Efficient and Decentralized Computation of Approximate Global State

S. Keshav
School of Computer Science, University of Waterloo
Waterloo, ON, Canada, N2L 3G1

Abstract— **The need for efficient computation of approximate global state lies at the heart of a wide range of problems in distributed systems. Examples include routing in the Internet, sensor fusion, search in peer-to-peer networks, coordinated intrusion detection, and Top-K queries in stream-oriented databases. Efficient algorithms that determine approximate global state could enable near-optimal local decision-making with little overhead. In this position paper, we model this problem and summarize recent work on randomized algorithms that navigate a four-way tradeoff between accuracy, robustness, performance and overhead. Despite these recent successes, many open problems remain. We believe that solving these problems can radically improve the design of robust, efficient and self-managed distributed systems.**

I. Introduction

The need for approximate global state arises in a wide range of settings:

- *Sensor networks* Consider a system of *N* sensors. Examples of global state that one may want to compute are the average sensor value; extremal sensor values, such as min or max; or quantile values, such as the median [2].
- *Distributed systems* Global state is necessary to implement distributed system primitives such as barrier synchronization, voting, leader election, and consensus [17].
- *Peer to Peer networks* In a hybrid peer-to-peer search system, that combines an unstructured flooding network with a distributed hash table (DHT) [12], an example of global state is the set of items that are popular, i.e., items whose copies exist on many peers. If this information can be efficiently computed, a queries for known-popular items can be flooded instead of being sent to the DHT [19].
- *Stream databases* In a stream-oriented database, such as a distributed database that records the number of hits to an item in a content distribution network, an example of global state is a Top-K query, i.e., the set of K documents that have the most hits.

Position paper presented at the Workshop on Self-Organizing Networks, Seattle, Washington, June 1-2 2005. This research was supported by grants from the National Science and Engineering Council of Canada, the Canada Research Chair Program, Nortel Networks, Sun Microsystems Canada, Intel Corporation, and Sprint Corporation.

- *Internet routing* Global state is needed at every router for Internet routing, because it has to forward incoming packets to the interface that provides the best path to the destination based on the current global topology. In Link State routing, the flooding of Link State Packets results in the global knowledge of this state.
- *Network security* Intrusion detection systems detect anomalies in network usage, such as port scanning. Recent work suggests that collating anomaly information across multiple detectors, a form of global state, may greatly increase the accuracy of the system [18].
- *File systems* In a distributed file system, each node needs to know the set of nodes that store a a given disk block (or file), and this set needs to be updated in response to changes in load, availability guarantees, and node failures [5].

The thread that unites these varied areas is the need to compute approximate global state in a system with a large number of nodes, where computation may be massively distributed, and where the values stored at each node change over time. In this position paper, we model this problem, survey recent work, and outline some open problems. We believe that the solution to these problems can radically improve the design of robust, efficient and self-managed distributed systems. We note that this intuition is also the basis for the Astrolabe system at Cornell [17]; unlike Astrolabe, we wish to address a much broader set of application areas, and seek to move beyond strict hierarchies to more general network topologies.

II. Model

We use the following simple abstraction to model the global computation. Consider a distributed system with N nodes where, at time t, the i^{th} node has local state information s_i^t and knows only of links to its neighbors. In problems of interest, N is very large and nodes arrive, depart, and fail over time. Moreover, communications between nodes may be lost.

Our goal is to have the nodes self-organize to compute a function $F(S^t)$ where $F(.) = \{f_1(.), f_2(.), ..., f_N(.)\}$ and $S^t = \{s_1^t, s_2^t, ..., s_i^t\}$. The subscript on f indicates

that, in general, each node in the network would be computing a different 'view' of the function F. This, for instance, models computation of routing tables, where the result of the computation is a different routing table at each node. Theorem 1 shows that, due to unreliable nodes and links, although F is well-defined, it may not be computable.

Theorem 1: $F(S^t)$ cannot be computed in a distributed system that suffer from node failures and message loss.

$Proof$: The proof is by construction. Consider a two-node system where node 1 and node 2 collaborate to compute $F(S^t)$. This requires that either node 1, or node 2, or some third node needs to obtain s_1^t and s_2^t and compute $F(S^t)$. Without loss of generality, suppose node 1 is chosen to act as the coordinator. Now, if node 2's state changes just before t, the communication of this changed state from node 2 to node 1 is lost, and immediately after communication node 2 also dies, then there is no way for node 1 to compute $F(S^t)$ $\diamond$

We conjecture that F can be correctly computed if either of these two conditions does not hold. For instance, if up to K nodes can fail, but messages are not lost, then a node can update $K+1$ other nodes with its new state every time its state changes. This allows the system to compute $F(S^t)$ despite node failures. Similarly, if messages can be lost but no nodes fail, then peer nodes can use any standard reliable transmission protocol with acknowledgments, timeouts, and retransmissions, to eventually deliver any message, and therefore allow computation of $F(S^t)$.

Theorem 1 shows that, in general, F is not computable. However, for most interesting real-world problems, F, when computed over a sufficiently large fraction of nodes, approximates its true value. More formally, let U^t be the set of nodes whose node states σ^t can be gathered at a coordinator node. For these functions, $F(\sigma^t) - F(S^t) < \epsilon$, where ϵ is the desired error bound. In the rest of this paper, we will only consider such functions F.

III. Taxonomy of the problem space

We now proceed from this abstract model to a taxonomy of the problem according to type of function being computed, underlying network topology, and state change model.

A. Function type

Although the function being computed, such as the one for Internet routing, can be complicated, in the literature, the functions studied can be categorized into one of a few simple types:

• *Extremal values:* these include the minimum and maximum values and Top-K queries. Although conceptually simple, these queries can be powerful: for instance, it turns out computing the min can implement distributed barrier synchronization. [17].

• *Counts,* such as of the total number of nodes, or of the number of nodes that have a particular property. A count query is a first step for many other distributed algorithms, and can be used as a basis for distributed voting, consensus, and leader election.

• *Histograms* of the number of nodes with a particular range of state values. This is a generalization of a count query, with count being done for each histogram bin. Because histograms approximate distributions, this allows us to compute attributes of distributions such as the mean, median, or mode, the cumulative distributions, and higher order moments.

• *Membership:* Maintaining membership in the system or to a user-defined group as a distributed query allows several powerful algorithms such as publish-subscribe, broadcast, and content-aware multicast ('SelectCast') [17].

These examples serve to illustrate the power of decentralized global state discovery. However, this approach has not been applied to some of the areas outlined in the introduction, such as routing, intrusion detection, and BGP policy coordination. Modeling such known problems as functions of approximate global state is an open area of research.

B. Network topology

The cost and performance of a solution depends on the assumed underlying network topology. For instance, with a clique, every neighbor can be reached at unit cost; with a unbalanced tree, the cost can be as large as $O(N)$. In the literature, the computation of approximate global state has usually been studied in the context of overlay networks, whose topology can be made as regular as desired: the tradeoff then is between the mean path length,the robustness of the topology, and the size of the routing table [9]. The following network topologies are relevant:

• *Clique:* In a clique, every node is one hop away from every other node, so all communication has unit cost. In particular, a node can communicate its state to all other nodes in one time step. However, the size of a routing table is $O(N)$ which automatically limits scalability. Much of the literature in this area implicitly assumes that the network is a clique.

• *Random graph of degree k:* Here, every node has k random neighbors. A random graph of degree k only needs a routing table of size k, but path lengths will be longer than with a clique.

• *Tree of degree k:* In a k-ary tree, every node has k children and one parent. A tree with degree k has a routing

table of size $k+1$, but, unlike a random graph, it is fragile with respect to failures. On the other hand, computations the propagate values from leaves to the root finish in $O(log_k N)$ time.

• *Hypercube:* This is a topology where nodes are arranged on the vertices of a hypercube, so that a node is connected to every other node such that the binary representation of their IDs differ in one bit position (if such a node exists). In a hypercube, both the routing table and the mean path length is $O(log_k N)$.

• *Power-law random graphs:* These are approximate models for Internet topology, where the node degree is power-law distributed.

• *Hierarchical power-law graphs:* These graphs, created by tools, such as BRITE [14], are meant to closely approximate Internet topology.

• *Measured Internet topology:* Tools such as Rocketfuel [16] map the actual Internet topology, so it is possible to evaluate the relative performance of various solutions on this graph. Note that all the other topologies can be thought of as regular overlays on Hierarchical PLRGs or measured Internet topologies.

The literature on graph topologies is deep and this list leaves out many interesting topologies such as de Bruijn graphs, rings with 'finger pointers', and butterflies. At this point, it is not clear which topology is 'best' in terms of a tradeoff between routing cost, memory cost, and robustness to failure. The problem is made more complex by realizing that regular overlay topologies, when overlaid on a irregular network, can often result in the same real link being part of many overlay links, causing both congestion and correlated link failures!

C. State change model

The global state being computed may change over time for several reasons. These include:

• *Change in node state* The state value at each node may change over time.

• *Change in number of nodes* Nodes may join, leave, or abruptly fail. Moreover, the failure may be permanent or transient.

• *Change in links* Links may be added, deleted, or fail. Message failure can be modeled as transient link failure.

Clearly, if the rate of change of network state is too rapid, global statistics are stale by the time they can be collected, making them less useful as hints for optimal local decisions. On the other hand, if the network is essentially static, global state needs to be computed only once, which allows the use of complex or time-consuming algorithms, whose cost can be amortized over long durations.

IV. Metrics

Any solution for computation of approximate global state must navigate a tradeoff among the following four quantities:

• *Accuracy* How close is the computed result to the true value of $F(S^t)$? Note that this metric can only be computed in a synthetic setting.

• *Cost* The cost of a solution has three components: the $computation$ cost at each node, the $memory$ cost for storage required by the state discovery algorithm as well as the overlay routing table, and the $communication$ cost, which is the number of bytes exchanged by the nodes to compute F.

• *Performance* The performance of a scheme has two aspects. First, how much time does it take to compute F? State discovery usually involves $rounds$ of computation, and this is therefore expressed in terms of the number of rounds. Delay can be measured either as the average or the worst case time for the computation to complete. Second, what fraction of the nodes present in the system at the time the computation ends compute F correctly (or, to be precise, within a small error bound of the true value of F)?

• *Robustness* How sensitive is the computation to node and link failure and message loss? This quantifies the error in the computed function as a function of the fraction of nodes that fail, or the fraction of messages that are lost.

Some tradeoffs are straightforward: for instance, accuracy for speed, or robustness for cost. Others are not so obvious, for instance, trading accuracy for robustness, by using randomized gossip. Our goal is to compare some well known algorithms with respect to these metrics, and potentially come up with a class of algorithms that are able to achieve every solution in the Pareto frontier.

V. Solution approaches

Several solutions to global state computation have been studied in the database, distributed systems, and networking communities. These approaches fall into the following broad categories.

A. Centralization

A centralized approach, where a designated root node collects s_i^t from all other nodes and computes F, has the best speed and accuracy. In the database community, this approach has been extensively studied for maintenance of materialized views [8]; the emphasis here is on techniques that minimize the cost of incremental view maintenance by exploiting properties of the query as well as the underlying data. The communication cost of this approach depends on the number of updates needed for computation of the

global state, and whether the root is reachable by one-hop paths from all other nodes. If so, then it also has the least cost. If not, state values need to be transferred on multiple hops, which adds to the cost.

Centralization is not scalable because the root node becomes a bottleneck. The solution is not robust, because the loss of the root node causes total failure. However, note that the solution is immune to failures in every node other than the root node. Therefore, in practice, this solution is commonly employed, with resources devoted to adequate protection of the root node.

B. Tree-based solutions

A generalization of the centralized approach consists of inducing a multi-level hierarchy or tree on the underlying graph and having each node send its state to its parent, which performs local aggregation and, in turn, sends the aggregated results upwards, eventually reaching the root. This approach was used by the TAG system [13] and in Astrolabe [17]. Similar to a centralized solution, this solution is fragile, in that the loss of a single node can disrupt the tree. Therefore, care must be taken to maintain backups for tree nodes, and to switch from a tree node to its backup (or re-elect a new tree node) in case of a failure. With in-network aggregation at each node, the communication costs are lower than with a centralized solution. Assuming a balanced tree, the computation time is $O(logN)$.

C. Flooding and Randomized flooding

Flooding is, in a sense, diametrically opposed to a centralized solution. Instead of having a single node $pull$ a single copy of data from every other node, with flooding, a node whose state has changed (an $infective$ node) $pushes$ its data to all or a random subset of its neighbors, who then become infective, and forward this message to some or all their uninfected neighbors and so on (this is called 'rumor spreading' or a 'complex epidemic' in [3]). If care is taken not to forward the same data twice–accomplished, for instance, by using source-specific sequence numbers–flooding requires at most $O(E)$ messages, where E is the number of edges in the network. For reasonable networks, $E = O(NlogN)$, so on surface, flooding appears to be a good idea. However, with naive flooding, there is no $in-network\ aggregation$, so, in practice, it is quite inefficient. Combining flooding with aggregation is more efficient.

D. Random walk-based solutions

A random walk is a style of computation where a node sends its state in a message to a randomly selected neighbor, which uses this message to update its local state, adds its local value to the message's state, and forwards the message to the next randomly chosen neighbor. A random walk message samples and updates state values at the subset of the nodes that it touches. More than one random walk may be in progress in parallel.

Note that with a random walk, a node's state may be sampled more than once. Therefore, it is necessary to somehow prevent 'double counting' [15]. A second issue is that the global statistics can only be computed probabilistically. Typically, we can only make statements such as with probability 1–δ the error in the computed value is less than ϵ.

The advantage of a random walk is that it relatively robust to node failures (as long as care is taken to regenerate a walk that is lost due to a failing node). In a stable network with sufficiently long message length and sufficiently long walks, one can compute any function of global state, but in a class of networks called expander graphs, even short walks are 'good' random samples of local states [6].

E. Randomized gossip based solutions

In each $round$ of computation of random gossip, every node talks to one or more randomly selected neighbors and exchanges some information with it or them (this is called 'anti-entropy' or 'a simple epidemic' in [3]). It turns out that, after approximately $logN$ rounds of computation, all nodes can, with high probability, compute the global state [1, 10]. Just like a random walk, we need to prevent double counting, and we can only make probabilistic statements about the computed values.

There are two subtle differences between a random walk and randomized gossip. FIrst, with standard randomized gossip, every node participates in message exchange, whereas with a random walk, if K walks are ongoing, only K nodes participate in message exchange. Of course, it is possible to device a randomized gossip protocol where only K of N nodes participate in gossiping. Second, two nodes A and B may both choose the same node C as a node with which to exchange state in either algorithm. With random gossip, C's value will be propagated only once to some other node. However, with a random walk, two walks leave C, so C's value will propagate to two other nodes. When the number of random walks is much smaller than N, this is unlikely to happen, and in that situation an network with K random walks look much like network where, in each round of computation, K of N nodes participate in random gossip.

VI. Sketches

The four mechanisms described in the previous section can be characterized as $transport$ mechanisms that move

information around in the network. In this section, we focus on $what$ is being moved instead of how it is moved.

When using a centralized approach, there is no in-network aggregation. However, with the other approaches (tree, flood, random walk, and randomized gossip), partial state is aggregated into $sketches$ to prevent the size of message scaling linearly with N. For instance, if the function that is being computed is min, then the sketch is the smallest of the values being aggregated. Sketches are essentially functions computed over partial state that, over time, converge to the final solution.

It is important to ensure that when a node receives a sketch, if the sketch already includes that node's value, the node should not add the its value to the sketch again (i.e. it should avoid $double\ counting$). This problem does not arise in centralized and tree-based solutions. For flooding, random walk, and randomized gossip-based solutions, we can avoid double counting in one of three ways.

• *Carry node IDs in the sketch* A sketch can carry the set of node IDs that contributed to it. This prevents double counting. However, this makes the sketch size $O(N)$. One can possibly restrict messages to a smaller region, such as an Astrolabe zone [17], but this requires the distributed construction of zones, and some mechanism for inter-zone communication.

• *Use order and duplicate insensitive sketches* This approach uses a special form of sketch (such as a Flajolet-Martin sketch [4]) that is insensitive to duplicates [15]. Essentially, this transforms a measure of central tendency into a measure of extremal values, which is inherently order and duplicate insensitive. However, this conversion often results in a loss of accuracy: an FM sketch can have an error as large as 33%.

• *Use push synopses* In this approach, proposed by Kempe et al [10], each node is associated with a weight, and when it shares its value with another node, it also sheds some of its weight. Using a principle of mass conservation, it can be shown that double counting is avoided. Mass conservation is appealing in theory, but is problematic to maintain in the face of message and node failures.

Each of these techniques has its own pros and cons. Moreover, they can sometimes be used in conjunction with each other. Therefore, determining a good mix of duplicate-suppression techniques for a particular problem is, at this time, more of an art than a science.

VII. Open problems

Given this lay of the land, many open problems remain. These fall into three broad categories. First, many theoretical issues are still open. For instance:

- Are there fundamentally new techniques besides those described above for global state discovery?
- Is there an algorithm that is better than all existing approaches simultaneously on all metrics?
- How do the various solution approaches compare on the chosen metrics for different query types and topologies?

Second, can we map real problems such as network routing and Top-K queries to these theoretically well-studied algorithms? How about distributed intrusion detection, or coordination of BGP policies? We think this is non-trivial.

Third, how should we actually implement a global state discovery in a real system? This requires solutions to problems such as:

- choosing the right transport and aggregation mechanism
- matching the overlay topology with the underlay topology
- removing stale data
- interfacing the new algorithms with legacy systems
- distributed determination of convergence
- punching holes through firewalls and NATs using techniques such as STUNT [7]
- taking advantage of heterogeneity in node connectivity and lifetime
- coping with message and node loss and
- preventing corruption in the computation due to malicious nodes [17].

We believe that finding answers to these issues will go a long way towards creating robust, scaleable, and efficient distributed systems.

VIII. Acknowledgments

I would like to acknowledge Joe Hellerstein for alerting me to this problem area, and to my students Matei Zaharia, Nabeel Ahmed, and David Hadaller for many interesting and insightful discussions.

References

[1] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip Algorithms: Design, Analysis, and Applications," Proc. IEEE INFOCOM 2005, March 2005.

[2] J. Considine, F. Li, G. Kollios, and J. Byers, "Approximate aggregation techniques for sensor databases," Proceedings of the International Conference on Data Engineering, March 2004.

[3] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Stuygis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," PODC, 1987.

[4] P. Flajolet and G.N. Martin, "Probabilistic Counting Algorithms for Database Applications," J. Computer and System Sciences, Vol. 31, 1985.

[5] S. Ghemawat, H. Gobioff, and S.T. Leung, "The Google File System," SOSP03, October 2003.

[6] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks," In Proceedings of IEEE INFOCOM, 2004.

[7] S. Guha, Y. Takeda and P. Francis. "NUTSS: A SIP based approach to UDP and TCP connectivity," in Proceedings of SIGCOMM'04 Workshops, Portland, OR, Aug 2004, pp. 43–48.

[8] A. Gupta and I.S. Mumick "Materialized views: techniques, implementations, and applications," MIT Press, 1999.

[9] K.P. Gummadi, R. Gummadi, S. Ratnasamy, S.D. Gribble, I. Stoica and S. Shenker, "The Impact of DHT Routin Geometry on Resilience and Proximity, " In Proc. ACM SIGCOMM 2003, August 2003.

[10] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-Based Computation of Aggregation Information," Proc. IEEE FOCS, 2003.

[11] D. Kempe and J. Kleinberg, "Protocols and Impossibility Results for Gossip-Based Communication Mechanisms," Proc. FOCS, 2002.

[12] B.T. Loo, R. Huebsch, I. Stoica, and J.M. Hellerstein, "The Case for a Hybrid P2P Search Infrastructure, IPTPS, 2004.

[13] S. Madden, M. J. Franklin,J. M. Hellerstein and W. Hong, "TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks," Proc. OSDI 2002.

[14] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An Approach to Universal Topology Generation," In Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems- MASCOTS '01, Cincinnati, Ohio, August 2001.

[15] S. Nath, P. Gibbons, S. Seshan, and Z. Anderson, "Synopsis Diffusion for Robust Aggregation in Sensor Networks," Proc. SenSys, Nov. 2004.

[16] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP Topologies with Rocketfuel," In Proceedings of ACM/SIGCOMM '02, August 2002.

[17] R. vanRenesse, K.P. Birman, and W. Vogels, "Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining," ACM Trans. on Computer Systems, Vol. 21, No. 2, May 2003, pp 164-206.

[18] V. Yegneswaran, P. Barford, and S. Jha, "Global Intrusion Detection in the DOMINO Overlay System," In Proceedings of NDSS, San Diego, CA, 2004.

[19] M. Zaharia and S. Keshav, "Adaptive Peer-to-Peer Search," University of Waterloo Technical Report 2004-55, Nov. 2004.