Toposemantic Network Clustering

Leonid B. Poutievski, Kenneth L. Calvert, and James N. Griffioen
{leon, calvert, griff}@netlab.uky.edu
Laboratory for Advanced Networking
University of Kentucky
Lexington, KY 40506-0046

***Abstract*— We study the problem of building an optimal network-layer clustering hierarchy, where the optimality can be defined using three potentially conflicting metrics: state, delay and bandwidth.**

The problem of network clustering where a node's addresses depends on the node's location (e.g. in the hierarchy) is well studied. We study a problem where network nodes are addressed by specifications that might not be tied to locations in the topology.

We propose and compare several distributed clustering algorithms: (i) clustering based solely on topology, (ii) clustering based solely on semantics (node specifications) and (iii) a combination of the above methods (toposemantic network clustering), where we specify a parameter that determines how much the clustering depends on topology and how much on semantics.

We show that the toposemantic method yields the best results when we know the right parameter value for a given topology and assignment of specifications. We propose an algorithm that does not require a parameter, but nevertheless yields better results than the first two methods. [1]

I. Introduction

Traditionally, network clustering is used to achieve scalability. Clustering reduces the amount of routing information that nodes have to store and exchange since the internal nodes and paths inside clusters are hidden from nodes outside of those clusters.

Hierarchical (multi-level) clustering was first introduced by McQuillan [1] and is used to further reduce routing costs by introducing hierarchy[2]. One of the first major analysis of the McQuillan hierarchy was written by Kleinrock and Kamoun [2]. Their paper points out the tradeoffs between network state (sum of sizes of routing tables at all nodes) and delay stretch (relative increase of the average path length) for hierarchical routing and the optimal clustering structures to achieve a minimal state.

Nodes in conventional hierarchical routing systems are identified and addressed by their location in the hierarchy. In this paper we consider a network-layer service where nodes are identified by specifications that might not correlate with locations in a network topology. The network's job is to deliver the packet to all nodes (and only those nodes) whose specifications match the *destination specification* carried in the packet. Unlike overlay networks, our delivery service does *not* rely on the existence of any underlying network service. As such, it can subsume traditional network-layer services (e.g. unicast and multicast) as well as support higher-level services like publish-subscribe systems.

Because specifications do not necessarily reveal anything about location, routing and forwarding algorithms need to maintain topological information along with the specifications. So in our problem the network state consists of two parts: topological and semantic (specification) information.

Besides hiding internal topological information, the benefit of clustering is that nodes with similar specifications can be combined to form a node whose specification may have a more compact representation (size). In addition, cluster specifications can be *abstracted*—replaced with less precise specifications, further reducing size. However, this state reduction comes at the cost of *overdeliveries* (false positives), i.e. bandwidth wasted by forwarding packets along paths that do not lead to valid destinations.

The main challenge of clustering is to aggregate topological and semantic information in a manner that minimizes: (i) the total network topological and semantic *state* (i.e. routing table sizes), (ii) the average delay, or path length (measured by the average path length over all pairs of destinations) and (iii) the network load (measured by the number of links an average packet traverses during forwarding, including *overdeliveries*). These three metrics are related, and we explore the tradeoffs between them.

II. Clustering Approaches

Clustering is difficult because so many possible assignments of nodes to clusters exist. We start by describing two approaches that can be used to cluster nodes: topological and semantic clustering.

A. Topological Clustering

In topological clustering specifications are not considered; clustering is only based on topology. The goal of topological clustering is to minimize the number of routing table entries to clusters. Kleinrock and Kamoun [2] show that the optimal type of hierarchy to achieve this goal is a *balanced* hierarchy, where all nodes have approximately the same depth in the hierarchy and all clusters are composed of the same number of lower level clusters. The advantages of the balanced hierarchy is that we can control the tradeoff between routing table size and delay stretch. This trade-off has also been studied in the area

[1]The support of the National Science Foundation under grant CNS-0435272 is gratefully acknowledged.

[2]Throughout this paper, "clustering" and "hierarchical clustering" are used interchangeably

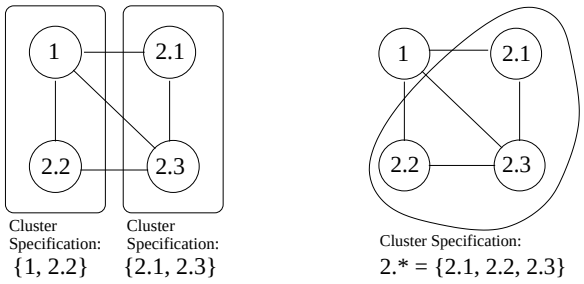


Fig. 1. (a) Topological Clustering vs (b) Semantic Clustering

of compact and interval routing ([3]). Their schemes assume unicast communication and a only single copy of a packet presented in a network.

Even though a balanced hierarchy is characterized by a scalable number of routing table entries, the disadvantage is that the size of the entry may grow linearly with the number of nodes that it represents. For example, if node specifications are both random and flat identifiers (there is no structure), the cluster specification will be constructed as a set of node specifications.

B. Semantic Clustering

Data clustering (sometimes called cluster analysis [4], [5]) attempts to partition a data set into subsets (clusters) so that data in each subset is similar for some definition of "similar". For example, hierarchical bottom-up clustering algorithms begin with each element in a separate cluster and then successively finds pairs of similar clusters and merges them.

It is possible to use data clustering techniques to define network clusters: nodes correspond to data elements, and similarity can be defined as a *semantic similarity (distance)* between node specifications. Unlike standard data clustering, our problem has an additional *constraint* – the network topology. Clusters can only be merged if there is an edge in the graph connecting them.

C. Topological vs Semantic Clustering

The advantage of semantic clustering is that the size of each cluster specification can be small. For example, if two nodes A and B with similar specifications are merged together, then the common part can be factored out of the specification of the newly formed cluster. In this case, node specifications consist only of a difference with the cluster specification. The disadvantage is that the number of routing table entries can be larger than in the balanced case.

The difference between topological and semantic clustering is illustrated in Figure 1 for a network consisting of 4 nodes. Topological clustering in Figure 1(a) groups nodes into equal size clusters (clustering is balanced). This minimizes the number of routing table entries (each node has an entry pointing to the other cluster and an entry to the other node in the same cluster). Semantic clustering (see Figure 1(b)) groups together nodes with similar specifications. This reduces specification sizes. For example, the abstracted specification $2.*$ can be used as a cluster specification instead of enumerating $2.1, 2.2$ and 2.3.

D. Other Clustering

An overview of clustering techniques is given in Perkins' overview [6], which describes many clustering algorithms, each with its own goal: minimizing router state, maximizing the connectivity inside each cluster, localizing high-intensity traffic within a cluster subject to constraints, such as limiting the number of child clusters or the diameter size. Our goal in this paper is to minimize router state, under a constraint that limits the number of child clusters.

Clustering algorithms have also been proposed for ad-hoc and sensor networks ([7], [8], [9]). Comparatively, in our algorithm we are trying to reduce the amount of routing information, consisting of both topological and semantic information.

III. TOPOSEMANTIC CLUSTERING

As the name suggests, we propose the use of an algorithm that combines both topological and semantic criteria in forming clusters, with the goal of minimizing total network state—which consists of both topological and semantic information. To avoid an excessive number of levels in the resulting hierarchy and thus a high delay stretch, we introduce the constraint *brMax*: a branching factor that determines the maximum number of child clusters. The value of *brMax* is assumed to be at least two.

Our algorithm performs bottom-up hierarchical clustering. It starts with each node in a separate cluster. At each step the algorithm selects a pair of neighboring clusters to merge. We distinguish two types of merging operations: *Push* and *Fuse*. *Push*(A, B) forms a new cluster N containing A and B as subclusters. *Fuse*(A, B) forms a new cluster N containing $A_0, A_1, \ldots, A_n$ and $B_0, B_1, \ldots, B_m$ as subclusters where the A_is are subclusters of A and B_is are subclusters of B (clusters A and B cease to exist).

Both *Push*(A, B) and *Fuse*(A, B) operations reduce a state in nodes that keep routing table entries corresponding to clusters A and B (i.e. nodes in clusters that are siblings to A and B).

Merging two clusters produces state reduction. First, a topological state is reduced, since external nodes will keep one routing table entry instead of two. Second, a semantic state can be reduced, since the size of combined cluster specifications might be smaller that the sum of the sizes of the two former specifications. Let $\mathsf{size}(X)$ denote the size of the routing table entry describing X at nodes that are outside of X. The amount of reduction ($\mathsf{sizeRed}(A, B)$) due to replacing entries corresponding to A and B with a single routing table entry corresponding to a new cluster N created by *Push* or *Fuse* can be calculated as

$$\mathsf{sizeRed}(A, B) = \mathsf{size}(A) + \mathsf{size}(B) - \mathsf{size}(N). \quad (0)$$

Since each routing table entry consists of both topological and semantic information, we denote its size as a weighted

sum, where τ is a weight of the topological state and σ is a weight of the semantic state,

$$\mathsf{size}(X) = \mathsf{specSize}(X)\sigma + \mathsf{topoSize}(X)\tau. \quad (1)$$

We assume that the size of the topological information is equal for every cluster, then without loss of generality we assume that $\forall X, \mathsf{topoSize}(X) = 1$. Let $\mathsf{specRed}(A, B)$ denote the reduction of the semantic state (i.e. $\mathsf{specRed}(A, B) = \mathsf{specSize}(A) + \mathsf{specSize}(B) - \mathsf{specSize}(N)$), then from (0) and (1) we get

$$\mathsf{sizeRed}(A, B) = \mathsf{specRed}(A, B)\sigma + \tau. \quad (2)$$

Let $\mathsf{sib}(A, B)$ denote the number of nodes in clusters that are siblings to A and B. The total reduction in state is then

$$H(A, B) = (\mathsf{specRed}(A, B)\sigma + \tau)\mathsf{sib}(A, B).$$

Since H estimates the amount of the reduction of the network state, we propose to use H as the metric for choosing which clusters to merge. That is, if we take $\sigma = 0$, then merging is based solely on topological state reduction; if we take $\tau = 0$ it is based entirely on reduction in semantic state.

We propose the following greedy centralized algorithm. Initially, there is one cluster containing all nodes. The following steps are iterated as long as it is possible to decrease the state:

1. Evaluate the heuristic H on every pair of neighboring clusters A and B such that the number of subclusters in the parent cluster of A and B exceeds *brMax*.
2. Pick a pair of neighboring clusters A and B with a maximum state reduction $H(A, B)$. If either the total number of subclusters of A and B does not exceed *brMax* or A or B contains only a single node, then *Fuse*(A, B); otherwise, *Push*(A, B).

In the distributed version, each cluster (e.g. a single cluster representative node per cluster) independently picks one of its neighbors as a candidate to merge with. (Link or cluster identifiers can be used to break ties when there are several choices with the same maximum heuristic value.) If two neighboring clusters pick each other, they merge. To calculate the sib function, each routing message corresponding to a cluster should carry the number of nodes inside that cluster.

IV. Results

We would like to quantify the tradeoffs among state, delay, and overdelivery (caused by "false positives" due to abstraction) under different clustering approaches. To evaluate these tradeoffs, we simulated topological, semantic and combined approaches. We first evaluated a topological clustering algorithm that ignores specifications and considers only topology in aggregation. We compare properties of resulting hierarchies built by topological and semantic clustering algorithms and show the drawbacks of these algorithms. We then evaluate a combination of the above algorithms (toposemantic clustering) and compare it to the two algorithms above. We show that the toposemantic algorithm performs better than the topological and semantic clustering algorithms.

A. Simulation Setup

All of our tests were run using transit-stub topologies of 600 nodes generated by GT-ITM [10] (3 stubs per transit node, 9 extra transit-stub edges and 24 extra stub-stub edges). All results are calculated as an average over 20 different topologies. For all graphs (except Figure 2) the branching factor *brMax* $= 10$.

To simulate network traffic, we implemented a *unicast* traffic model for which a source and destination node were selected randomly. Each node is assigned a unique specification (described in section IV-D.1). A packet was then sent from the selected source with the destination node specification carried in the packet. This procedure was repeated to create traffic load across the network for the duration of the simulation. Forwarding is performed in several steps. First, a source node sends a packet toward visible clusters those specifications match the destination specification, once packet enters these clusters, the same procedure is repeated for subclusters and so on.

B. Metrics

The state metric consists of two components. First, we measure the *topological state*—the average number of routing table entries (*visible clusters*) over all nodes in the network. Second, we consider the *specification state*, i.e. the total size of cluster specifications stored at all nodes. We also measure a (relative) *specification state ratio*—specification state for a given clustering divided by the specification state with no clustering.

We define the *delay cost* to be the number of edges crossed on the way from the source to a node matching the destination specification. We define *delay stretch* to be the ratio of the algorithm's measured delay cost versus the minimum possible delay cost (over the shortest path).

Network load is measured as the number of links over which a message is forwarded en route to its destination. Note, this includes all links over which the message is forwarded, even if the link does not lead to any node that matches the destination specification. We define the *network load ratio* to be the ratio of the total number of edges crossed by a message versus the number of links in the shortest-path from a source to a destination.

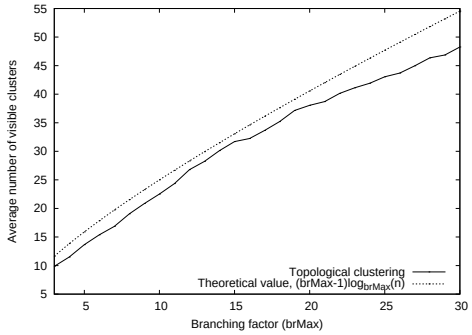
Ratio of overdeliveries is used to measure the cost of abstraction and is calculated as the ratio of network load under abstraction to the network load under no abstraction. For unicast, this metrics is equal to load divided by delay.

C. Topological Clustering

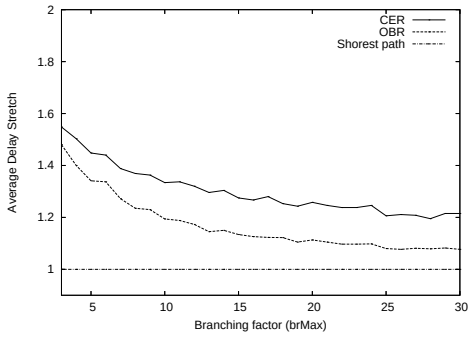
We start by confirming that our clustering algorithm with parameter $\sigma = 0$ ($\tau \neq 0$) in fact builds a balanced hierarchy. Figure 2(a) shows that with increasing branching factor *brMax* the number of routing table entries (visible clusters) increases approximately as the asymptotic growth function $(\textit{brMax} - 1)log_{\textit{brMax}}n$, where n is the number of nodes in the network.

Figure 2(b) confirms the tradeoff (studied by Kleinrock et al.) between the topological state (number of routing table

entries) and delay. Results are shown for two routing schemes: *Closest Entry Routing (CER)* where a cluster is viewed as a single node by the outside nodes and *Overall Best Routing (OBR)* where the exact cost of forwarding through a cluster is known. In the following experiments we use Closest Entry Routing.



(a) Visible clusters. (Maximum 90% confidence interval ± 0.075, coefficient of variation is in the range from 0.16 to 0.38)



(b) Delay. (Maximum 90% confidence interval ± 0.05, coefficient of variation is in the range from 0.03 to 0.24)

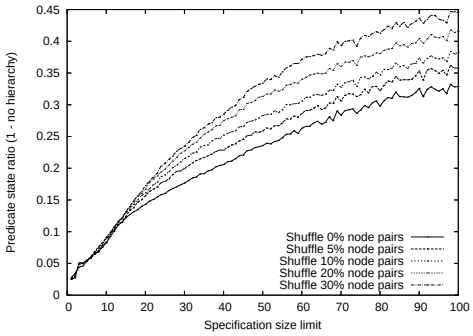
Fig. 2. Topological clustering ($\sigma = 0$)

D. Specification Abstraction

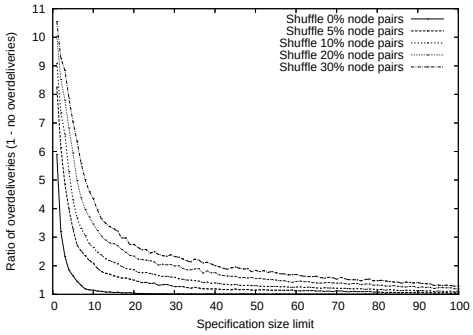
We can control the load-state tradeoff by controlling the maximum size of a cluster specification. By controlling the maximum cluster specification size, we can control the maximum possible network state, since the number of routing entries is known for a given *brMax* and the cluster specification adds a bounded amount of state to each routing table entry. To quantify this tradeoff, we measure the amount of overdeliveries caused by the specification abstraction.

1) Abstractable Specification Language: For our investigation we use a simple abstractable specification language. It reassembles generalized IPv4 addresses, where each specification can correspond to a set of IPv4 network addresses.

Each specification is represented by a tree with labels assigned to nodes. A root node always has a label **true**. Leaf nodes might have a wildcard (*****) label. Two specifications match if in the corresponding trees, every branch of one tree overlaps some branch of the other tree. Two branches overlap if all elements are equal up to the last element or up to the wildcard (*****) label in one of them.



(a) Specification state ratio. (Maximum 90% confidence interval ± 0.03, coefficient of variation is in the range from 0.05 to 0.4)



(b) Amount of overdeliveries. (Maximum 90% confidence interval ± 0.042, coefficient of variation in the range from 0.08 to 0.78)

Fig. 3. Topological clustering ($\sigma = 0, brMax = 10$). Specification abstraction effect

A specification can be abstracted by replacing any branch in the specification tree with a wildcard. A specification tree can be reduced to a given upper bound u on the number of tree nodes (*specification size limit*) by performing a breadth first search, leaving the first u nodes, and replacing all other branches with wildcards.

Each node is assigned a specification based on its hierarchical GT-ITM address. GT-ITM addresses resemble IP addresses and depend on the node's location in the transit-stub topology. We convert GT-ITM addresses to our simple abstractable language, e.g. a GT-ITM address $1.2.3.4$ is represented by a single branch: **true** $\rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$. For specification size limit 2, it can be abstracted to **true** $\rightarrow 1 \rightarrow 2 \rightarrow$ *****.

To measure the load-state tradeoff we vary the maximum cluster specification size. Hierarchy is balanced ($\sigma = 0$), and the branching factor *brMax* $= 10$.

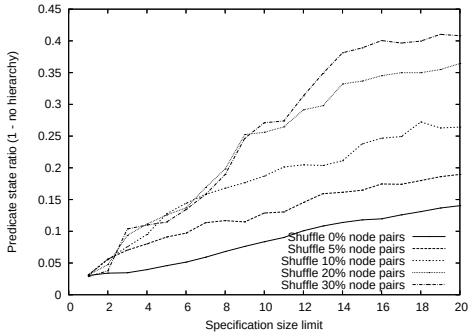
Our simulation results (shown in Figure 3, "shuffle 0% node pairs" curve) confirm that we can effectively control the network state by controlling the maximum size of the cluster specifications, and it shows that the network load ratio (overhead due to abstraction) is inversely proportional to the network state.

E. Semantic Clustering

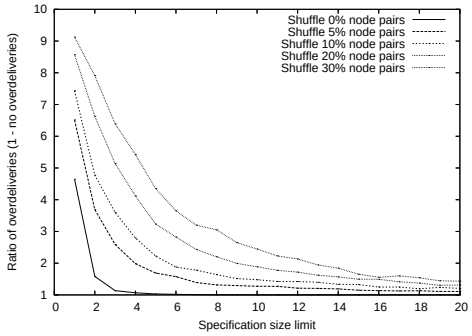
We have quantified the tradeoff between topological state and delay and between semantic (specification) state and

amount of overdeliveries for the topological clustering. Now we can compare these tradeoffs for semantic clustering for the same predicate assignment.

Our expectation is that semantic clustering will perform well when nodes with similar specifications are "close together" in the topology. We expect that semantic clustering is not going to perform as well for randomized specification assignment. We want to quantify the tradeoffs for different levels of *locality*, i.e. correlation between node specifications and node' locations in a topology. We start with our initial specification assignment based on GT-ITM addresses (high locality), then we pick random pairs of nodes and "swap" their specifications. The number of swapped pairs is written as the percentage of all nodes in the network. More "swaps" lead to lower locality, and 50% of swapped specifications gives mostly random assignment of specifications. Instead of pure semantic clustering ($\tau = 0, \sigma \neq 0$), we use mostly semantic clustering ($\tau = 1, \sigma = 10^6$), where sizes of clusters are considered only as a tiebreaker between pairs with equal semantic reductions.



(a) Specification state ratio. (Maximum 90% confidence interval ± 0.03, coefficient of variation is in the range from 0.05 to 0.4)



(b) Amount of overdeliveries. (Maximum 90% confidence interval ± 0.03, coefficient of variation in the range from 0.02 to 0.73)

Fig. 4. Semantic clustering ($\tau = 1, \sigma = 10^6, brMax = 10$): small cluster predicates, but large number of routing table entries

First, as we expected, under high locality assignment (0% shuffle), semantic clustering (Figure 4) is better than topological clustering (Figure 3). For the same specification size limit, semantic clustering gives lower state and ratio of overdeliveries.

Then by comparing Figures 3 and 4 we confirm our guess that in semantic clustering the size of each cluster specification can be much smaller than in topological clustering, but the number of such clusters can be significantly larger than in a balanced hierarchy.

In Figure 4(b) we can see the tradeoff between the amount of overdeliveries and the maximum specification size for different amounts of locality. Compared to the topological clustering in Figure 3(b), the amount of overdeliveries for the same maximum specification size and locality is much lower for semantic clustering. (Note the difference in scales of the two graphs.)

The drawback of semantic clustering, as shown in Figure 4(a), is that even small randomness in the specification assignment sharply increases the number of visible clusters which in turn increases the total network state, even though the size of each cluster specification is small. Figure 5 shows that even small decreases in locality lead to large increases in number of visible clusters.

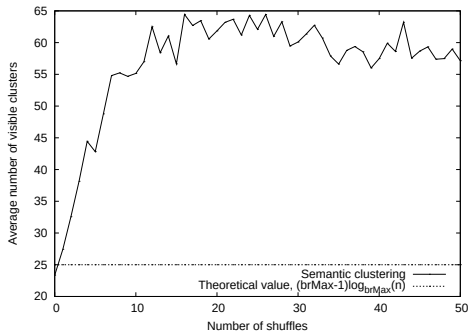


Fig. 5. Semantic clustering ($\tau = 0$, specification limit size = 4): even small amount of randomness leads to unbalanced hierarchy

F. Comparison

In the toposemantic clustering algorithm, we try to combine the best properties of previously studied topological and semantic clustering: a small number of routing table entries as in the topological clustering and small cluster specifications as in the semantic clustering.

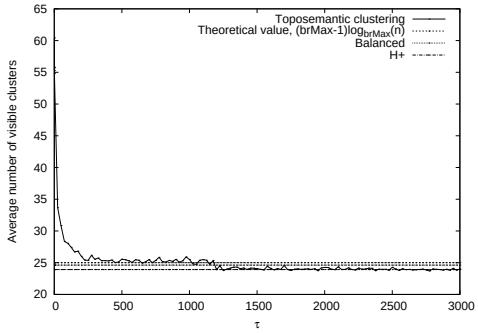
In Figure 6 we compare topological, semantic and toposemantic clustering (for toposemantic clustering we fix $\sigma = 1$ and increase τ from 1 to 3000). We also consider a special case of toposemantic clustering (called H^+) with $\sigma = 1$ and τ is very large ($\forall A, \tau > \mathsf{specSize}(A)$). The expectation is that with such parameters, the hierarchy will be built balanced and specifications will be considered as a secondary criteria when topological conditions are equal.

In Figure 6, toposemantic clustering changes from mostly semantic on the left side to more balanced (topological) on the right. We notice that the number of routing table entries (visible clusters) sharply decreases for values from 0 to 250. The same is true for the specification state ratio.

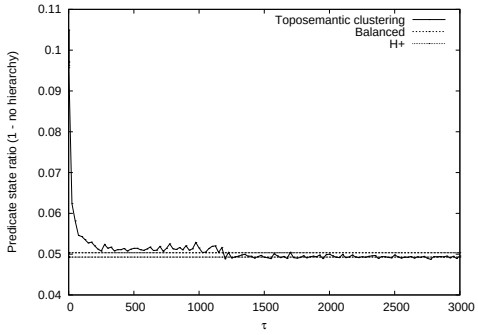
Since the load decreases with the increasing parameter τ, the best value of τ is the smallest value such that number of visible clusters is close to the number corresponding to a balanced

clustering. In Figure 6 the best results for the toposemantic clustering are achieved for approximately $\tau = 250$.

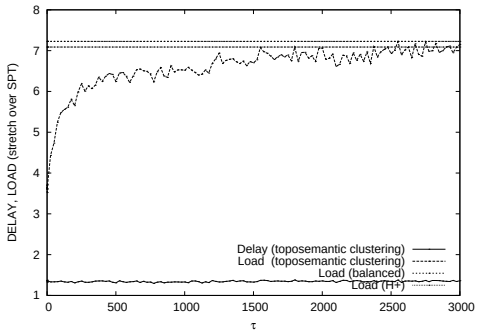
In Figure 6(c) we notice that once the hierarchy built by the toposemantic algorithm becomes mostly balanced ($\tau > 250$), it is better than the hierarchy build by the topological clustering. For approximately the same state and delay, the load is lower. Similarly, in Figure 6(c) we can see that H^+ performs better than the topological clustering.



(a) Number of visible clusters



(b) Specification state ratio



(c) Delay, load

Fig. 6. Toposemantic routing ($\sigma = 1$, specification limit size = 4, shuffle 10% of node pairs)

Finally, in Figure 7 we compare 3 types of clustering: balanced, H^+ and toposemantic with the best parameter values found in the previous experiment ($\sigma = 1, \tau = 250$). We measure the tradeoff between the specification state ratio and the ratio of overdeliveries by changing the maximum specification size. The best clustering corresponds to points closest to the origin of coordinates (0,0) in Figure 7.

As we can see, the toposemantic clustering builds the best hierarchy, but requires a parameter that depends on topology and the specification assignment. H^+ clustering is the second best, but does not require any parameters. It is best suited for a distributed algorithm where it is impossible to calculate the best parameter value off-line.

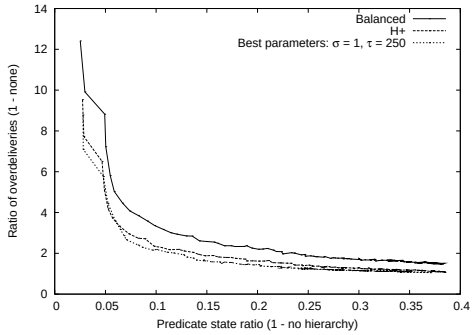


Fig. 7. Tradeoff between network load and network state (specification limit size = 4, shuffle 10% of node pairs)

V. Conclusions

We have presented a network clustering problem for a general network-layer service where nodes are addressed by specifications that might not correlate with locations in a network topology. We believe that a study of solutions for this problem can lead to insights into techniques for scalable routing in the Internet. Our results can also be applied for self-organizing networks.

We have studied the properties of topological and semantic clustering and presented a novel toposemantic algorithm that builds a clustering hierarchy based on topology and specifications assigned to nodes. We have analyzed our algorithm and found that it produces better results than semantic and topological algorithms.

References

[1] J. McQuillan, "Adaptive routing algorithms for distributed computer networks," Tech. Rep. 2831, BBN, May 1974.

[2] Leonard Kleinrock and Farouk Kamoun, "Hierarchical routing for large networks. performance evaluation and optimization," *Computer Networks: The International Journal of Distributed Informatique*, vol. 1, no. 3, pp. 155–174, Jan. 1977.

[3] Mikkel Thorup and Uri Zwick, "Compact routing schemes," in *SPAA '01: Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, New York, NY, USA, 2001, pp. 1–10, ACM Press.

[4] Charles H. Romesburg, *Cluster analysis for researchers*, Lifetime Learning Publications, 1984.

[5] B. S. Everitt, S. Landau, and M. Leese, *Cluster analysis*, A Hodder Arnold Publication, 2001.

[6] Charles E. Perkings, *Ad hoc networking*, chapter "Cluster-based networks" by Martha Steenstrup, Addison Wesley, 2001.

[7] Elizabeth M. Belding-Royer, "Multi-level hierarchies for scalable ad hoc routing," *Wirel. Netw.*, vol. 9, no. 5, pp. 461–478, 2003.

[8] Rajesh Krishnan, Ram Ramanathan, and Martha Steenstrup, "Optimization algorithms for large self-structuring networks.," in *INFOCOM*, 1999, pp. 71–78.

[9] Jie Wu, "Extended dominating-set-based routing in ad hoc wireless networks with unidirectional links," *IEEE Transactions on Parallel and Distributed Computing*, vol. 22, pp. 327–340, 2002.

[10] E. Zegura and K. Calvert, "Georgia Tech Internet Topology Models," http://www.cc.gatech.edu/projects/gtitm.