Towards Planning the Transformation of Overlays

Young Yoon\textsuperscript{1}, Nathan Robinson\textsuperscript{2}, Vinod Muthusamy\textsuperscript{3}, Sheila McIlraith\textsuperscript{2}, Hans-Arno Jacobsen\textsuperscript{2}
\textsuperscript{1}Samsung Electronics, \textsuperscript{2}University of Toronto, \textsuperscript{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 \cite{1}–\cite{3}. 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 \cite{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 \cite{6}.

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 \cite{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 \cite{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 \cite{9,10}.

Note that this paper differs from existing work on the mechanics of reconfiguring an overlay at runtime \cite{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 \cite{8}, i.e., $\text{SHIFT}(v_i, v_j, v_k)$. 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, $\text{SHIFT}(v_i, v_j, v_k)$ replaces the edge $(v_i, v_j)$ with edge $(v_i, v_k)$, where $\{v_i, v_j\} \in N(v_k)$ and $v_i \neq v_k$. $N(v_k)$ is the set of neighbors of $v_k$. We introduce another atomic operation: $\text{MOVE}(v_i, v_j, v_k)$, 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 $\text{MOVE}$ operations. With the first plan, the message stream between $v_0$ and $v_4$ is disrupted by a $\text{MOVE}$ 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 \cite{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 \cite{14} nor PROBE \cite{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 \((v_x, v_y)\) that consist of a goal edge \(g\) as \(P^g\) \((\equiv P(v_x, v_y))\). 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.

**Theorem 1.** A **move or a shift** operation \(\alpha\) to achieve a goal edge, \(g(v_x, v_y)\), is valid, if a removable edge that is on \(P^g\) is involved in \(\alpha\).

**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 \(T\) 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 \(P^g(v_x, v_y)\) (denoted also as \(P^g\)). It selects a removable edge in \(P^g\) 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 **shift**s 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 \(P^g\) 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


