Towards Planning the Transformation of Overlays

Young Yoon[1], Nathan Robinson[2], Vinod Muthusamy[3], Sheila McIlraith[2], Hans-Arno Jacobsen[2]
[1]Samsung Electronics, [2]University of Toronto, [3]IBM T.J. Watson Research Center

I. Introduction

Overlay networks are prevalent in distributed systems, and are used by a wide variety of applications [1]–[4]. The topology of an overlay network is critical to the performance of the overlay, and there is extensive work on designing overlay topologies to achieve a variety of performance metrics such as minimizing path lengths, controlling node degrees, or maintaining redundant paths [5,6]. In a dynamic system, an overlay may become obsolete and require reconfiguration. For example, changes in traffic patterns, physical network characteristics, and application requirements may render an existing topology sub-optimal, and necessitate reconfiguration [6].

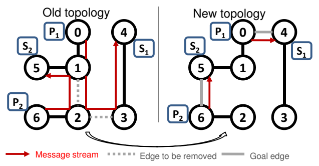

Fig. 1. Reconfiguring an overlay

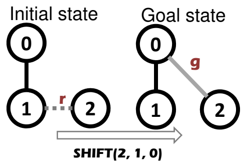

Fig. 2. Example of SHIFT operation

Suppose we have an overlay, as shown in Figure 1. Brokers (B) are interconnected to route messages from P_1 and P_2 to S_1 and S_2, who are interested in the messages. Assume that S_1 and S_2 demand more timely delivery of messages. This demand may be met by reducing the average length of the paths over which the messages produced by P_1 and P_2 must travel. This reduction can be achieved by reconfiguring the topology, That is, new routing paths B_0-B_4 and B_5-B_6 are established, while B_1-B_2 and B_2-B_3 are disconnected.

In this work, we define the incremental topology transformation (ITT) problem, which seeks to find a plan, or sequence of incremental overlay reconfiguration steps, that yields the least disruption [7]. Each incremental step is chosen from a set of reliable primitive transformation operations that ensure reliable message delivery. Any such plan must consider the disruption to message delivery that arises from any routing state updates applied during the overlay reconfiguration. As well, the plan must carefully coordinate the reconfiguration steps to avoid transient loops, and message loss or reordering [8]. Moreover, the solution space of possible plans grows extremely quickly with the size of the overlay, and an exhaustive search is infeasible for typical enterprise overlays with hundreds of nodes [9,10].

Note that this paper differs from existing work on the mechanics of reconfiguring an overlay at runtime [11,12]. Here we are concerned with devising a transformation *plan* that minimizes disruption *during* reconfiguration.

II. Problem and Solution

An operation that is suitable for an incremental transformation of a running topology was first introduced in

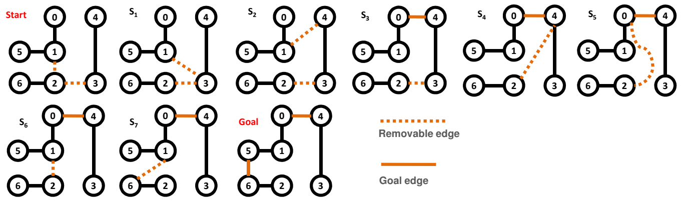

Fig. 3. A plan with 7 SHIFT steps

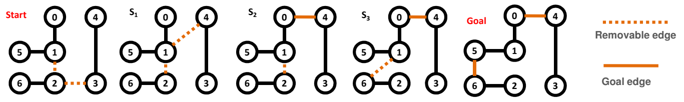

Fig. 4. A shorter plan of the problem in Figure 3

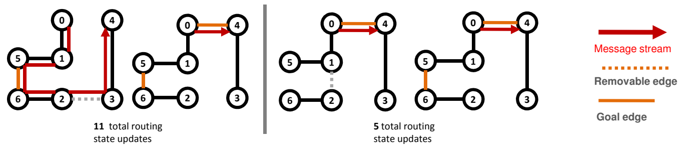

Fig. 5. A plan with MOVE operations. The order of executing the two MOVE operations can yield different number of routing state updates.

[8], *i.e.*, SHIFT(v_i,v_j,v_k). It is already proven that there exists a sequence of these operations to transform any topology of the type we consider into any other with the same type. As shown in Figure 2, SHIFT(v_i,v_j,v_k) replaces the edge (v_i,v_j) with edge (v_i,v_k), where $\{v_i,v_k\} \in \mathbb{N}(v_j)$ and $v_i \neq v_k$. $\mathbb{N}(v_x)$ is the set of neighbors of v_x. We introduce another atomic operation: MOVE(v_i, v_j, v_k, v_l) which directly replaces the removable edge (v_i,v_j) with the goal edge (v_k,v_l).

A plan may seek to minimize the number of steps in the plan or the number of message streams affected by the transformation. Consider the two plans in Figures 3 and 4 that achieve the same transformation but with the latter requiring fewer steps. Alternatively, consider the two plans in Figure 5 that use the same number of MOVE operations. With the first plan, the message stream between v_0 and v_4 is disrupted by a MOVE operation that requires all vertices to update their routing states. For the second plan, the goal edge (v_0,v_4) is established first, and requires fewer total routing state updates.

We have encoded the ITT problem into PDDL [13] but found that the state-of-the-art domain-independent planning systems are unable to solve PDDL encodings of our problems effectively. For example, neither LAMA [14] nor PROBE [15] were able to solve problems with more than 50 nodes, where 50% of edges were changed between the start and goal graphs, within 10 minutes on a machine with 16 GB of memory and an Intel Xeon 3.00 GHz processor.

From an in-depth study of the problem, we have identified the following properties, from which we derive a number of heuristics that we use in our custom algorithms. We denote the path between the two nodes (v_x and v_y) that consist of a goal edge g as $\overrightarrow{p}^{\diamond}(g)$ ($\equiv\overrightarrow{p}(v_x,v_y)$). Given this notation and the observation we made above, we set a rule for selecting a removable edge for the transformation operations as specified in Theorem 1 that is based on Corollary 2.[1]

Theorem 1. *A* MOVE *or a* SHIFT *operation α to achieve a goal edge, $g(v_x,v_y)$, is valid,* iff *a removable edge that is on $\overrightarrow{p}^{\diamond}(g)$ is involved in α.*

Corollary 2. *There must be at least one removable edge on the path between v_x and v_y of a goal edge $g(v_x,v_y)$ in a problem $\mathcal{T}$ of transforming a connected acyclic topology T_S to another connected acyclic topology T_G.*

Property 3. *A stationary edge should not be involved in any transformation operation.*

We develop two algorithms that capture the domain specific knowledge. The GREEDY algorithm quickly finds a plan consisting of SHIFT operations. For each goal edge $g(v_x,v_y)$ in the set of goal edges G, the algorithm first searches for the path $\overrightarrow{p}(v_x,v_y)$ (denoted also as $\overrightarrow{p}^{\diamond}(g)$). It selects a removable edge in $\overrightarrow{p}^{\diamond}(g)$ and performs a sequence of SHIFT operations to replace it with the goal edge. The process is repeated until the goal topology is achieved. The BFS algorithm does a best-first-search and is similar to the algorithm underlying the state-of-the-art planners LAMA [14] and PROBE [15]. Unlike these existing algorithms, it does not ground all actions upfront and uses domain-specific heuristics. A heuristic function measures a goal distance for a state s by assigning removable edges in s to unsatisfied goal edges such that goal edges are assigned removable edges greedily, according to the distance. The distance from a removable edge r to a goal edge g is computed as the number of SHIFTs it would take to move r to g. Note that, as discussed above, a removable edge can only be moved to a goal edge if it is currently on $\overrightarrow{p}^{\diamond}$ of a goal edge.

III. EVALUATION

We evaluated our planning algorithms with simulated ITT problems, with parameter values that reflect the typical scale we have observed in enterprise distributed publish/subscribe systems such as GooPS [9]. We measure the overhead of the algorithms and the quality of the plan our algorithms find. We vary the number of nodes in the start and goal graphs and the degree of change in terms of the percentage of links to be changed from an initial topology.

a) Overhead: The overhead of a planning algorithm is measured as the elapsed time to find a plan. The GREEDY algorithm found a plan in 1 ms for transforming a 20-node network with 10% change, and about 8 seconds for a 400-node topology with 60% change. The BFS algorithm, which is tuned to find a higher-quality plan, performs considerably worse: it took about 2 ms for a 20-node topology with 10% change. For a 100-node graph with 60% change, BFS took about a minute to find a plan. This difference is a result of exploring a larger search space to find a better plan.

The performance of our planners are considerably better than the generic state-of-the-art planners. Those planners could not even find a plan for transforming a graph with more than 50 nodes with over 50% change, whereas GREEDY can find a plan well under a second. This is clear empirical evidence that the domain-specific heuristics we derived are effective.

b) Plan Quality: The quality of a plan is measured by the number of actions in it and the number of routing state updates it incurs. The settings of the problem are the same as the experiments in the previous section that assessed overhead. We observe that GREEDY generated about six times more actions than BFS. For a 20-node topology with 10% change, BFS yielded an average of four actions in the plan. For a 100-node topology with 30% change and a 400-node topology with 10% change, BFS required about 50 and 200 actions, respectively. In the latter case, the plans found by GREEDY required over 1500 actions.

IV. CONCLUSIONS

The planning of overlay transformations that seek to minimize the disruption to a live overlay network has received little attention. In this paper, we showed that applying AI techniques to this problem is clearly promising but requires further study. The interested reader may refer to our extended paper for more details [7].

REFERENCES

[1] I. Koenig, "Event processing as a core capability of your content distribution fabric," in *Gartner Event Processing Summit*, 2007.

[2] G. Li, V. Muthusamy, and H.-A. Jacobsen, "A distributed service-oriented architecture for business process execution," *ACM Trans. Web*, 2010.

[3] Y. Yoon, C. Ye, and H.-A. Jacobsen, "A distributed framework for reliable and efficient service choreographies," in *WWW*, 2011.

[4] M. Sadoghi, M. Jergler, H.-A. Jacobsen, R. Hull, and R. Vaculín, "Safe distribution and parallel execution of data-centric workflows over the publish/subscribe abstraction," *IEEE TKDE*, 2015.

[5] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz, "OverQos: an overlay based architecture for enhancing internet QoS," in *NSDI*, 2004.

[6] Y. Zhu, B. Li, and J. Guo, "Multicast with network coding in application-layer overlay networks," *Selected Areas in Communications, IEEE Journal on*, 2004.

[7] Y. Yoon, N. Robinson, V. Muthusamy, S. McIlraith, and H.-A. Jacobsen, "Planning the transformation of distributed messaging middleware," Tech. Rep., 2014. [Online]. Available: http://msrg.org/papers/planning2014

[8] Y. Yoon, V. Muthusamy, and H.-A. Jacobsen, "Foundations for highly available content-based publish/subscribe overlays," in *ICDCS*, 2011.

[9] J. Reumann, "Pub/Sub at Google," lecture & Personnal Communications at Middleware 2009 and CANOE Summer School 2009.

[10] J. Fan and M. H. Ammar, "Dynamic topology configuration in service overlay networks: A study of reconfiguration policies," in *INFOCOM*, 2006.

[11] R. Baldoni, R. Beraldi, L. Querzoni, and A. Virgillito, "Subscription-driven self-organization in content-based publish/subscribe," in *ICAC*, 2004.

[12] G. Chockler, R. Melamed, Y. Tock, and R. Vitenberg, "Constructing scalable overlays for pub-sub with many topics," in *ACM PODC*, 2007.

[13] S. Edelkamp and P. Kissmann, "PDDL 2.1: The language for the classical part of IPC-4," in *IPC-4*, 2004.

[14] S. Richter and M. Westphal, "The LAMA planner: Guiding cost-based anytime planning with landmarks," *Journal of Artificial Intelligence Research (JAIR)*, 2010.

[15] N. Lipovetzky and H. Geffner, "Searching for plans with carefully designed probes," in *ICAPS*. AAAI, 2011.

[1]The proofs and algorithms are in an extended paper [7].