Arosa: testbed resource allocation using late-binding and constraints

Qin Yin* Timothy Roscoe
Systems Group, Department of Computer Science, ETH Zurich

1 Introduction

Distributed network testbeds like ProtoGENI, VINI, etc., face an increasing challenge in resource allocation. Especially nowadays, these testbeds scale to even larger number of network resources which become more diverse, meanwhile, dependencies among these resources become more constrained. Faced with these trends, designers of testbed platforms need to solve two closely related problems: for the client, how to express the reservation request for computational and networking resources; in turn for the testbed provider, how to commit or offer resources in a way that makes efficient usage of the platform.

For the moment, a query-based model for resource allocation is usually adopted [1,5,6]. In this model, clients submit a declarative specification of resources they want, then if possible the provider issues a ticket for *concrete specific* resources meeting the requirements in the request, later the client redeems the ticket for a lease to use the resources. The request is a set of constraints on the resources to be allocated, and (optionally) some objective function to allow the provider to select for the client the most advantageous allocation from all the available options.

We take the idea a step further: not only do clients specify their requests as constraints, but providers reply with resource promises, which are *also* expressed as sets of declarative constraints on resources. Our simple first experiments [8] suggest that *late-binding* resources to requests, enabled by representing resource tickets as constraints, gives providers much greater flexibility in resource allocation. It also opens up a significant space of techniques for the optimization of resource usage by testbed providers.

We are now in the process of applying this idea into the implementation of a resource allocator called *Arosa*. Arosa allows users to reserve resources on nodes, switches and (virtual) links using a VF2-based [4] embedding algorithm. We are also exploring the optimization space introduced by late-binding by evaluating our system with an extensive test workload. The test workload is generated based on an analysis of a trace of experiments submitted to the popular Emulab [2] testbed.

*Qin Yin is a student author and will present the poster.

2 Basic idea: late-binding

We take a simple example in Figure 1 to illustrate the drawbacks of binding concrete specific resources to tickets early, in other words, early commitment.

Suppose the provide has a physical network of 2 switches and 8 hosts. REQUEST1 asks for 4 hosts, based on a simple strategy to minimize the total bandwidth of virtual links mapped across physical inter-switch links, the provider allocates 4 hosts under one switch: H1, . . ., H4, and binds them to TICKET1 as a reservation promise.

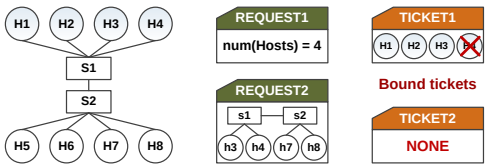

Figure 1: Drawbacks of early commitment

If any node in TICKET1 fails, inevitably the client will not be able to redeem TICKET1 for a lease; if another client submits REQUEST2, since H1, . . ., H4 has been bound to TICKET1, REQUEST2 can not be satisfied with the remaining resources.

Late-binding resources to requests addresses these two issues. As shown in Figure 2, instead of issuing a ticket for 4 concrete specific hosts, the provider issues a ticket for 4 hosts without specifying which four. Only when the ticket is redeemed, the requests are bound to specific

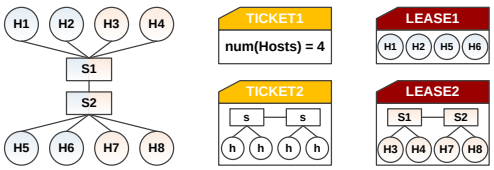

Figure 2: Late-binding resources to requests

concrete resources. Since resources can be reallocated before they are required, changes in physical resource availability can be masked from the clients by reallocation, furthermore, REQUEST2 can be satisfied with the allocation solution shown in Figure 2.

3 Approach

In Arosa, we use constraints to describe not only resource reservation requests, but also resource promises made by the provider. We also retain the distinction between tickets and leases: a lease grants access to specific, named resource components, whereas a ticket simply describes (in more or less detail) a set of resources which the client may gain a lease in the future.

In Arosa, the ticket does not bind a specific set of concrete resources. Instead, the ticket is simply another set of constraints. These might refer to specific switches or nodes, but are more likely to be "unbound": they simply describe elements in the virtual network are not yet mapped to a physical component. Leases, on the other hand, will always refer to specific resources.

The provider maintains an up-to-date list of all physical resources available, their current condition, together with a list of all valid tickets and leases that it has issued. The provider will typically also have some kind of objective function it will seek to maximize – most likely utilization in the case of a networking testbed, but in commercial settings this will probably be some function of yield or revenue.

If the request is for a lease, the provider will return a concrete resource assignment in the lease. If the request is for a ticket, the provider has considerable freedom in deciding whether to issue a ticket, and for what. One option is to try to produce, from scratch, a complete resource assignment satisfying all outstanding tickets and respecting all outstanding leases. A second option, which mirrors the strategy of existing systems, is to perform a purely incremental allocation for each new request.

Arosa adopts the middle ground: it maintains a concrete resource assignment internally for each ticket and returns a ticket not bound to any specific set of concrete resources. This allows Arosa to transparently do a limited reassignment of resources if a request for a ticket cannot be trivially satisfied (respecting all privately stored ticket assignments and outstanding leases), or some other event causes a change in the system state. Furthermore, late-binding also allows Arosa to run an off-line optimizer periodically (or by request) to remap the assignments for issued tickets so as to maximize the objective function specified by the provider, or to alleviate network bottlenecks to accommodate more future requests.

4 Current status and future work

Arosa now accepts resource requests with constraints on when the request should be satisfied, types and characteristics of computational and networking resources, high-level network connectivity properties, and concrete virtual network topology. The solving engine inside Arosa assigns switches and links from physical infrastructure so as to satisfy the requested network configurations. The virtual network mapping algorithm, core of the solving engine, is implemented based on a thorough survey of various virtual network mapping algorithms including simulated annealing [7], subgraph isomorphism detection [3], etc.

Meanwhile, we are analyzing a 5-year trace of experiments submitted to the Emulab testbed to generate an extensive test workload. We are using this test workload to evaluate the performance of Arosa and its solving engine, and to explore the potentials of late-binding such as: accommodating physical resource changes, achieving better testbed resource utilization, avoiding unnecessarily committing scarce resources, and supporting multi-stage resource negotiation.

References

[1] CHASE, J. ORCA control framework architecture and internals. Technical report, Duke University, September 2009.

[2] Emulab - Network Emulation Testbed. http://www.emulab.net/.

[3] LISCHKA, J., AND KARL, H. A virtual network mapping algorithm based on subgraph isomorphism detection. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures* (New York, NY, USA, 2009), VISA '09, ACM, pp. 81–88.

[4] P. CORDELLA, L., FOGGIA, P., SANSONE, C., AND VENTO, M. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell. 26* (October 2004), 1367–1372.

[5] PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services. http://www.planet-lab.org/.

[6] ProtoGENI. http://www.protogeni.net/trac/protogeni.

[7] RICCI, R., ALFELD, C., AND LEPREAU, J. A solver for the network testbed mapping problem. *SIGCOMM Comput. Commun. Rev. 33* (April 2003), 65–81.

[8] YIN, Q., AND TIMOTHY, R. A better way to negotiate for testbed resources. In *Proceedings of the second ACM asia-pacific workshop on Workshop on systems* (2011), APSys '11, pp. 93–97.