SOLVING NETWORK TESTBED MAPPING PROBLEM WITH GENETIC ALGORITHM

Yi Liu, Kaiping Xiao and Yanping Li
Department of Computer, Xi'an Jiaotong University, Xi'an, Shannxi, 710049, China E-mail: Liuyi97@263.net

Abstract: To provide remote service and auto-configuration mechanisms, network emulation testbed needs to map the logic-topology, which the researchers require, to the testbed physical-topology. This network testbed-mapping problem is discussed and formulated in this paper; an evaluation function is also defined in which both the number of network links and bandwidth are considered. This paper also gives a solution for considering delay-node. Based on the model, a solver for the problem using Genetic Algorithm is implemented. Evaluation result shows that the genetic algorithm can find near-optimal solutions in reasonable time.

Key words: network emulation, genetic algorithm, network testbed

1. INTRODUCTION

Experiment plays an important role in researches of various network technologies. Considering the cost and complexity of network experimentations in live networks, network simulation and emulation are widely used. Meanwhile, traditional simulation tools, such as widely used ns-2, are more available and economical, due to their ease of installation, management and relatively low resource requirements. Unfortunately, simulation-based experiments may be deemed less convincing, due to their lack of fidelity to real-world environments. Compared to the live network and simulation, emulation provides a good trade-off between fidelity and cost.

The goal of network emulation is to build an environment nearing real-world network, which researchers require in an experimental network. The primary difference between emulation and simulation is: simulators are totally implemented in software, in which network topologies and links are simulated; as a contrast, emulators are commonly built on one or several inter-connected LANs, in which network topologies are implemented by setting virtual LAN (VLAN) on network switches, and network links directly correspond to physical links in the LAN, delay-nodes are inserted to emulate low-speed network links. A number of recently proposed testbeds have clearly demonstrated the advantage and promise of this approach, such as Emulab/Netbed, ModelNet, vBET, and PlanetLab.

To provide remote service and auto-configuration mechanisms, network emulation testbed needs to map the logic-topology, which the researchers require, to the testbed physical-topology. In this paper, we explore this network testbed mapping problem. After building a model of the problem, this paper implements a solver for the problem using Genetic Algorithm. Finally, the solver is evaluated and results are discussed.

The rest of this paper is organized as follows. Section 2 gives an overview of network testbed mapping problem. Section 3 builds a model of the problem. Section 4 solves the problem using genetic algorithm, a well-known general-purpose method for solving difficult problems. Section 5 evaluates the solver using 100 randomly generated samples. Section 6 concludes the paper.

2. OVERVIEW OF NETWORK TESTBED MAPPING PROBLEM

The kernel of the network emulation testbed is to simulate the network environment, which researchers require in an experimental network. One of the key challenges to the testbed is how to construct required network topology on the physical topology of the experimental environment. This work can be done either manually or automatically. When it is done by the system automatically, the system must have the ability to map the logic-topology that researchers require to the physical topology of the testbed, and the mapping process is called “network testbed mapping problem” [1] or “network mapping problem” [3].

The mapping process includes two aspects: mapping of nodes (hosts & routers) and mapping of network links. The first one determines which physical nodes are used to emulate the logic-nodes. The assigned physical node must satisfy the hardware requirements of the logic-topology, such as processor speed of host, or packet routing delay of routers, etc. The second aspect, mapping of network links is relatively complex since it involves the use of the delay node sometimes. For example, to simulate 64 Kbps link on a 100Mbps network, delay nodes must be inserted between two ends of the 100Mbps link, by running dedicated emulation software on the delay node, the emulated link can have the same characteristics with 64Kbps link, such as bandwidth, transport delay and packet loss rate. Another example to use delay node is emulating long delay links such as satellite communications in a LAN.

Figure-1 gives a simple mapping example. There are three hosts and one router in experimenter's logic-topology. The physical topology includes one switch and four PCs, each of them has three network interfaces. Figure-1(c) shows the mapping result: all the nodes in logic-topology are mapped to the PC nodes, network links are emulated by configuring VLANs on the switch. The real lines in Figure-1 represent used links while the dotted lines represent unused links. Three VLANs are configured in this mapping, to emulate links between router and node 1,2 and 3 respectively. VLANs will isolate these three links form each other.

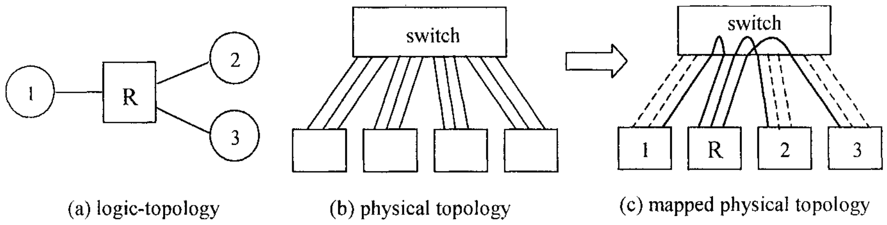

Figure 1. A Simple Mapping Example

To utilize resources more efficiently, emulation testbed should support as many experiments as possible simultaneously. This requires that the mapping of network topology should use as few resources as possible for an experiment, in order to reserve as many resources as possible for future experiments.

Network testbed mapping problem is finally turns to partitioning of graph. With different kinds of restrictions, the problem becomes NP-hard[7].

The main related researches about network testbed mapping problem include: [2] implements a solver using simulated annealing algorithm without considering the delay node; [3] presents a heuristic algorithm based on process allocating algorithms from parallel processing, with the aim of traffic-based load balancing among nodes.

3. MODELING OF THE PROBLEM

3.1 Modeling of the problem

The network topology in network testbed mapping problem can be expressed by graphs, in which vertices represent hosts & routers and graph edges represent network links.

Let graph *GP* and *GL* represent the physical topology of the testbed and experimenter's logic-topology respectively.

Graph *GL* is composed of node set $\{NL_i\}$ and link set $\{EL_{jk}\}$, $i,j,k=0,1,\dots,n-1$. A node, which corresponding to a host or a router, is represented by a tuple with each item reflects one configuration-parameter of the node. As a example, we use 4-tuple $< pl, ml, dl, ll>$ to represent a node in this paper, where *pl* is the processing capability, *ml* is the memory space, *dl* is the hard disk space, *ll* is the delay of routing. An element E_{jk} in link set represents an edge between node NL_j and NL_k, while the bandwidth of the network link is represented by the edge's weight *W*.

Similarly, graph *GP* is composed of the node set $\{NP_i\}$ and the link set $\{EP_{jk}\}$, $i,j,k=0,1,\dots,m-1$. The node in the graph is represented by the 4-tuple $<pp, mp, dp, lp>$.

According to the definitions above, the mapping from logic-topology *GL* to physical topology *GP* can be described as follows:

(1) **Mapping of nodes**: each node NL_i in *GL* is mapped to a NP_j in *GP*, and hardware configuration requirements of NL_i are satisfied by NP_j. That is:

$$f(NL_i) = NP_j,\ i=0,1,\dots n-1,\ j=0,1,\dots m-1$$
$$\text{and } pl_i \leqslant pp_j,\ ml_i \leqslant mp_j,\ dl_i \leqslant dp_j,\ ll_i \geqslant ll_j$$

(2) **Mapping of links**: there are two situations as follows.

- Network link EL_{ij} in *GL* is mapped to network link EP_{kl} in *GP*, while the two ends of EL_{ij} are mapped to two ends of EP_{kl}, and bandwidth requirement is satisfied by EP_{kl}. That is:

  $$g(EL_{ij}) = \{EP_{kl}\}$$
  $$\text{and } f(NL_i) = NP_k,\ f(NL_j) = NP_l \quad i,j=0,1,\dots n-1;\ k,l=0,1,\dots m-1$$
- Network link EL_{ij} in *GL* is mapped to multiple chained links in *GP*, while the two ends of EL_{ij} are mapped to two ends of those chained links, and each of those chained links satisfies bandwidth requirement of EL_{ij}. Nodes in the middle of the chain are all switching nodes. That is:

  $$g(EL_{ij}) = \{EP_{kl}, EP_{lo}, \dots, EP_{xy}\}$$
  $$\text{and } f(NL_i) = NP_k,\ f(NL_j) = NP_y \quad i,j=0,1,\dots n-1;\ k,l,o,\dots x,y=0,1,\dots m-1$$

(3) **Evaluation method of a mapping**:

As discussed in section 2, a mapping of network topology should use as few resources as possible for an experiment. According to this principle, the evaluation method can be defined as follows.

Supposing for a mapping scheme, the set of used network links in *GP* is:

$U = \{EP_{ij}, EP_{jk}, ..., EP_{xy}\}$ $i,j,...,x,y=0,1,...m-1$

The evaluation value of the mapping scheme can be calculated according to:

$$E = (\alpha \times NU) + (\beta \times \sum W_{ij}) \tag{3-1}$$

Where *NU* is the element number of set *U*; W_{ij} is the link's weight in set *U*; α and β are scaling values satisfying $\alpha + \beta = 1$, which are used to achieve trade-off between number of network links and bandwidth.

In fact, the two restrictions of network-link number and bandwidth are independent each other, and may have conflicts sometimes. For example, there is no absolute standard to judge which of the following two mapping schemes is better: using three 10Mbps links to emulate a logic link, or using one 100Mbps link. Commonly, it needs the network administrator to make the decision according to various situations.

Considering this, the scaling values α and β are added. When α is comparably bigger and β is smaller, the evaluation function tends to choose the result that uses less links and vice versa. In most situations, network administrator is more willing to reserve more links for future experiments, so usually $\alpha > \beta$.

In all the mapping schemes, the one that has the smallest evaluating value *E* according to the expression (3-1) is the best solution.

3.2 About the delay node

The delay node is not considered by the modeling of the problem in section 3.1, which is used to emulate a low speed (such as Dial-Up networking) or long delay network link (such as Satellite communication) on a high-speed network. The delay node connects two network interfaces, and by using dedicated emulation software, it can forward packets between network interfaces, while adjusts transmission delay and packet lost to emulate the specified network link. Adding delay nodes dynamically during the mapping process will make the problem more complex, so many related works ignore it.

The problem can be solved by a pre-processing to the logic-topology before the mapping. In the pre-processing step, each link in logic-topology is checked, those low speed and long delay network links are substituted with "high speed link + delay node + high speed link" in sequence, and the pre-processed topology can be used in the following mapping process.

4. SOLVING THE PROBLEM WITH GENETIC ALGORITHM

Genetic Algorithm (GA) is a computing model which simulating genetic choice and natural elimination in the biological evolution process. It treats each possible solution of the problem as one individual in the population, and codes each individual in the form of a string. Then a fitness function is used to calculate a fitness for each individual, which is used to evaluate the individual. At the beginning of the algorithm, several possible individuals are selected randomly to form the original population; secondly, based on each individual's fitness, they are operated using the gene (selection, crossover, mutation) to generate the new population.

Based on the model in section 3, we implements a solver for the network testbed mapping problem using genetic algorithm. The implementation detail is as follows:

(1) **Coding**: a simple coding scheme is used in which each octet corresponds to a physical node, and the value of the octet represents which logic-node it is mapped. For example, to map 9 logic-nodes to 12 physical nodes, the coding result have 12 octets, the code 1 5 2 3 4 -1 6 -1 7 8 9 –1 means physical node 1 is used for logic-node 1, physical node 2 is used for logic-node 5, etc. Value –1 means the physical node is unused.

(2) **Fitness function**: same with the evaluation function in expression (3-1).

(3) **Selection mechanism**: selection means the process of copying individuals to the new population according to the fitness of them. The bigger the fitness is, the higher the possibility of the individual been copied and therefore has more offspring. Selection mechanisms being widely used include: roulette wheel selection, elitist model, expected value model, stochastic tournament, ranking, and mixing. In our implementation, roulette wheel selection is used, in which the possibility for an individual being selected is directly determined by its fitness in the population.

(4) **Crossover**: for sequence coding, crossover can't be simply done in one-point crossover or multi-point crossover. As Figure-2 shows, for two parent individual A and B, if position 2 and 6 are chosen as the cross point, the resulted children of regional crossing are shown in Figure-2(b), in which logic-node 1 and 3 are mapped twice in child A, while logic-node 2 and 7 are not mapped to any physical node. The same problem also exists in child B. therefore, it is not a reasonable mapping method.

Solution to the problem we used is: after position 2 and 6 are chosen as the cross point, we put B's crossing region in front of A and put A's crossing region in front of B, then we have A1 and B1, finally, overlapped genes are deleted from A1 and B1 and child A2 and B2 are generated. As shown in Figure-2(c), both children A2 and B2 are legal mapping results.

```
Parent A:  1 3| 5 7 2 4 |6 8 9        Child A :  1 3 |3 1 4 5 |6 8 9
Parent B:  2 6| 3 1 4 5 |8 9 7        Child B :  2 6 |5 7 2 4 |8 9 7
```

(a) Parent (b) Children from one-point crossover

```
A1:  3 1 4 5 |1 3 5 7 2 4 6 8 9       Child A2:  3 1 4 5 7 2 6 8 9
B1:  5 7 2 4 |2 6 3 1 4 5 8 9 7       Child B2:  5 7 2 4 6 3 1 8 9
```

(c) Revised crossover

Figure 2. Crossover

(5) **Mutation**: mutation randomly changes value of octets in some individuals in a small probability. Exchange mutation is used in our implementation, in which two positions of an individual are selected and their values are exchanged. For example, for individual A, if position 3 and 6 are randomly selected, new individual A1 will be generated by exchanging values of position 3 and 6, as shown in Figure-3.

```
A:  1 3 5^ 7 2 4^ 6 8 9               A:   1 3 5^ 7 2 4^ 6 8 9
A1: 1 3 4^ 7 2 5^ 6 8 9               A1:  1 3 5^ 4 2 7^ 6 8 9
```

Figure 3. Mutation *Figure 4.* Evolution inversion

(6) **Evolution inversion**: to improve the regional searching capability of genetic algorithm, evolution reversion is used in our implementation. As shown in Figure-4, for individual A, if position 3 and 6 are selected as the inversion points, octets from position 3 to 6 are exchanges and new individual A1 are generated.

5. EVALUATION AND COMPARISON

The algorithm is evaluated using 100 randomly generated logic-topologies. The node number in logic-topologies varies from 4 to 16. The physical-topology used in evaluation is composed of 24 nodes, each of which have three network interfaces installed, and all connected to backplane switches.

For all the logic-topologies, optimal solutions are exhaustively searched and compared with genetic algorithm solutions. Furthermore, the exhaustive searching algorithm is improved by eliminating unnecessary searching steps, this improved algorithm cost much less time than the old.

For all the solutions, evaluation value E is calculated using expression (3-1), where scaling value α and β are fixed to 0.8 and 0.2.

Parameters for our implementation of genetic algorithm include: population size=300, number of iterations in each generation=200, crossover probability=0.85, and mutation probability=0.005.

Figure-5 gives the error rate of genetic solutions, which is calculated by comparing value E of genetic and optimal solutions. As Figure-5 shows, with number of nodes increases, error rates increases slowly and stays at about 10%. This demonstrates that genetic algorithm can find near-optimal solutions.

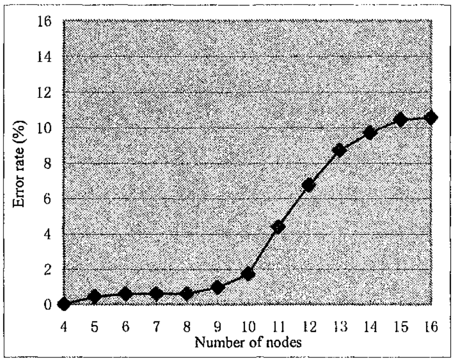

Figure 5. Error Rates of Genetic Solutions

Table 1. Average cost of algorithms

Number of nodes	Exhaustive searching	Improved exhaustive	Genetic algorithm
4	0 s	0 s	1.0 s
5	0 s	0 s	1.4 s
6	0 s	0 s	2.0 s
7	0 s	0 s	2.4 s
8	0.4 s	0 s	2.6 s
9	11.2 s	0 s	3.6 s
10	76.6 s	0.6 s	4.2 s
11	734 s	2.6 s	4.4 s
12	8817 s	7.6 s	4.6 s
13	114628 s	128 s	5.2 s
14	1604803 s	225 s	5.4 s
15	24072048 s	587 s	6.4 s
16	5885687808 s	2046 s	7.4 s

Tested on Windows XP with Intel P4 2.4GHz processor

Table 1 gives the computation cost of genetic, exhaustive-searching and improved exhaustive-searching algorithms. The data shows that scalabilities of genetic algorithms are preferable.

6. CONCLUSION

To provide remote service and auto-configuration mechanisms, network emulation testbed needs to map the logic-topology, which the researchers require, to the testbed physical-topology. In this paper, we explore this network testbed mapping problem. After building a model of the problem, this paper implements a solver for the problem using Genetic Algorithm. The evaluation result shows that the solver can find near-optimal solutions in reasonable time.

REFERENCES

[1] Brian White, et.al, An Integrated Experimental Environment for Distributed Systems and Networks, In Proceedings of the Fifth Symposium on Operating Systems Design and Implementation, Dec 2002.

[2] Robert Ricci, et.al, A Solver for the Network Testbed Mapping Problem, ACM SIGCOMM Computer Communication Review, Apr 2003.

[3] Xin Liu, Andrew A. Chien, Traffic-based Load Balance for Scalable Network Emulation, In Proceedings of Super Computing 2003 (SC'03), Nov 2003.

[4] Amin Vahdat, et.al, Scalability and Accuracy in a Large-Scale Network Emulator, In Proceedings of the Fifth symposium on Operating Systems Design and Implementation, Dec 2002.

[5] K. Fall, Network Emulation in the Vint/NS Simulator, In Proceedings of the Fourth Symposium on Computers and Communications, 1999.

[6] Larry Peterson, et.al, A Blueprint for Introducing Disruptive Technology into the Internet, First ACM Workshop on Hot Topics in Networks (HotNets-I), Oct 2002.

[7] D. Anderson. Theoretical Approaches To Node Assignment, Dec 2002. Unpublished manuscript. http://nms.lcs.mit.edu/papers/andersen-assign.ps

[8] Chen Guo-liang, et.al, Genetic Algorithm and Its Applictions, China Posts & Telecom Press, 1996.