of vertices in the given graph, and function in Neighbours, which returns the next not yet visited neighbour of a given vertex in the given graph. Using an adequate representation of graph G (lists of neighbours) can implement both functions to run in a constant time.

Two different types of integer arrays (tableVert and tableDeg) are also introduced. Both of them are of length n. The only difference is how we index their elements. Start with index 1 in tableVert and with index 0 in tableDeg. The algorithm is implemented by the procedure cores. Its input is a graph G, represented by the variable g of type graph; the output is array deg of type tableVert containing the core number for each vertex of graph G. also need (04-07) some integer variables and three additional arrays. The array vert contains the set of vertices, sorted by their degrees. The positions of vertices in array vert are stored in array pos. The array bin contains for each possible degree the position of the rst vertex of that degree in array vert.

In a real implementation of the proposed algorithm dynamically allocated arrays should be used. To simplify our description of the algorithm replaced them by static.At the beginning we have to initialise some local variables and arrays (09-15). First determine n, the number of vertices of graph g. Then compute its degree for each vertex v in the graph g and store it into the array deg. simultaneously also compute the maximum degree md. The following algorithm explains it briefly:

```
01  procedure cores(var g: graph;
02  var deg: tableVert);
03  var
04  n, d, md, i, start, num: integer;// Declaration of variables
05  v, u, w, du, pu, pw: integer;
06  vert, pos: tableVert;// Designing the table vertex
07  bin: tableDeg;
08  begin
09  n := size(g); md := 0;
10  for v := 1 to n do begin
11  d := 0;
12  for u in Neighbours(v) do inc(d);// Identifying the neighbor nodes
13  deg[v] := d;
14  if d > md then md := d;
15  end;
                16  for d := 0 to md do bin[d] := 0;// Creating network for our own design
17  for v := 1 to n do inc(bin[deg[v]]);
18  start := 1;
19  for d := 0 to md do begin// Designing the node for varying sizes
20  num := bin[d];
21  bin[d] := start;
22  inc(start, num);
23  end;
24  for v := 1 to n do begin//fixing the position of the node
25  pos[v] := bin[deg[v]];
26  vert[pos[v]] := v;
27  inc(bin[deg[v]]);
28  end;
29  for d := md downto 1 do
30  bin[d] := bin[d-1];
31  bin[0] := 1;
32  for i := 1 to n do begin
33  v := vert[i];
34  for u in Neighbours(v) do begin
35  if deg[u] > deg[v] then begin// Positioning the vertices in the network
36  du := deg[u];
37  pu := pos[u];
38  pw := bin[du];
39  w := vert[pw];
40  if u <> w then begin
41  pos[u] := pw;// Identifying the position
```

```
42  pos[w] := pu;
43  vert[pu] := w;// Assigning vertices in the path where ever needed
44  vert[pw] := u;
45  end;
46  inc(bin[du]);// Incrementation of bin
47  dec(deg[u]);// Decrement the array value
48   end;
49  end;
50  end;
51  end;
```

Since the values of degrees are integers from the interval 0 .. n

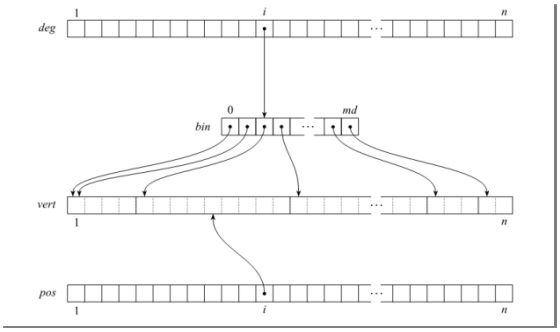

Fig. 1: Hierarchical Structure of bin

previous bin. To avoid an additional array we used the same array (bin) to store the starting positions of bins. Now we can put (24-28) vertices of the graph G into the array vert. For each vertex we know to which bin it belongs and what is the starting position of that bin. So we can put the current vertex to the proper place, remember its position in the table pos, and increase the starting position of the bin we used. The vertices are now sorted by their degrees. In the nal step of the initialisation phase we have to recover the starting positions of the bins (29-31). We increased them several times in previous step, when we put vertices into corresponding bins. It is obvious, that the changed starting position is the original starting position of the next bin. To restore the right starting positions we have to shift the values in array bin for one position to the right. We also have to reset the starting position of the bin 0 to value 1.

The cores decomposition, implementing the for each loop from the algorithm described above, is done in the main loop (32-50) that runs over all vertices v of the graph g in the order, determined by the table vert. The core number of the current Vertex v is the current degree of that vertex. This number is already stored in table deg. For each neighbour u of vertex v with higher degree we have to decrease its degree by 1 and move it for one bin to the left. Moving vertex u for one bin to the left is an operation that can be done in a constant time. First we have to swap the vertex u and the rst vertex in the same bin. We also have to swap their positions in the array pos. Finally we increase the starting position of the bin (we increase the previous and reduce the current bin for one element).

C. Time Complexity

Show that the described algorithm runs in time O(max(m; n)). To compute (09-15) the degrees of all vertices we need time O(max(m; n)) since we have to consider each line at most twice. The bin sort (16-31) consists of ve loops of size at most n with constant time O(1) bodies therefore it runs in time O(n). The statement (33) requires a constant time and therefore contributes O(n) to the algorithm. The conditional statement (35-48) also runs in constant time. Since it is executed for each edge of G at most twice the contribution of (34-49) in all repetitions of (32-50) is O(max(m; n)). Summing up | the total time complexity of the algorithm is O(max(m; n)). Note that in a connected network m n.

D. Adaption of the Algorithm for Directed Graphs

For directed simple graphs without loops only few changes in the implementation of the algorithm are needed depending on the interpretation of the degree. In the case of in degree (out-degree) the function in Neighbors in line 12 must return the next not yet visited in-neighbor (out-neighbor), and the function in Neighbors in line 34 must return the next not yet visited out-neighbor (in-neighbor).If the degree is defined as in-degree + out-degree, the maximum degree can be at most 2n. Basic approach for finding the k smallest spanning trees We first find a minimum spanning tree of our graph, using the fast algorithm of for general graphs or for planar graphs. Then we use Eppstein's technique to reduce the problem to one in which there are O(E) vertices and edges [E].

This uses an algorithm for the sensitivity analysis of minimum spanning trees [Tl],[T2],[T3], and a linear-time selection algorithm. Let Ti denote the i-th smallest spanning tree of the graph. Having found 7'1, we generate the k - 1 spanning trees T2, . . - , Tk one at a time. Each tree Ti with i > 1 will be derived from some tree Tg, j < i, by a swap (ei, fi), in which a tree edge e is replaced by a nontree edge fi. So that no tree is derived more than once, we use an inclusion-exclusion approach presented.

Associated with each Ti will be a best-swap structure R,, which we will implement in the next section as an ambivalent data structure. We mention a few of its properties now. Structure R.j will represent all spanning trees derivable from Ti by a sequence of swaps, and will identify a minimum-cost swap for Ti. The algorithm will maintain a heap on the costs of the trees After setting up RI, the algorithm will perform 2(k - 1) updates of best-swap structures, along with k - 1 eztractmins and 2(k - 1) inserts. The time for all heap operations can be reduced from O(h1og k) to O(k) by using the algorithm in [F2] to select the k-th smallest value in a min-heap. Correct the costs of spanning trees derived n exclusion process form a min-heap.

E. Dynamic 2-edge-connectivity in Embedded Planar Graphs

Use the edge-ordered topology tree as in section 3 as a basis for our data structure for maintaining 2-edge-connectivity information in embedded planar graphs. In addition, use the partial and complete paths as in section .As in section we shall maintain a modified path m(Pj) for each partial path Pj, but will maintain different information in the corresponding edge-ordering tree. Between two consecutive vertices 2 and y of m(4) we shall have an edge with a value bcinit. The value bcinit will be 0 if 2 and y represent different vertices and 1 otherwise. Each leaf in an edge-ordering tree will contain the bcinit values of the path edges to the left and right of the path vertex corresponding to the leaf. In each node of the edge-ordering tree there will be a field project. This will either be null or the name of a vertex on Pj. For an edge in the boundary set with endpoint U, proj(j, U) will be the first non-null project value on the path from the root to the leaf representing the edge in the edge-ordering tree.

Also maintain a field distproj in each node such that the sum of the distproj values of nodes on a path from the root to down to a node containing a non-null project value, where all ancestors have null project values, is the distance on the partial path from proj(j,u) to one end of the partial path. This information is used to identify best covering edges when two clusters 51 and 511 are unioned to give cluster. For example, we identify the edge (if any) in the boundary set(s) of 41 whose endpoint U is such that proj(j, U) is as far as possible from ∼II on P, and adjust the somehow and alcove values of O(1og n) nodes in the balanced tree for the path accordingly.

SOLUTION AND ANALYSIS

We experimentally evaluate the algorithms given in Section 3.1, Section 3.2, on graphs of various structure. As a baseline, we evaluate our methods on a set of simple synthetic topologies, including graphs having uniform random edge placement. Additionally, we utilize graphs based on real datasets from a large-scale peer-to-peer system: the popular UUSee streaming television service [3].These graphs represent a typical scenario where users select peers or servers with which to communicate based on measured network path properties. The key measure of comparison for this study is the amount of reduction in measurements required between hosts in the graph if we employ inference on the subgraphs (components(_)) that our algorithms output and direct measurements on the remaining edges, compared to performing all there quested direct measurements. We also report the cost of performing a single inference on the entire graph. We give the running time of the spanning forest algorithm, and investigate the values of its two key parameters (number of forests f and threshold score s). We also study the relationship between sand the constant k of the inference mechanism.

CONCLUSION

We have investigated the network load induced by inference mechanisms, and presented efficient algorithms to identify subgraphs where replacing direct measurements with inference is most advantageous. Our results show that we achieve significant measurement savings with small-world graphs, which represent popular peer-to-peer and distributed system measurement request patterns. We demonstrate the ability to identify subgraphs of measurement graphs which see no cost benefit from inference, and accordingly use more accurate direct measurements in those subgraphs. We analyze the performance of our algorithms, and make recommendations for the discretionary parameter s provided to the spanning forest and hybrid algorithms based on both theory and empirical results.

REFERENCES

[1] Das, J., Das, M. P., & Velusamy, P. (2013). Sesbania grandiflora leaf extract mediated green synthesis of antibacterial silver nanoparticles against selected human pathogens. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, *104*, 265-270.

[2] Umanath, K.P.S.S.K., Palanikumar, K., & Selvamani, S. T. (2013). Analysis of dry sliding wear behaviour of Al6061/SiC/Al2O3 hybrid metal matrix composites. *Composites Part B: Engineering*, *53*, 159-168.

[3] Udayakumar, R., Khanaa, V., Saravanan, T., & Saritha, G. (1786). Cross layer optimization for wireless network (WIMAX). *Middle-East Journal of Scientific Research*, *16*(12), 1786-1789.

[4] Kumaravel, A., & Rangarajan, K. (2013). Algorithm for automaton specification for exploring dynamic labyrinths. *Indian Journal of Science and Technology*, *6*(5S), 4554-4559.

[5] Pieger, S., Salman, A., & Bidra, A.S. (2014). Clinical outcomes of lithium disilicate single crowns and partial fixed dental prostheses: a systematic review. *The Journal of prosthetic dentistry*, *112*(1), 22-30.

[6] Vijayaraghavan, K., Nalini, S.K., Prakash, N.U., & Madhankumar, D. (2012). One step green synthesis of silver nano/microparticles using extracts of Trachyspermum ammi and Papaver somniferum. *Colloids and Surfaces B: Biointerfaces*, *94*, 114-117.

[7] Khanaa, V., Mohanta, K., & Satheesh, B. (2013). Comparative study of uwb communications over fiber using direct and external modulations. *Indian Journal of Science and Technology*, *6*(6), 4845-4847.

[8] Khanaa, V., Thooyamani, K. P., & Udayakumar, R. (1798). Cognitive radio based network for ISM band real time embedded system. *Middle-East Journal of Scientific Research*, *16*(12), 1798-1800.

[9] Vijayaraghavan, K., Nalini, S.K., Prakash, N.U., & Madhankumar, D. (2012). Biomimetic synthesis of silver nanoparticles by aqueous extract of Syzygium aromaticum. *Materials Letters*, *75*, 33-35

[10] Caroline, M.L., Sankar, R., Indirani, R.M., & Vasudevan, S. (2009). Growth, optical, thermal and dielectric studies of an amino acid organic nonlinear optical material: l-Alanine. *Materials Chemistry and Physics*, *114*(1), 490-494.

[11] Kumaravel, A., & Pradeepa, R. (2013). Efficient molecule reduction for drug design by intelligent search methods. *International Journal of Pharma and Bio Sciences*, *4*(2), B1023-B1029.

[12] Kaviyarasu, K., Manikandan, E., Kennedy, J., Jayachandran, M., Ladchumananandasiivam, R., De Gomes, U. U., & Maaza, M. (2016). Synthesis and characterization studies of NiO nanorods for enhancing solar cell efficiency using photon upconversion materials. *Ceramics International*, *42*(7), 8385-8394.

[13] Sengottuvel, P., Satishkumar, S., & Dinakaran, D. (2013). Optimization of multiple characteristics of EDM parameters based on desirability approach and fuzzy modeling. *Procedia Engineering*, *64*, 1069-1078.

[14] Anbuselvi S., Chellaram, C., Jonesh S., Jayanthi L., & Edward J.K.P. (2009). Bioactive potential of coral associated gastropod, Trochus tentorium of Gulf of Mannar, Southeastern India. *J. Med. Sci,* 9(5), 240-244.

[15] Kaviyarasu, K., Ayeshamariam, A., Manikandan, E., Kennedy, J., Ladchumananandasivam, R., Gomes, U. U., & Maaza, M. (2016). Solution processing of CuSe quantum dots: Photocatalytic activity under RhB for UV and visible-light solar irradiation. *Materials Science and Engineering: B*, *210*, 1-9.

[16] Kumaravel, A., & Udayakumar, R. (2013). Web portal visits patterns predicted by intuitionistic fuzzy approach. *Indian Journal of Science and Technology*, *6*(5S), 4549-4553.

[17] Srinivasan, V., & Saravanan, T. (2013). Reformation and market design of power sector. *Middle-East Journal of Scientific Research*, *16*(12), 1763-1767.

[18] Kaviyarasu, K., Manikandan, E., Kennedy, J., & Maaza, M. (2015). A comparative study on the morphological features of highly ordered MgO: AgO nanocube arrays prepared via a hydrothermal method. *RSC Advances*, *5*(100), 82421-82428.

[19] Kumaravel, A., & Udhayakumarapandian, D. (2013). Consruction of meta classifiers for apple scab infections. *International Journal of Pharma and Bio Sciences*, *4*(4), B1207-B1213.

[20] Sankari, S. L., Masthan, K. M. K., Babu, N. A., Bhattacharjee, T., & Elumalai, M. (2012). Apoptosis in cancer-an update. *Asian Pacific journal of cancer prevention*, *13*(10), 4873-4878

[21] Harish, B. N., & Menezes, G. A. (2011). Antimicrobial resistance in typhoidal salmonellae. *Indian journal of medical microbiology*, *29*(3), 223-229.

[22] Manikandan, A., Manikandan, E., Meenatchi, B., Vadivel, S., Jaganathan, S. K., Ladchumananandasivam, R., & Aanand, J. S. (2017). Rare earth element (REE) lanthanum doped zinc oxide (La: ZnO) nanomaterials: synthesis structural optical and antibacterial studies. *Journal of Alloys and Compounds*, *723*, 1155-1161.

[23] Caroline, M. L., & Vasudevan, S. (2008). Growth and characterization of an organic nonlinear optical material: L-alanine alaninium nitrate. *Materials Letters*, *62*(15), 2245-2248.

[24] Saravanan T., Srinivasan V., Udayakumar R. (2013). A approach for visualization of atherosclerosis in coronary artery, Middle - East Journal of Scientific Research, 18(12), 1713-1717.

[25] Gokula Krishnan, C.A., & Dr. Suphalakshmi, A. (2017). An Improved MAC Address Based Intrusion Detection and Prevention System in MANET Sybil Attacks. *Bonfring International Journal of Research in Communication Engineering, 7*(1), 1-5.

[26] Kurian, S., & Franklin, R.G. (2013). Trustworthy Coordination of Web Services Atomic Transaction for Net Banking. *The SIJ Transactions on Advances in Space Research & Earth Exploration, 1*(1), 6-9.

[27] Dr.Gopinath, B., Kalyanasundaram, M., Karthika, V., & Pradeepa, M. (2018). Development of Power Quality Event Using Diode Clamped Multilevel Inverter in Conjunction with AANF. *Bonfring International Journal of Software Engineering and Soft Computing, 8*(1), 17-22.

[28] Dr. Chaturvedi, A., Bhat, T.A., & Kumar, V. (2013).Movement based Asynchronous Recovery System in Mobile Computing System. *The SIJ Transactions on Computer Networks & Communication Engineering (CNCE), 1*(3), 1-5.

[29] Jerin Jose, M., Akmal Jahan, S., Arunachalam, R., Karnan, R., & Kishore, V. (2017). Automobile Accident Sensing Unit and Notifier using Arduino. *The SIJ Transactions on Industrial, Financial & Business Management (IFBM), 5*(1), 5-8.

[30] Hoa, N.T., & Voznak, M. (2019). High Speed and Reliable Double Edge Triggered D- Flip-Flop for Memory Applications. *Journal of VLSI Circuits and Systems*, *1*(1), 13-17.

[31] Shamim, F.M., & Vishwakarma, S. (2016). Exploiting the Motion Learning Paradigm for Recognizing Human Actions. *Bonfring International Journal of Advances in Image Processing, 6*(3), 11-16.

[32] Kumar, K.A., Sadulla, S., & A. Surendar, (2018). Statistical Analysis of Reliable and Secure Transmission Gate based Arbiter Physical Unclonable Functions (PUFs).*Journal of Computational Information Systems,* 14(3), 62 - 69.

[33] Puliyath, S. (2014). Advanced Secure Scan Design against Scan Based Differential Cryptanalysis. *International Journal of Advances in Engineering and Emerging Technology, 5*(6), 274-279.

[34] Rinesh, S., and Jagadeesan, S. (2014). Detection and Localization of Multiple Spoofing Attackers in Wireless Networks. *Excel International Journal of Technology, Engineering and Management, 1*(1), 17-20.

[35] Poongothai, S., Ilavarasan, R., & Karrunakaran, C.M. (2010). Simultaneous and accurate determination of vitamins B1, B6, B12 and alpha-lipoic acid in multivitamin capsule by reverse-phase high performance liquid chromatographic method. *International Journal of Pharmacy and Pharmaceutical Sciences*, *2*(4), 133-139.

[36] Udayakumar, R., Khanaa, V., & Saravanan, T. (2013). Synthesis and structural characterization of thin films of SnO 2 prepared by spray pyrolysis technique. *Indian Journal of Science and Technology*, *6*(6), 4754-4757

[37] Anbazhagan, R., Satheesh, B., & Gopalakrishnan, K. (2013). Mathematical modeling and simulation of modern cars in the role of stability analysis. *Indian Journal of Science and Technology*, *6*(5S), 4633-4641.

[38] Caroline, M.L., & Vasudevan, S. (2009). Growth and characterization of bis thiourea cadmium iodide: A semiorganic single crystal. *Materials Chemistry and Physics*, *113*(2-3), 670-674.

[39] Sharmila, S., Jeyanthi Rebecca, L., & Das, M. P. (2012). Production of Biodiesel from Chaetomorpha antennina and Gracilaria corticata. *Journal of Chemical and Pharmaceutical Research*, *4*(11), 4870-4874.

[40] Thooyamani, K.P., Khanaa, V., & Udayakumar, R. (2013). An integrated agent system for e-mail coordination using jade. *Indian Journal of Science and Technology*, *6*(6), 4758-4761.

[41] Caroline, M. L., Kandasamy, A., Mohan, R., & Vasudevan, S. (2009). Growth and characterization of dichlorobis l-proline Zn (II): A semiorganic nonlinear optical single crystal. *Journal of Crystal Growth*, *311*(4), 1161-1165.

[42] Caroline, M.L., & Vasudevan, S. (2009). Growth and characterization of L-phenylalanine nitric acid, a new organic nonlinear optical material. *Materials Letters*, *63*(1), 41-44.

[43] Kaviyarasu, K., Xolile Fuku, Genene T. Mola, E. Manikandan, J. Kennedy, and M. Maaza. Photoluminescence of well-aligned ZnO doped CeO2 nanoplatelets by a solvothermal route. *Materials Letters*, *183*(2016), 351-354.

[44] Saravanan, T., & Saritha, G. (2013). Buck converter with a variable number of predictive current distributing method. *Indian Journal of Science and Technology*, *6*(5S), 4583-4588.

[45] Parthasarathy, R., Ilavarasan, R., & Karrunakaran, C. M. (2009). Antidiabetic activity of Thespesia Populnea bark and leaf extract against streptozotocin induced diabetic rats. *International Journal of PharmTech Research*, *1*(4), 1069-1072.

[46] Hanirex, D. K., & Kaliyamurthie, K. P. (2013). Multi-classification approach for detecting thyroid attacks. *International Journal of Pharma and Bio Sciences,* 4(3), B1246-B1251

[47] Kandasamy, A., Mohan, R., Lydia Caroline, M., & Vasudevan, S. (2008). Nucleation kinetics, growth, solubility and dielectric studies of L-proline cadmium chloride monohydrate semi organic nonlinear optical single crystal. *Crystal Research and Technology: Journal of Experimental and Industrial Crystallography*, *43*(2), 186-192.

[48] Srinivasan, V., Saravanan, T., Udayakumar, R., & Saritha, G. (2013). Specific absorption rate in the cell phone user's head. *Middle-East Journal of Scientific Research*, *16*(12), 1748-50.

[49] Udayakumar R., Khanaa V., & Saravanan T. (2013). Chromatic dispersion compensation in optical fiber communication system and its simulation, Indian Journal of Science and Technology, 6(6), 4762-4766.

[50] Vijayaragavan, S.P., Karthik, B., Kiran, T.V.U., & Sundar Raj, M. (1990). Robotic surveillance for patient care in hospitals. *Middle-East Journal of Scientific Research*, *16*(12), 1820-1824.