Dynamic Topology Configuration in Service Overlay Networks: A Study of Reconfiguration Policies

Jinliang Fan Mostafa H. Ammar
College of Computing, Georgia Institute of Technology Atlanta, GA 30332-0280
{jlfan,ammar}@cc.gatech.edu

***Abstract*— The routing infrastructure of the Internet has become resistant to fundamental changes and the use of overlay networks has been proposed to provide additional flexibility and control. One of the most prominent configurable components of an overlay network is its topology, which can be dynamically reconfigured to accommodate communication requirements that vary over time. In this paper, we study the problem of determining dynamic topology reconfiguration for service overlay networks with dynamic communication requirement, and the ideal goal is to find the optimal reconfiguration policies that can minimize the potential overall cost of using an overlay. We start by observing the properties of the optimal reconfiguration policies through studies on small systems and find structures in the optimal reconfiguration policies. Based on these observations, we propose heuristic methods for constructing different flavors of reconfiguration policies, i.e., never-change policy, always-change policy and cluster-based policies, to mimic and approximate the optimal ones. Our experiments show that our policy construction methods are applicable to large systems and generate policies with good performance. Our work does not only provide solutions to practical overlay topology design problems, but also provides theoretical evidence for the advantage of overlay network due to its configurability.**

I. Introduction

While the Internet has been fully commercialized and evolved into a ubiquitous medium of communication, its native routing infrastructure has become resistant to fundamental changes. This hinders the development of new network functionality (e.g., multicast, QoS) that heavily rely on such fundamental changes. The use of *overlay networks* has been proposed as an alternative solution that can potentially provide the desirable flexibility and control of the routing infrastructure [1], [2], [3], [4]. Researchers have successfully used overlay networks to solve problems in various areas. For example, overlay networks have been employed to implement application layer multicast [5], [6], [7], [8], provide testbeds for new technologies [9], [10], [11], circumvent BGP faults and constraints [12], and provide countermeasures to DoS attacks [13]. In this paper, we focus our discussion on *service overlay networks* [14], [12], [15] that are deployed and maintained by *overlay network providers*. Overlay network providers deploy a number of specially designed *overlay nodes* across the Internet. As third-party service providers these overlay network providers contract with underlying ISPs and buy network bandwidth between these overlay nodes. They, in turn, provide value-added network services to end-systems, which access the overlay networks through one of the overlay nodes. Traffic between end-systems is carried by and routed through the overlay networks instead of the native networks.

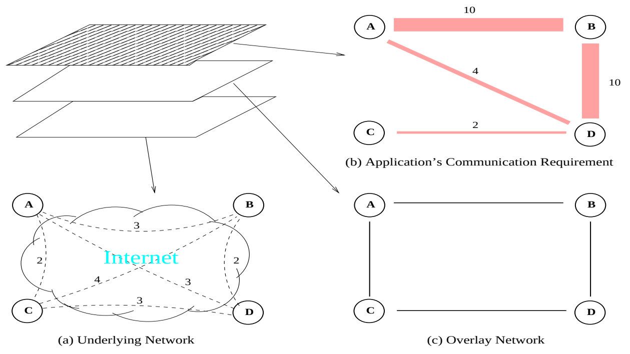


Fig. 1. Factors in Overlay Topology Design

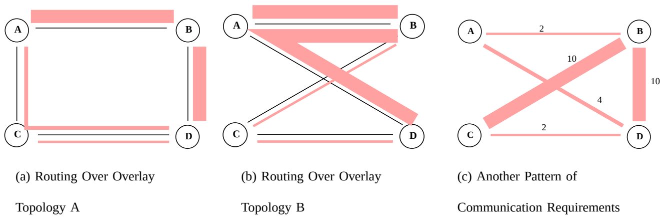

A
B
C
D
(a) Routing Over Overlay Topology A
A
B
C
D
(b) Routing Over Overlay Topology B
A
2
B
10
10
4
C
2
D
(c) Another Pattern of Communication Requirements

Fig. 2. Routing Data Through Overlay Networks

One of the most important issues in the design of a service overlay network is the configuration of its overlay topology. Positioned between the native networks and the ultimate customers, an overlay topology constructed in favor of both could significantly improve the performance or reduce the operation cost of the whole system. Consider the four-node example overlay network shown in Fig. 1. Fig. 1.a characterizes the service agreement between the overlay network provider and the underlying ISPs; the labels on the dashed lines show the operation costs of the potential overlay links between every pair of nodes — the price ISPs charge the overlay network provider for shipping one unit of data over these overlay links. Fig. 1.b shows a snapshot of the communication requirements aggregated over all customers; the labels on the lines (or the thickness of the lines) denote the data rates between every pair of overlay nodes, on behalf of the customers. Fig. 1.c shows a candidate overlay topology (Topology A); each edge denotes an overlay link. Provided that the overlay network ships data over the overlay paths that incur the minimum operation cost, the flow of data is shown in Fig. 2.a. Fig. 2.b presents the flow of data if another topology (Topology B) is adopted instead.

In terms of operation cost, Topology B is not as good as A — although some data can now take a lower-cost path, a higher volume of data have to take a higher-cost path. Generally, the operation cost of an overlay varies when it runs on different overlay topologies and this raises the problem of finding the overlay topology that minimize the operation cost for given communication requirements.

If the communication requirement is constant over time, the optimal choice of overlay topology is static. If communication requirements change over time, for example, from the ones shown in Fig. 1.b to the ones shown in Fig. 2.c, the overlay topology may need to be reconfigured. While the static topological design problem has been extensively studied for native networks, the design of dynamic topology did not receive the same attention because hard-wired native networks are usually not reconfigurable over small time scales. For overlay networks, however, topology reconfigurability is one of their most appealing features and the dynamic topology design problem becomes interesting. For overlay networks a small or moderate number of overlay nodes, it is feasible for a provider to monitor and statistically model the communication requirements in the system. The topology reconfigurability allows an overlay network to be tuned dynamically when the communication requirements change [16], [17].

A question needs be answered, however, namely, *when and how the overlay topology should be reconfigured under dynamic communication requirements*. One may suggest that whenever the communication requirements change, the topology of the overlay network should be immediately reconfigured to the topology that minimizes the operation cost for the new communication requirements (and the dynamic overlay design problem is reduced to the static one). However, while overlay topology is reconfigurable in small time scale, changing overlay topology is not cost-free; it may incur both management overhead as well as potential disruption of end-to-end flows: overlay links need to be established or torn down; routing tables need to be updated; data in transit may get lost, delayed, or erroneously routed. In the presence of these costs, changing the overlay topology for every change in communication requirements may not be the best policy. Intuitively, if in the long run the benefits of making a change in the overlay topology cannot justify the costs of the change, the overlay topology should not be changed. Even if a change is favorable, the next overlay topology should take into account the long-term changes in communication requirements.

In this paper, we are most interested in the dynamic overlay topology design problem, i.e., the problem of determining dynamic topology reconfiguration in the presence of dynamic communication requirements. In the process of solving this problem, we also study the static overlay topology design problem, the solution to which is used as a subroutine in the solution to the dynamic problem. Particularly, we concentrate on reconfiguration policies that guide the topology selection any time the communication requirements change. Similar problems have been previously studied for the design of optical networks [18], [19] but have not been systematically studied in the context of service overlay networks. In this paper, we formulate both the static and dynamic overlay topology problem. To solve the dynamic topology design problem, we systematically observe how optimal reconfiguration policies are affected by the reconfiguration overhead, through case studies on small systems. Based on the observation, we propose heuristic methods for constructing different flavors of reconfiguration policies for large systems. In the process, we also describe a simulated-annealing based solution for the static overlay topology design problem, which is also used as a subroutine in solving the dynamic problem. Our work does not only provide solutions to practical overlay topology design problems, but also *provides theoretical evidence for the advantage of overlay network due to its configurability*.

The remainder of this paper proceeds as follows. In Section II, we identify the costs that could be incurred in a service overlay network, and formally define the static and dynamic overlay topology design problems. In Section III, we observe the structure of the problem, as well as the properties and structures of the optimal reconfiguration policies, through experiments on small systems. In Section IV, we use these observations as heuristics and propose methods for constructing good reconfiguration policies that approximate the optimal ones for large systems. We evaluate our methods in Section V and conclude in Section VI.

II. Problem Formulation

As shown in Fig. 1, once an overlay topology is established the operation cost for carrying traffic over a service overlay network depends on three factors: the operation cost for shipping data over overlay links, the aggregated communication requirements over all customers and the overlay topology. In this paper, we assume the overlay network provider is charged by the underlying ISPs an operation cost proportional to the amount of traffic carried on an overlay link once the overlay link is established. We use a *link cost matrix*, d, to denote the operation cost per unit of data rate(e.g., bits per second or bytes per second) for the overlay link between every pair of nodes. The aggregated communication requirements over all customers are characterized with a matrix of end-to-end traffic rates between every pair of overlay nodes, named *communication patterns*. We denote the communication requirements at time t by $X(t)$ and assume $X(t)$ takes on values from a set of distinct communication patterns, $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_s\}$.

The *overlay topology*, is a graph $G = <V, E>$, where V includes all the overlay nodes and E includes all the established overlay links between the nodes. Theoretically, there are a total of $2^{n(n-1)/2}$ possible overlay topologies over n nodes. However, not all of these topologies are desirable in practice. An overlay topology is usually required to be *connected* so that every node remains contact with the rest of the overlay network. Furthermore, while some research work in overlay networks assumes fully-meshed overlay topology, in this paper we consider more general cases where the overlay topology is *degree-bounded*, i.e., the number of direct neighbors of a node is limited by an upper bound. Service overlay network providers may prefer a degree-bounded topology over a fully-meshed one for several reasons. First, underlying ISPs may charge the overlay network providers for a certain amount of fixed maintenance cost for an overlay link in addition to operation cost proportional to the actual bandwidth usage. Second, maintaining an overlay link between two overlay nodes incurs control overhead to the overlay network provider themselves(e.g., link condition probing overhead [12], [20]). The above types of overlay link maintenance costs are not directly related to the consumption of link bandwidth for delivering user data and are not formulated in the link cost matrix. Instead, we characterize these types of link maintenance costs using the degree bounds on the overlay topologies — a higher degree bound implies more overlay links and potentially more maintenance cost[1], and a degree bound serves to reflect an overlay network provider's aversion to these types of cost. In this paper, we consider only overlay topologies that are connected and degree-bounded. We assume for simplicity that all nodes are subject to the same degree bound K, $2 \leq K \leq n-1$. (Note that the case $K = n-1$ corresponds to the scenario where node degree is not constrained.) We denote the set of feasible overlay topologies by 0-1 adjacency matrices $\{\mathcal{T}_1, \mathcal{T}_1, \ldots, \mathcal{T}_r\}$.

Finally, an *overlay topology reconfiguration policy* is the sequence of overlay topologies used by an overlay network over time, $Y(t)$, in response to the communication requirement, $X(t)$, changing over time. The problem we are addressing in this paper is finding the optimal reconfiguration policy $Y(t)$ for a given communication requirement $X(t)$. This determination is guided by the cost functions which we discuss later in the section.

A. Static Overlay Topology Design Problem

In an overlay network with topology $\mathcal{T}$, the cost of delivering a unit of data from one node u to another node v is the sum of the operation costs of all overlay links on their overlay routing path. Formally, assume the overlay routing path between u and v is $\mathcal{P}^{u,v}$, then the cost of delivering one unit of data from u to v through the overlay network is:

$$\mathcal{L}^{u,v}(\mathcal{T}) = \sum_{(u,v)\in\mathcal{P}^{u,v}} d(u,v) \tag{1}$$

Then an overall operation cost to support a communication pattern $\mathcal{C}$ on this topology is

$$f(\mathcal{C},\mathcal{T}) = \sum_{u,v\in V} \mathcal{L}^{u,v}(\mathcal{T}) \cdot \mathcal{C}(u,v) \tag{2}$$

in every unit of time. Obviously, function $f(\mathcal{C},\mathcal{T})$ is minimized for a given $\mathcal{C}$ and a given $\mathcal{T}$ if the data are routed through the overlay using minimum-operation-cost paths, which are equivalent to shortest paths if we consider $d(u,v)$ as the distance between u and v. In this paper, we assume the data are always routed through the overlay network following the minimum-operation-cost paths.

[1]According to the handshaking lemma, if the degree bound is K, then the maximum number of overlay links is $\frac{nK}{2}$.

The static overlay topology design problem is the problem of finding an overlay topology $\mathcal{T}$, under the constraints of connectivity and degree-bound, that can minimize the cost function $f(\mathcal{C},\mathcal{T})$ for a communication pattern $\mathcal{C}$. We term such a topology *optimal-static topology* for $\mathcal{C}$ and denote it by $\mathcal{T}^*(\mathcal{C})$. Similar to most other topological design problems, the static overlay topology design problem can be modeled as an integer linear programming problem and is an NP-hard problem. For an ILP formulation of the problem and a proof of its NP-Hardness, please refer to [21].

If the communication requirement $X(t)$ is constant over time, i.e., $X(t) = \mathcal{C}$ for any t, then the optimal overlay topology reconfiguration policy is one that always uses the optimal-static topology for $\mathcal{C}$, i.e., $Y(t) = \mathcal{T}^*(\mathcal{C})$ for all t.

B. Dynamic Overlay Topology Design Problem

The problem formulation in Section II-A applies when the communication requirements do not change over time. In this paper, we are most interested in the more general cases where the communication requirements change over time.

We consider two categories of costs in an overlay network: *Occupancy Cost* and *Reconfiguration Cost*. The occupancy cost is incurred while the overlay network is configured in a particular topology whereas the reconfiguration cost is incurred whenever the overlay topology is reconfigured.

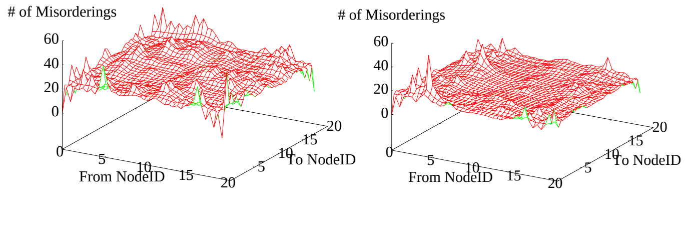
Occupancy Cost: The occupancy cost is the total operation cost for the overlay network to deliver the traffic specified by the dynamic communication requirements $X(t)$ over the dynamic overlay topology $Y(t)$ specified by the overlay topology reconfiguration policy. That is

$$COST_o(\Delta t) = \int_0^{\Delta t} f(X(t), Y(t))dt \tag{3}$$

where Δt is the time horizon of interest and the function $f(\mathcal{C},\mathcal{T})$ is defined in Eq.2.

Reconfiguration Cost: Every time the system reconfigures its overlay topology to adapt to changes in communication requirements, a reconfiguration cost is incurred. This cost is the overhead or the impairment to performance incurred by the transition from one overlay topology to another.

Various costs could be incurred during a topology reconfiguration, depending on the implementation details of the overlay. For example, establishing and tearing down overlay links incur *control and management overhead*, especially when underlying ISPs are involved in the processes. Furthermore, data in transit during topology reconfiguration is subject to routing disturbance that incurs *rerouting overhead*. Depending on the overlay implementation, when overlay topologies and routing tables change, data in transit could be simply dropped by intermediate nodes and requires end-to-end retransmission, or be rerouted, maybe several times, wandering through a path with a high operation cost. Finally, rerouting overhead can be magnified at the end-systems. Data loss and misordering caused by overlay topology reconfiguration are much more significant than those caused by factors in underlying networks. They last longer and their impact is system wide. End-systems' flow control mechanisms that assume low loss rate



(a) Protocol A (b) Protocol B

Fig. 3. Flow Disturbance Caused by Topology Reconfiguration

and short misordering sequence, e.g., TCP's window control system, the impact of overlay topology reconfiguration can be very disruptive.

Fig.3 gives evidence to the existence of reconfiguration cost, e.g., data misordering. The experiments are conducted over the PlanetLab [11] using 20 *.edu* nodes in North America, which form a simple overlay. Each node runs a UDP-based routing daemon and is remotely controlled from a monitor in Georgia Tech using XML-RPC remote function calls. To conduct the experiment, we let each node generate one packet per 10 milliseconds to every other nodes and count the pairs of misordered packets caused by an overlay topology reconfiguration. To reconfigure the topology, the monitor sends out XML-RPC calls simultaneously to all nodes, commanding them to update their neighboring status and routing tables, using one of two protocols. In Protocol A, each nodes updates its routing table immediately after receiving the XML-RPC calls. Due to variation in their network distance from the monitor, the routing tables are actually updated at different times at different nodes. In Protocol B, the monitor provides a future time point in the XML-RPC calls and asks all nodes to synchronize their updating to that time point. The figures show the resulting amount of misordering for each end-to-end flow when the overlay makes transition from one degree-4 topology to another one. The figures show that Protocol A incurs less flow disturbance than B. But in both protocols, the disturbance on flows is significant and system-wide.

The formulation of the reconfiguration cost in topology transition naturally varies for different overlay network implementations (e.g., protocol for the transition), and it is not our goal to deal with each specific implementation in this paper. Instead, at this stage, we are most interested in understanding the dynamic overlay topology design problem in a more general manner. We use the total number of overlay links that need to be changed during a overlay transition as a general approximate metric for the reconfiguration cost. Formally, the topology reconfiguration cost metric is

$$\mathrm{g}(\mathcal{T}_{old}, \mathcal{T}_{new}) = \sum_{u,v} |\mathcal{T}_{old}(u,v) - \mathcal{T}_{new}(u,v)| \qquad (4)$$

where $\mathcal{T}_{old}, \mathcal{T}_{new}$ are the old and new overlay topology respectively[2]. The total reconfiguration cost over a time horizon Δt, denoted by $COST_r(\Delta t)$, is the sum of the reconfiguration costs incurred for all overlay topology transitions that happen during Δt. The reconfiguration policy construction methods developed in this paper allows plugging in other formulations of the function $g(\cdot)$. In Section V-B, we test the use of some other formulations on the performance of our policy construction methods and find that the performance of resulting reconfiguration policies remains unchanged.

[2]System specific weighting factors are absorbed in β in Eq.5.

Overall Cost: The overall cost of using the overlay over a time horizon Δt is the sum of both the occupancy cost and the reconfiguration cost incurred in the period. Formally,

$$COST(\Delta t) = COST_o(\Delta t) + \beta \cdot COST_r(\Delta t) \qquad (5)$$

where factor $\beta \in [0, 1]$ reflects the relative weight of reconfiguration cost and occupancy cost; its actual value depends on the implementation details, particularly how the topology transitions are performed and how different performance/cost factors (e.g., data loss rate, retransmission, performance of end-systems) are evaluated against each other. The long-run average of the overall cost is

$$\liminf_{\Delta t \to \infty} \frac{COST(\Delta t)}{\Delta t} \qquad (6)$$

Topology Reconfiguration Policy: Given the communication patterns, $X(t)$, a *reconfiguration policy*, $Y(t)$, is essentially a set of rules specifying when and how the overlay topology should be reconfigured. The following are three examples.

- Policy 1: Whenever the communication pattern changes, the overlay topology should be reconfigured to a random one.
- Policy 2: Every 10 minutes, the overlay topology should be reconfigured to the static-optimal one for the current communication pattern.
- Policy 3: Whenever the communication pattern changes, the overlay topology should be reconfigured to static-optimal one for the new communication pattern.

In this paper, we concentrate on the class of reconfiguration policies with the following properties:

- Reactive. A change in overlay topology can only be triggered by a change in communication pattern. Policies 1 and 3 have this property while Policy 2 does not.
- Memoryless. Assume the overlay topology changes at time t, the choice of the new overlay topology $Y(t+\delta)$ depends only on the current communication pattern $X(t)$, the current overlay topology $Y(t)$ and the new communication pattern $X(t+\delta)$. All the three example policies above have this property.
- Deterministic. Given $X(t)$, $Y(t)$ and $X(t+\delta)$, the choice of $Y(t+\delta)$ is deterministic rather than probabilistic. Policies 2 and 3 have this property while Policy 1 does not.

A reconfiguration policy in this class is essentially a function that maps each possible triple of current communication pattern $\mathcal{C}_{old}$, current overlay topology $\mathcal{T}_{old}$ and new communication pattern $\mathcal{C}_{new}$ to a new topology $\mathcal{T}_{new}$. It is possible that

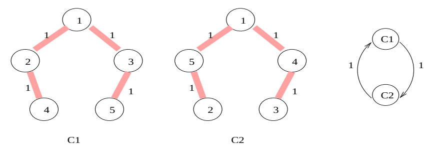


Fig. 4. $X(t)$ With Two Communication Patterns

$\mathcal{T}_{new} = \mathcal{T}_{old}$, in which case the overlay topology remains the same despite a change in communication pattern. The ideal goal is to find the *optimal reconfiguration policy* that can minimize Eq.6 for a given model of $X(t)$. The problem is NP-hard because even the static version of the problem (described in Section II-A) is NP-Hard. Please refer to Appendix A and [21] for a detailed discussion.

C. Methodology

Our methodology is, therefore, to start by observing the general properties and structures of the optimal reconfiguration policy on small, solvable cases, and then use the observations as heuristics to solve the problem for large cases. Particularly, in Section III, we study systems that have a small number of nodes and in which $X(t)$ is a continuous Markov process, intending to understand, for example, how the optimal policies are affected by various characteristics of $X(t)$ and the level of reconfiguration cost. Then in Section IV, we use our observations as heuristics and propose heuristic-based solutions that are applicable to problems with large number of nodes or with non-Markovian $X(t)$.

Note that for systems where $X(t)$ is a continuous Markov decision process, the dynamic topology design problem can be modeled as a continuous time *Markov decision process* [22]. Each state in the decision process consists of one communication pattern from $X(t)$ and a feasible overlay topology. A policy of the Markov decision process, corresponding to a topology reconfiguration policy in the original problem, is composed of a set of decisions, one at each state. When the number of nodes in the system is small (e.g., no more than 7), the number of states is still manageable and the decision process can be practically solved using Howard's policy iteration algorithm [22]. Please refer to [21] for a detailed discussion.

III. Properties and Structure of Optimal Reconfiguration Policies

Our goal in this section is to understand the structure and properties of the optimal overlay reconfiguration policies. To that end, we experiment on a sample network overlay with 5 nodes and present our findings in this section.

Reconfiguration Aggressiveness: Fig. 4 shows a simple communication requirement transition model with two communication patterns $\mathcal{C}_1$ and $\mathcal{C}_2$. We set $d(u_i, u_j) = 1$ for any pair of nodes u_i and u_j. The degree bound of the overlay networks is set to 2, so there are 12 feasible topologies[3]. We find that the system keeps the same optimal policy when we gradually increase the weight of reconfiguration cost(β in Eq. 5), until it reaches certain thresholds. Fig. 5 shows the optimal policies when we set the β to different values. The system's state is denoted by a tuple of a communication pattern and an overlay topology. An arrow pointing from $<\mathcal{C}_{old}, \mathcal{T}_{old}>$ to $<\mathcal{C}_{new}, \mathcal{T}_{new}>$ indicates that the optimal policy reconfigures the overlay topology to $\mathcal{T}_{new}$ when it is in state $<\mathcal{C}_{old}, \mathcal{T}_{old}>$ and the communication pattern changes to $\mathcal{C}_{new}$. We observe that the system, starting from any state, steers itself into a recurrent chain of states, possibly after a few steps of initial topology reconfiguration, and stays within the chain (The corresponding chains are shown Fig. 6. We omit the figure for $\beta = 10000000$ since it is the same as Fig. 5.(d).) We term the chain an *optimal reconfiguration chain* of the optimal policy. Note that an optimal policy could have more than one chain.

Not all feasible overlay topologies show up in the chains. The number of distinct topologies in a chain generally indicates the system's aggressiveness. When the weight of reconfiguration cost is low (e.g., $\beta = 0$ or $\beta = 0.3$), each optimal reconfiguration chain has two distinct topologies and the system always reconfigures the topology whenever the communication pattern changes. When the weight is high (e.g., $\beta = 0.4$ or $\beta = 10000000$), the system becomes conservative and tries to avoid reconfiguration cost, and each optimal reconfiguration chain, therefore, contains only one topology. The existence of more one than chain in an optimal reconfiguration policy is also related to the system's aggressiveness: the system prefers the "nearest" optimal reconfiguration chain that it can reach with minimum initial reconfiguration cost. Our calculation shows that the initial reconfiguration cost for different initial states varies little unless the value of β is high. In the extreme case where $\beta = 10000000$, the potential initial reconfiguration cost is so high that the system simply sticks with its initial overlay topology.

Fig. 7 presents a more complex system whose $X(t)$ has six communication patterns. Fig. 8.a plots the number of distinct overlay topologies in the optimal reconfiguration chains, the reconfiguration cost (after weighting with β), the occupancy cost and the overall cost, respectively[4]. The figure shows that the system's sensitivity to the weight of transition cost is discrete and threshold-based: the whole range of β is divided into ranges by some thresholds; the system's strategy stays basically the same when the value of β stays within the same range but changes abruptly when the value of β crosses into another range. Within the same range, the system's aggressiveness remains at the same level, the occupancy cost remains at the same level, and the reconfiguration cost (before weighting with β) remains the same. When the value of β crosses up into another range, however, the system's aggressiveness drops into a lower level, the occupancy cost jumps up to a higher level, and the reconfiguration cost drops to a new level. The plot for the overall cost, however, is always continuous and monotonic.

[3]There are actually 72 overlay topologies that are connected and with degree bound 2. For simplicity of presentation, we only count the topologies in which each node's degree is *exactly* 2.

[4]If there are multiple optimal reconfiguration chains in the optimal policy, we take the expected value, assuming the system starts from a random state with equal probability for all states.

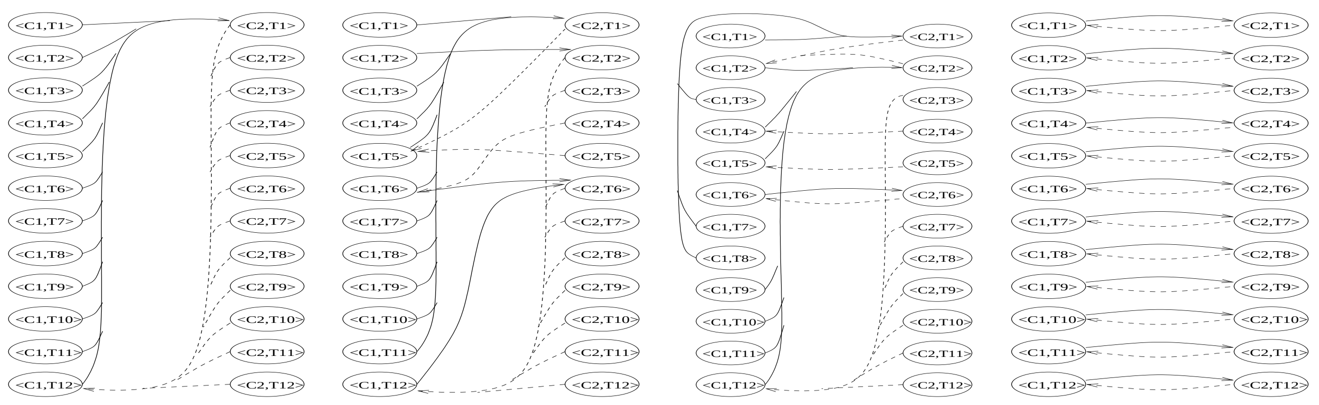

(a)$\beta = 0$ (b) $\beta = 0.3$ (c) $\beta = 0.4$ (d) $\beta = 10000000$

Fig. 5. Optimal Policies. Solid arrows and dashed arrows show how the overlay topology is reconfigured when the communication pattern changes from $\mathcal{C}_1$ to $\mathcal{C}_2$ and from $\mathcal{C}_2$ to $\mathcal{C}_1$, respectively.

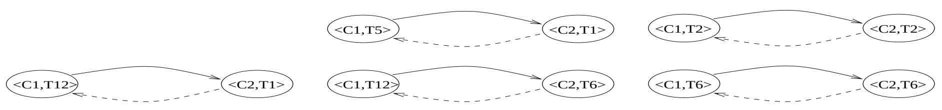

(a) $\beta = 0$ (b) $\beta = 0.3$ (c) $\beta = 0.4$

Fig. 6. Optimal Reconfiguration Chains. Solid arrows and dashed arrows have the same meanings as in Fig. 5.

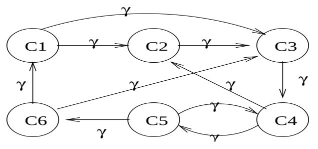
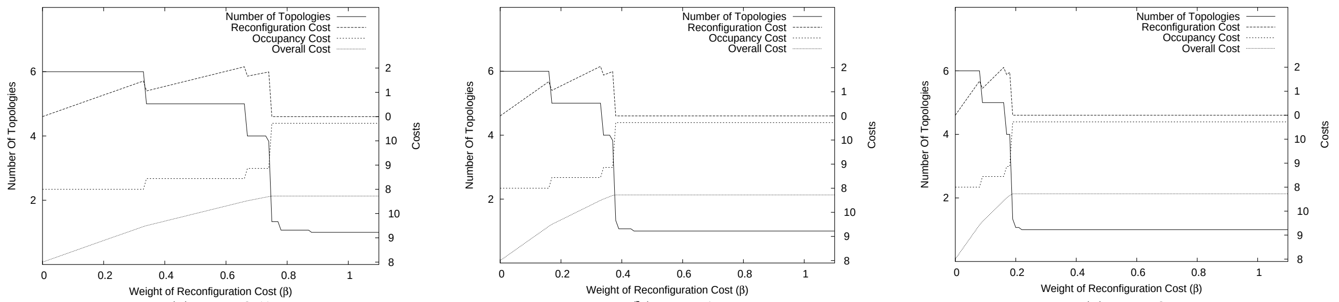


Fig. 7. $X(t)$ with Six Communication Patterns



(a) $\gamma = 0.5$ (b) $\gamma = 1$ (c) $\gamma = 2$

Fig. 8. Trends of Aggressiveness and Costs

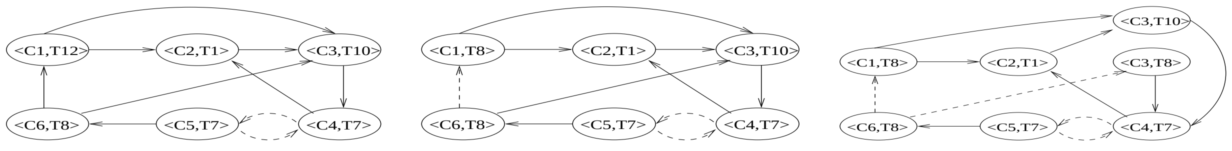

(a)$\gamma = 0.5, \beta = 0.26$ (b)$\gamma = 0.5, \beta = 0.7$ (c) $\gamma = 0.5, \beta = 0.73$

Fig. 9. Optimal Reconfiguration Chains. Solid arrows represent transitions between states with different overlay topology, while dashed arrows represent transitions between states with the same overlay topology.

We also find that the thresholds and the division of β ranges depend on the level of transition rates between the communication patterns. Fig. 8.b and c presents the graphs when the transition rates are at two other levels. They show that the β thresholds move to the left when the transition rates are scaled up; the higher the transition rates are, the less aggressive the system is.

Internal Structure of Optimal Reconfiguration Chains: The optimal reconfiguration chains also have internal structure. Fig. 9 presents some optimal reconfiguration chains for the communication requirement transition model shown in Fig. 7. In Fig. 9.a, $\mathcal{C}_4$ and $\mathcal{C}_5$ share the same overlay topology and therefore there is no topology reconfiguration when the system make a transition between them. In Fig. 9.b, $\mathcal{C}_6$ and $\mathcal{C}_1$ are also sharing the same overlay topology. Informally, we refer to the set of all communication patterns that share the same overlay topology in the optimal reconfiguration chain as a *cluster*. When each communication is associated with only one overlay topology in the optimal reconfiguration chain, the clusters are mutually exclusive, as those in Fig. 9.a and b. In such cases, the formation of clusters in an optimal reconfiguration chain can exactly define the chain: the system keeps its overlay topology when making a transition between communication patterns in the same cluster and reconfigures when it crosses clusters; when a reconfiguration occurs, the system always reconfigures to the overlay topology shared by the next cluster. There are cases, however, that a communication pattern may be associated with multiple overlay topologies in the optimal reconfiguration chain. Fig. 9.c shows an example in which $\mathcal{C}_3$ belongs to two clusters, one including only itself, another one including also $\mathcal{C}_1, \mathcal{C}_6$. When clusters overlap, the formation of clusters in an optimal reconfiguration chain cannot exactly define the chain. Our experiments, though, show that most communication patterns are associated with only one overlay topology in optimal reconfiguration chains.

We observe that the formation of clusters within an optimal reconfiguration chain is affected by two categories of factors. Global factors, such as the weight of the reconfiguration cost and the level of transition rates, affect the number of distinct overlay topologies, as previously discussed, and, therefore, the number of clusters. In another category are local factors that affect whether two given individual communication patterns will be put into the same cluster. We identify three important local factors in our experiments, namely, the *imbalance of the occupancy time* of two communication patterns, their *level of coupling* and their *similarity*. Due to space limitations, we skip detailed discussion. Please refer to [21] for the definitions of these factors and their effects.

Summary In summary, from the case studies we understand the *structure of the optimal policy*: The optimal policies are composed of one or more optimal reconfiguration chains. The system steers itself into one of the optimal reconfiguration chains after a few steps of initial reconfiguration if it follows the optimal reconfiguration policy. Within an optimal reconfiguration chain, communication patterns form clusters, defined by a distinct topology in the chain. Clusters can be overlapping or exclusive but most communication patterns are associated with only one overlay topology. Topology reconfiguration is triggered only when the system makes a transition across clusters. Two categories of factors affect the formation of clusters: global factors affect the number of clusters while local factors affect whether two individual communication patterns fall into the same cluster.

From the above case studies, we also learn the *structure of the problem* of constructing an optimal reconfiguration policy, which roughly divides the whole range of β into three areas:

1) Lower Extreme Area: β is lower than a threshold β_0. The reconfiguration cost is trivial and the optimal policy is the one that always reconfigures the overlay topology to the static-optimal one for the next communication pattern. We call this policy the *Always Change Policy (ACP)*.
2) Upper Extreme Area: β is lower than another threshold β_1. The system is extremely conservative and always keeps its overlay topology unchanged (or at least after an initialization period). The reconfiguration cost is zero and the optimal policy is the one that minimize the overall occupancy cost. We call this policy the *Never Change Policy (NCP)*.
3) Middle Area: β is between β_0 and β_1. The system's behavior is fine tuned. This area can be divided into more threshold-based ranges. Simple policies, such as ACP and NCP policies, are not good and we need a more complicated method for constructing the optimal policy when the value of β is in this area.

IV. Constructing Reconfiguration Policies for Large Systems

When the system has a large number of nodes or the process of transition among communication patterns is not Markovian, we cannot obtain the optimal reconfiguration policies by solving a Markov decision process. In this section, we propose methods for constructing good reconfiguration policies for these more general systems. We develop heuristic methods that reflect the properties and conform to the structures we observed in the optimal policies in Section III.

A. Always-Change Policy and Never-Change Policy

The Always-Change Policy is simple: the system always reconfigures to the static-optimal topology for the next communication pattern whenever the communication pattern changes. The subroutine for finding the static-optimal topology is discussed in Appendix A.

The Never-Change Policy never changes the overlay topology. Since the reconfiguration cost is zero, the optimal policy is one that minimizes the overall occupancy cost. Assume there are N distinct communication patterns in $X(t)$ and the percentage of occupancy time for communication pattern $\mathcal{C}_i$ is π_i, then the long-run average of the overall cost is:

$$\liminf_{\Delta t \to \infty} \frac{COST(\Delta t)}{\Delta t} = \sum_{i=1}^{N} \pi_i \cdot f(\mathcal{C}_i, \overline{\mathcal{T}})$$

$$= \sum_{i=1}^{N} \pi_i \sum_{u,v \in V} \mathcal{L}^{u,v}(\overline{\mathcal{T}}) \cdot \mathcal{C}_i(u,v) = f(\sum_{i=1}^{N} \pi_i \mathcal{C}_i, \overline{\mathcal{T}}) \tag{7}$$

where $\overline{\mathcal{T}}$ is the common overlay topology in the NCP policy. Eq.7 shows that to minimize the long-run average of the overall cost, $\overline{\mathcal{T}}$ is actually the static-optimal topology for a virtual communication pattern $\sum_{i=1}^{N} \pi_i \mathcal{C}_i$ and can be found using the subroutine in Appendix A.

B. Cluster-Based Policies

Inspired by the properties we observed in Section III, we propose Cluster-Based Policies(CBP) as heuristic approximation to the optimal reconfiguration policy. The basic idea of constructing CBP policies is to mimic the structure of the optimal ones:

1) We group the communication patterns into non-overlapping clusters and find a common overlay topology for each cluster; the common overlay topology for

a cluster is one that can minimize the overall occupancy cost for the whole cluster.

2) Whenever the a communication pattern transition occurs, if the next communication pattern is in the same cluster as the old one, the system keeps the current overlay topology, otherwise it switches to the common overlay topology of the newly entered cluster.

The CBP policies maintain the essential structure of the real optimal policy but simplify it in the following terms: first, there is only one optimal reconfiguration chain in the CBP policies; second, every communication pattern is associated with only one overlay topology (and therefore only one cluster) and all clusters in CBP policies are non-overlapping. The simplification allows us to construct the CBP policies efficiently. Performance evaluation in Section V, however, shows that CBPs are almost as good as the real optimal policies.

Fig. 10 sketches the algorithm for constructing CBP policies. The algorithm needs three types of input about $X(t)$: the set of distinct communication patterns, $\{\mathcal{C}_i | 1 \leq i \leq N\}$; the average percentage of occupancy time that the system runs with $\mathcal{C}_i$, denoted by π_i; and the average number of transition from $\mathcal{C}_i$ to $\mathcal{C}_j$ per unit of time, denoted by b_{ij}. The value of π_i's and b_{ij}'s can be estimated in a real system via statistical modeling methods. Specially, for applications in which the transitions between communication patterns can be modelled with Markov or Semi-Markov processes, π_i's and b_{ij}'s can be derived from the transition model by calculating the stationary probabilities of the processes.

As most k-means style clustering algorithms, the proper number of clusters needs to be estimated beforehand, based on the weight of reconfiguration cost. The proper number can also be found by trying out all numbers in a certain range (in worst case, from 1 to the total number of communication patterns) and choosing the one that generates the best result.

Assume we are clustering N communication patterns into L clusters. The algorithm starts by randomly assigning the N communication patterns into L clusters. During each iteration, the algorithm reassigns each communication pattern to the cluster that is most "suitable" for it. The iterations stop when the clustering converges and the policy corresponding to the final clustering is returned as the result of the algorithm.

In Fig. 10, lines (a1) and (a2) involve a procedure that converts a clustering to its corresponding cluster-based reconfiguration policy and line (b) involves a procedure that calculates the cost of the policy. We describe the two procedures in detail in the remainder of this section.

Converting Clustering To Policy The major task in this procedure is to find one common overlay topology for each cluster. Once the common overlay topologies are decided, the policy is fully defined by the following rule: whenever the system makes a transition between communication patterns within the same cluster, the overlay topology is not changed; whenever it makes a transition across clusters, it reconfigures the overlay topology to the common overlay topology for the newly entered cluster.

```
PROCEDURE CONSTRUCT-CBP-POLICY
Initialize-Clusters :
  randomly assign $\mathcal{C}_1$, $\mathcal{C}_N$, ... $\mathcal{C}_N$ to $S_1$, $S_2$, ... $S_L$ ;
Adjust-Clusters :
  for each communication pattern $\mathcal{C}_i$, $1 \leq i \leq N$:
    find the most suitable cluster $S^*$ for $\mathcal{C}_i$:
      for each cluster $S_j$, $1 \leq j \leq L$:
        temporarily move $\mathcal{C}_i$ to $S_j$;
        convert the clustering ($S_1$, $S_2$, ... $S_L$) to policy $P$;    — line (a1)
        evaluate the cost of policy $P$;                                   — line (b)
      Let $\mathcal{S}^*$ is the clustering with the least cost in the above loop;
      move $\mathcal{C}_j$ to $S^*$;
Loop-Back :
  go back to Adjust-Clusters until the clustering does not change;
Post-Handle:
  convert the clustering ($S_1$, $S_2$, ... $S_L$) to policy $P$;          — line (a2)
  return policy $P$;
```

Fig. 10. Pseudo-code of Constructing Cluster-Based Polices

The common overlay topology for a cluster is one that can minimize the total occupancy cost of the cluster. Formally, consider a cluster containing communication patterns $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_s$, the common topology of the cluster is an overlay topology $\mathcal{T}$ that minimizes the occupancy cost of the whole cluster: $COST_c(\Delta t) = \sum_{i=1}^{s} \pi_i \Delta t \cdot f(\mathcal{C}_i, \mathcal{T})$. Similar to the logic in Section IV-A, this optimal common overlay topology is actually the optimal-static topology for a virtual communication pattern $\sum_{i=1}^{s} \pi_i \mathcal{C}_i$ and can be found using the subroutine in Appendix A.

Calculating Policy Cost: Assume $\mathcal{T}_i$ is the topology assigned to communication pattern $\mathcal{C}_i$ by the policy. Because in a CBP policy, each communication pattern is associated with only one overlay topology, we have: $\liminf_{\Delta t \to \infty} \frac{COST_p(\Delta t)}{\Delta t} = \sum_{i=1}^{N} \pi_i f(\mathcal{C}_i, \mathcal{T}_i) + \beta \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} g(\mathcal{C}_i, \mathcal{T}_i, \mathcal{C}_j, \mathcal{T}_j) \cdot b_{ij}$.

V. Performance Of Approximate Policies

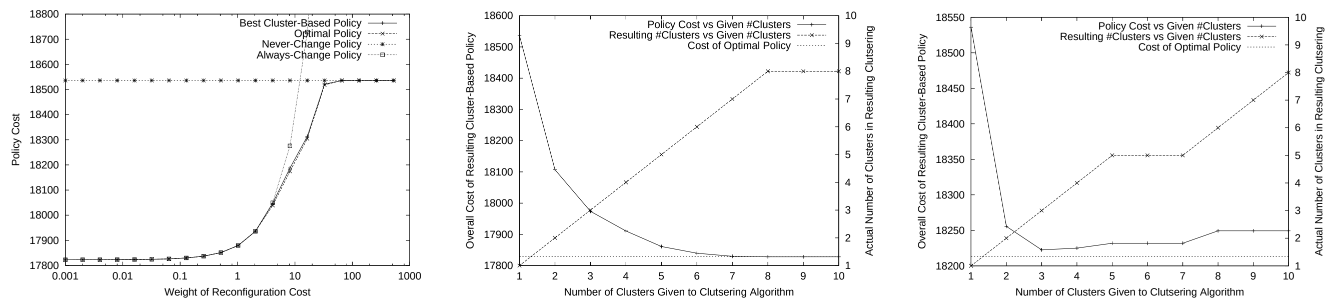
A. Performance of Constructed Policies — Small Networks

Table I shows the actual experimental parameters we use to generate a typical case problem.

Fig. 11.a compares the cost of the optimal policy, the ACP policy, the NCP policy and the *best* CBP policy for the same case constructed using the parameters in Table I. Among all CBP policies resulting from using different number of clusters in the clustering algorithm (K in Fig. 10), the best CBP policy is defined as the one whose cost is most close to that of the optimal policy. The y-axis represents the policy cost. The x-axis represents the spectrum of the weight assigned to the reconfiguration cost. The figure shows that the cost of the NCP policy does not change when the weight of reconfiguration cost is increased because the policy operate with a single policy and does not incur a reconfiguration cost. The cost of the NCP policy coincides with that of the optimal policy for a large β, which means that NCP policy is actually optimal when the weight of the reconfiguration cost is beyond a certain threshold. The figure also shows that the ACP policy is actually optimal when the weight of the reconfiguration cost is below a certain threshold but its cost increases very quickly when the weight of the reconfiguration cost is large. Finally, the figure shows that the CBP policies approximate well the optimal

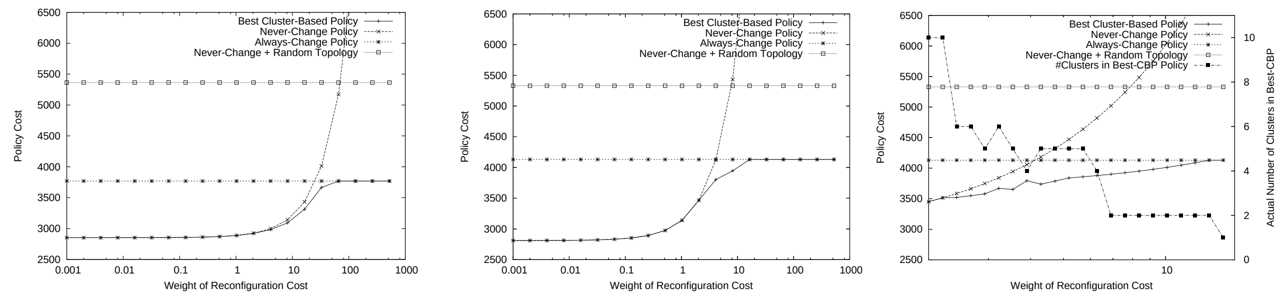
parameters	small network	large network
underlying native network	-	Internet graph generated using GT-ITM, 1400 nodes: 1 transit domain(200 nodes), 20 stub domains(60 nodes in each stub)
number of overlay nodes	5	40, randomly selected from the stub domains
overlay link cost matrix	random value in range [10,15]	number of hops in native network
number of communication patterns	10	10
number of transitions from each communication pattern	random integer in range [1,3]	random integer in range [1,3]
transition rates between communication patterns	random value in range [2,6]	1
data demand between pairs of overlay nodes in communication patterns	random value in range [0.1, 100]	1 for 15 percent of total pairs (randomly chosen), 0 for other pairs
degree bound on feasible overlay topologies	2	4
number of feasible overlay topologies	72	-

TABLE I
EXPERIMENT PARAMETERS



(a) Policies (b) $\beta = 0.1$ (c) $\beta = 10$

Fig. 11. Performance of Approximate Policies Against Optimal Policies — Experiment With 5 Nodes



(a) Markovian Cases (b) Non-Markovian Cases (c) Magnification of (b)

Fig. 12. Performance of Approximate Policies Against Optimal Policies in — Experiment With 40 Nodes

policy when the weight of the reconfiguration cost is in its middle range.

For the same cases, figures Fig.11.(a)-(b) show how the resulting clustering and the performance of corresponding CBP policies are affected by the number of clusters input to the clustering algorithm. Figures (a) and (b) are for two different weights on reconfiguration cost (β in Eq. 5), respectively. The x-axis represents the number of clusters, K, that we input to the algorithm. The right y-axis represents the actual number of *non-empty* clusters in the resulted clustering; the figure shows that clustering algorithm does not necessarily use up all the clusters if it finds that less clusters give better result. The right y-axis represents the cost of resulting cluster-based policy. The figures show that a good estimation of the proper number of clusters can help the algorithm to find the best cluster-based policy whose cost is most close to that of the optimal policy.

B. Performance of Constructed Policies — Large Networks

We also experiment on problems with a larger number of nodes. We generate a native network using GT-ITM topology generator [23] and select some native nodes from the stub domains as overlay nodes. We assume the transitions between communication patterns are Markovian. Table I shows the actual experimental parameters we use to generate a typical case problem.

Fig. 12.a presents the costs of different policies for this problem. The plots of NCP, ACP and best-CBP are very similar to those in Fig. 11.a. The figure also presents the cost of a naive policy that randomly chooses a feasible overlay topology and sticks with it, a policy that is actually used

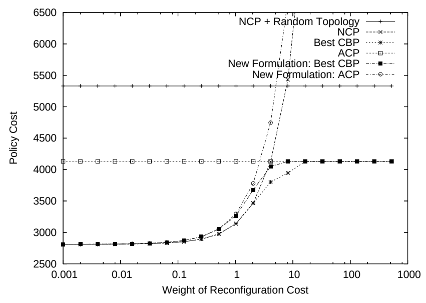
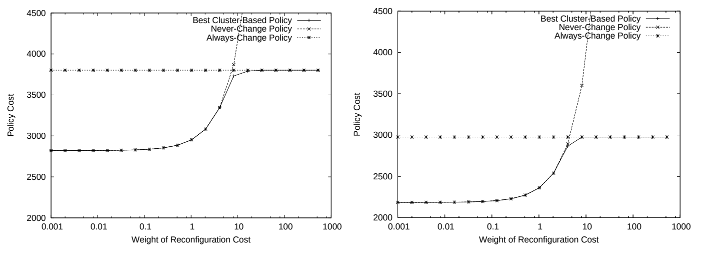
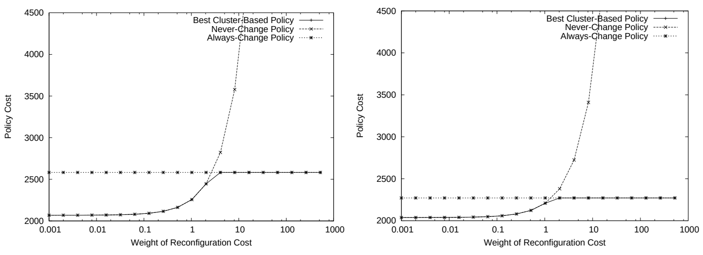


Fig. 13. Using Other Formulations of Reconfiguration Cost



(a) Degree = 4 (b) Degree = 6



(c) Degree = 8 (d) Degree = 12

Fig. 14. Degree of Overlay Networks versus Cost of Reconfiguration Policies

by many overlay applications. The cost is the mean over 20 random feasible overlay topologies. The figure shows that NCP, ACP and CBP policies perform much better than the naive policy in their applicable ranges.

Our policy construction methods are applicable to non-Markovian cases. Fig. 12.b presents the performance of NCP, ACP, best-CBP and the naive policy for a non-Markovian problem. Most parameters for the experiment are the same as those given in Table I. The transitions between communication patterns, however, are semi-Markovian this time: we first randomly assign the transition probabilities in the embedded Markov chain, and then, for each transition, we assign the average pre-transition occupancy time to a random value in the range $[0.2, 1]$.

The plots in the figure are very similar to those in Fig. 11.b: the NCP, ACP and CBP policies perform much better than the naive policy in their applicable ranges and the CBPs adapt well when the weight of reconfiguration cost varies. Fig. 12.c magnifies Fig. 12.b for the portion where β is in range [2,16] and CBPs outperform both NCP and ACP. Both the policy-cost and the number-of-cluster plots for the best-CBPs show general trends similar to those in Fig. 8. This reaffirms our belief that the way we construct CBPs reflects the basic structures of the optimal policies and manages to achieve the essential tradeoff between the occupancy cost and the reconfiguration cost inherent in the dynamic overlay topology reconfiguration problems. Note that there are ”ripples”, however, in both plots. This is due to the fact that the constructed CBPs are merely approximate alternatives to the optimal policies and the fact that our clustering algorithm and simulated annealing algorithm are also approximate. The ripples, however, are minor; the constructed CBPs perform consistently in approximating the optimal policies.

Finally, in our policy construction methods, the formulation of function $g(\cdot)$ in Eq.4 can be substituted with other formulations without affecting the applicability of the algorithms. Fig. 13 shows the cost of resulting policies when another formulation

$$\sum_{u,v} w(u,v) \cdot |\mathcal{T}_{old}(u,v) - \mathcal{T}_{new}(u,v)|$$

is used, where $w(u,v)$ weights the overlay link (u,v) with the number of end-to-end minimum-operation-cost paths that cross the link in topology $\mathcal{T}_{old}$. For NCP and random-NCP policies, the use of a new formulation does not affect the resulting policies and their cost. For ACP, the policies themselves are not affected; their costs changed, reflecting the new formulation, but the difference is trivial when β is in its lower range (where ACP policies are applicable). So only CBPs are affected. Fig. 13 compares the experiment results from the new formulation (Formulation 2) to those in Fig. 12.a (the two sets of experiments use the same set of parameters except for the reconfiguration cost formulation). The figure shows that the CBPs perform similarly under the new formulation. They generally become less aggressive though, because the new formulation numerically generates higher reconfiguration cost than the old one.

C. Effect of Degree Bound

As discussed in Section II, the degree bound of an overlay characterize the types of link maintenance costs that are not directly related to the bandwidth consumption for delivering user data. It reflects an overlay network provider’s aversion to these types of cost. In this section, we are interested in understanding how dynamic topology reconfiguration policies are affected by the degree bound, and the other way, what dynamic topology reconfiguration implies to the choice of degree bound.

Fig. 14 shows how the degree of an overlay network affects the cost of the reconfiguration policies. Most parameters for the experiment are the same as those given in Table I. The transitions between communication patterns are Markovian. From each communication pattern, the system can make up to 4 transitions and the transition rates are random values in range [1,5].

The four figures show the cost of the NCP, ACP and best-BCP policies when we set the degree bound of the overlay network to 4,6,8, and 12, respectively. We have three interesting observations from these experiments. First, the cost of the policies decreases when the degree bound increases. Intuitively, on one hand, with larger degree bound, there are more feasible overlay topologies; the system may be able to find better overlay topologies with lower occupancy cost. On the other hand, with larger degree bound, the system may be able to establish new overlay links without having to tear down the old ones; this reduces the reconfiguration cost. This suggests the overlay network designers use larger degree bound when possible. Second, however, the gain of using a larger degree bound varies. In Fig. 14, when the degree bound is increased from 4 to 6, both the cost of ACP and that of the NCP decrease significantly. When the degree bound is increased from 6 to 8, however, only the cost of the NCP decreases significantly. This suggest that overlay network designers should not pursue larger degree bounds blindly. The benefit may be limited, especially when the weight of reconfiguration cost is at its low range — by allowing the overlay topology to be dynamically reconfigured, the need for a higher degree bound can be reduced. Finally, consider the space below the plot of the best-CBP and above the plot of the ACP in the figures. The space indicates how much the CBPs can outperform the ACP and the NCP in their applicable range. Notice that the space becomes narrower when the degree increases. This suggests that the CBPs outperform ACP and NCP most when the overlay networks have a very restrictive degree bound.

VI. Concluding Remarks

We have studied the problem of dynamically reconfiguring the topology of an overlay network in response to the changes in the communication requirements. We have considered two costs of using an overlay: the occupancy cost and the reconfiguration cost. The ideal goal is to find the optimal reconfiguration policies that can minimize the potential overall cost of using an overlay. The problem is NP-hard and good approximate policies are called for. We have studied the properties of the optimal policies through experiments on small systems. We then used our observations as heuristics and proposed methods of constructing approximate policies.

Our studies have shown that dynamic overlay topology reconfiguration can significantly reduce the overall cost of using an overlay. It allows an overlay network provider to take advantage of the superior configurability generally provided by the overlay networks. Our studies have also shown that the optimal reconfiguration policies have structure and this allows us to construct good-performance approximation policies by mimic the observed structure of the optimal reconfiguration policies. In many scenarios, a simple topology reconfiguration policy (ACP or NCP) can actually serve as a optimal policy. In other scenarios, the more complicated CBP policies can be used to approximate the optimal policies. Our studies have shown that our methods of constructing ACP, NCP and CBP policies are applicable to various models for the dynamics of the communication requirements and formulations for the reconfiguration cost, and the resulting policies have good performance. Finally, our studies have shown that, dynamic topology reconfiguration helps to reduce the need to very high node-degrees in overlay topologies, which potential incurs high overlay link maintenance cost.

References

[1] Larry Peterson, Scott Shenker, and Jonathan Turner, “Overcoming the Internet Impasse Through Virtualization,” in *Proceedings of the 3rd ACM Workshop on Hot Topics in Networks (HotNets-III)*, November 2004.

[2] J. Touch, Y. Wang, L. Eggert, and G. Finn, “A virtual internet architecture,” Technical report, ISI, ISI-TR-2003-570, March 2003.

[3] Nick Feamster, Hari Balakrishnan, Jennifer Rexford, Aman Shaikh, and Jacobus van der Merwe, “The case for separating routing from routers,” in *FDNA ’04: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*. 2004, ACM Press.

[4] Micah Beck, Terry Moore, and James S. Plank, “An end-to-end approach to globally scalable programmable networking,” in *FDNA ’03: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*. 2003, ACM Press.

[5] Yang-Hua Chu, Sanjay G. Rao, and Hui Zhang, “A case for end system multicast,” in *ACM SIGMETRICS 2000*, Santa Clara, CA, June 2000, ACM, pp. 1–12.

[6] S. Banerjee, B. Bhattacharjee, C. Kommareddy, and G. Varghese, “Scalable application layer multicast,” in *Proceedings of the ACM SIGCOMM 2002*, New York, 2002, pp. 205–220.

[7] Dimitris Pendarakis, Sherlia Shi, Dinesh Verma, and Marcel Waldvogel, “ALMI: An application level multicast infrastructure,” in *Proceedings of the 3rd USNIX Symposium on Internet Technologies and Systems (USITS ’01)*, San Francisco, CA, USA, Mar. 2001, pp. 49–60.

[8] Jorg Liebeherr and Tyler K. Beam, “Hypercast: A protocol for maintaining multicast group members in a logical hypercube topology,” in *NGC ’99: Proceedings of the First International COST264 Workshop on Networked Group Communication*. 1999, pp. 72–89, Springer-Verlag.

[9] Robert Fink, “Network integration — boning up on IPv6 — the 6bone global test bed will become the new Internet,” *Byte Magazine*, vol. 23, no. 3, pp. 96NA–3–96NA–8, Mar. 1998.

[10] Hans Eriksson, “Mbone: the multicast backbone,” *Commun. ACM*, vol. 37, no. 8, pp. 54–60, 1994.

[11] Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe, “A blueprint for introducing disruptive technology into the internet,” *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 59–64, 2003.

[12] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris, “Resilient overlay networks,” in *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP-01)*, New York, 2001, pp. 131–145.

[13] A. D. Keromytis, V. Misra, and D. Rubenstein, “SOS: Secure overlay services,” in *Proceedings of the ACM SIGCOMM 2002*, New York, 2002, pp. 61–72.

[14] Zhenhai Duan, Zhi-Li Zhang, and Yiwei Thomas Hou, “Service overlay networks: Slas, qos, and bandwidth provisioning,” *IEEE/ACM Trans. Netw.*, vol. 11, no. 6, pp. 870–883, 2003.

[15] S. L. Vieira and Jörg Liebeherr, “Topology design for service overlay networks with bandwidth guarantees.,” in *IWQoS*, 2004, pp. 211–220.

[16] A. Sundararaj, A. Gupta, and P. Dinda, “Dynamic topology adaptation in virtual networks of virtual machines,” in *LCR 2004: Proceedings of the Seventh Workshop on Langauges, Compilers and Run-time Support for Scalable Systems*.

[17] Joe Touch, “Dynamic internet overlay deployment and management using the x-bone,” *Computer Networks*, vol. 36, no. 2-3, pp. 117–135, 2001.

[18] G. N. Rouskas and M. H. Ammar, “Dynamic reconguration in multihop WDM networks,” *Journal of High Speed Networks*, 1995.

[19] I. Balldine and George Rouskas, “Dynamic load balancing in broadcast WDM networks with tuning latencies,” in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, March/April 1998.

[20] Zhi Li and Prasant Mohapatra, “Impact of topology on overlay routing service.,” in *INFOCOM*, 2004.

[21] Jinliang Fan and Mostafa H. Ammar, “Dynamic topology reconfiguration of overlay networks: Structure and approximation of optimal policies. http://www.cc.gatech.edu/˜jlfan/policy-techreport.pdf,” Technical report, Georgia Institute of Technology, 2004.

[22] Ronald A. Howard, *Dynamic Programming and Markov Processes*, The MIT Press, Cambridge, Massachusetts, 1960.

[23] Kenneth L. Calvert, Matthew B. Doar, and Ellen W. Zegura, “Modeling internet topology,” *IEEE Communications Magazine*, vol. 35, no. 6, pp. 160–163, June 1997.

[24] S. Kirkpatrick, “Optimization by simulated annealing: quantitative studies,” *Journal of Statistical Physics*, vol. 34(5/6), pp. 975–986, 1984.

[25] A. Anagnostopoulos, L. Michel, P. Van Hentenryck, and Y. Vergados, “A simulated annealing approach to the traveling tournament problem,” in *Proceedings CPAIOR’03*, Montreal, Canada, 2003.

```
PROCEDURE FIND-OPTIMAL-TOPOLOGY
Initialize :
  construct a random initial point and calculate its cost;
  let Optimal-Topology be the current topology;
  let Optimal-Cost be the current cost;
  estimate the initial temperature $T_0$ and let current temperature $T = T_0$;
Repeat phases until $T$ is close enough to zero:
  set the stop temperature $T_e$ of this phase to $\frac{1}{\phi}T$ ;      — $\phi > 1$
  Repeat moves until $T$ reaches $T_e$:
    randomly mutate the current topology using one of the operations;
    repeat the above step until the resulting topology is a connected one;
    evaluate the cost function $f$ of the new topology
    calculate the change of cost,$\Delta f$;
    if $\Delta f < 0$, accept the new topology; else accept with probability $e^{\frac{\Delta f}{T}}$;
    if $\Delta f < 0$, update Optimal-Topology and Optimal-Cost;
    let $T = \rho T$;      — geometric cooling; $\rho$ is typically in [0.95,1)
  let $T = \varsigma T_e$;      — reheating; $\phi > \varsigma > 1$
Output Optimal-Topology and Optimal-Cost;
```

Fig. 15. Pseudo-code of Finding Optimal-Static Topology Using Simulated Annealing

Appendix A: Finding Static-Optimal Overlay Topology for A Single Communication Pattern

In this appendix, we present a Simulated Annealing based algorithm for solving the problem of finding the static-optimal topology for a given single communication pattern. The problem is previously formulated in Section II-A and is NP-hard [21]. An heuristic algorithm for solving this problem is a necessary subroutine for constructing the NCP, ACP and CBP policies discussed in section IV. We only sketch the algorithm here. Please refer to [21] for a complete treatment (i.e., literature and performance evaluation).

Simulated Annealing is a global optimization meta-algorithm. Starting from an initial solution and an initial *temperature*, the meta-algorithm walks randomly in the solution space. Cost-decreasing moves are certainly accepted while cost-increasing ones are accepted only with a probability $P = e^{\frac{\Delta f}{T}}$, where Δf is the increase in the cost function f and T is the temperature. By decrementing and possibly incrementing the temperature following a deliberate *annealing schedule*, this probabilistic process will finally stabilize at a final solution. The overall structure of our algorithm is shown in Fig. 15.

Initial Point The algorithm starts from a random overlay topology that is connected and subject to degree bound K.

Searching for a Solution Finding another feasible solution in the neighborhood of the current feasible solution is the key task in the probabilistic walk-around. We accomplish the task by first mutating the current feasible overlay topology into a *slightly different* topology that still satisfies the degree bound and then verifying the connectivity of the new topology. We use the following mutating operations in the algorithm.

- Operation 1: This operation randomly chooses two nodes A and B whose degrees are less than K and adds an overlay link between the two nodes.
- Operation 2: This operation randomly chooses four nodes that form a local setup like Fig. 16.a and converts it to either Fig. 16.b or Fig. 16.c.

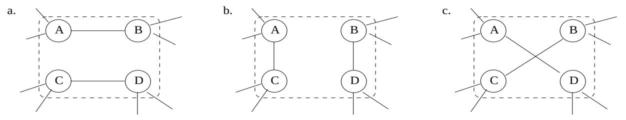


Fig. 16. Mutating Operation 2

- Operation 3: This is a derivative of Operation 2. This operation randomly chooses four nodes that form a local setup like Fig. 17.a and converts it to either Fig. 17.b or Fig. 17.c.

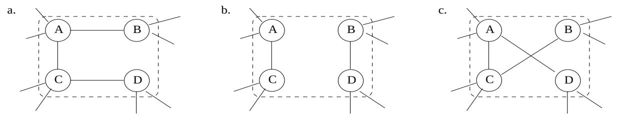


Fig. 17. Mutating Operation 3

These operations allow the algorithm to move towards any feasible solution gradually and continuously without ever leaving the space of feasible solutions.

Initial Temperature Kirkpatrick [24] suggested to set the initial temperature to one that results in an average acceptance probability of about 0.8 for uphill moves from the initial point. We estimate the initial temperature T_0 in the following way: we attempt a number of random cost-increasing moves, all from the initial point, observe the average increase in cost, $\overline{\Delta f}$, and then calculate the initial temperature T_0 by: $T_0 = \frac{\overline{\Delta f}}{\ln(0.8)}$.

Annealing Schedule In the literature, the choice of annealing schedule is quite problem specific. In several large combinatorial problems, researchers use geometric cooling for expediency yet get good result with the help of reheating [25]. We follow the same direction. We try different types of schedules and find the following one is good for our problem. The schedule is composed of phases. In the pth phase, starting from the initial value T_0^p, the temperature is multiplied by ρ each time the algorithm attempts a move (no matter if the move is accepted or rejected), i.e., $T_{k+1}^p = \rho T_k^p$, where ρ is typical a value between 0.95 and 1. When the temperature reaches down $\frac{1}{\phi}T_0^p$, where $\phi > 1$, the algorithm ends the current phase. It reheats the system by multiplying the temperature by ς, where $\varsigma > 1$, and enters the $(p+1)$th phase with a starting temperature $T_0^{p+1} = \frac{\varsigma}{\phi}T_0^p$. By choosing $\phi > \varsigma$, e.g., $\phi = 3$ and $\varsigma = 2$, the temperature goes down after each phase and eventually gets close to zero, where the annealing schedule ends and the algorithm exits.