Hybrid approach for Experimental Networking Research

Amine Abidi[1], Sonia Mettali Gammar[1], Farouk Kamoun[1], Walid Dabbous[2], Thierry Turletti[2], and Arnaud Legout[2]

[1] National School of Computer Science, Manouba, Tunisia
amine.elabidi@cristal.rnu.tn,
sonia.gammar@ensi.rnu.tn,farouk.kamoun@ensi.rnu.tn
[2] INRIA Planète group, Sophia Antipolis, France
{Walid.Dabbous, Thierry.Turletti, Arnaud.Legout}@inria.fr

Abstract. Simulation is often used for the evaluation of new network protocols and architectures. In order to perform more realistic simulations, modern simulators such as ns-3 integrate more detailed models and even support direct execution of real protocol code. However, such complex models require more computational and memory requirements. In this paper, we study the feasibility of a hybrid approach based on distributing a complex simulation scenario on several nodes in a grid network. We show that by exploiting the real time operation of the ns-3 simulator, it is possible to map such complex scenarios on grid nodes. We run experiments to define the operational zone in which the obtained results are accurate. We also propose a basic mapping algorithm to distribute a simulation scenario in several nodes.

1 Introduction

In networking research, different methods are used to evaluate newly proposed protocol: analytical models, physical testbeds, simulation and emulation[5]. Analytical models are purely mathematical studies of the proposed solution. They give theoretical formulation and a clear idea about the problem complexity and design. However, these models do not evaluate implementation details and constraints that may be critical in many cases.

Physical testbeds are platforms formed by real equipments designated for tests and experimentations. In most cases, they consist of a small connected network with highly performing nodes. Large testbeds are costly and difficult to build. Overlay networks like PlanetLab[7] are an opportunity to have access to realistic large scale distributed experimentation. However, this approach is limited by the lack of experiments repeatability. It is difficult to control or even to know the network conditions during the experiment.

Network emulation is a method that provides a controlled and repeatable environment and still allows the use of actual protocols and applications. In network emulation, real hosts running actual protocols and applications are able to interact through an environment that simulates specified network conditions.

Possible uses of network emulation include stress testing of Web servers and testing of Internet games. Emulab[6] is one of the most used emulation system with 18000 experiments per year. Nevertheless, using and configuring such platform remind a complicated task specially with large scenarios.

Simulation provides a virtual representation of different network components. Users can easily manipulate any parameter to configure different scenarios. It is also possible to repeat the experiments as scenarios are fully controllable. Simulation is a widely used approach in networking research community. The complexity of this virtual representation depends on the granularity level. Simple models reduce the level of details and provide a high-level abstraction, but many functional details are omitted in this case. In order to validate new network protocols, complex and realistic models must be used. However, such required detailed models require an important amount of processing and memory resources. Consequently, large scenarios are difficult to evaluate. In this paper, we focus on how to distribute a complex simulation scenario over several nodes interconnected by a high speed emulated grid network, thus leveraging the availability of large amount of processing and memory resources on such networks. The objective is to lower the simulation load per machine and therefore be able to run both complex and large scale simulation scenarios easily. Then, it is important to study the operational zone in which we obtain the accurate results of the simulation, i.e., the zone where such that a distributed scenario provides the same results when run over single simulation machine or on the grid network. In other words, what are the applicability limits where the distributed scenario results are meaningful. Another problem to solve is how to split a single machine scenario on several parts to be executed over grid nodes, or try to find the more efficient the mapping algorithm.

Defining distribute simulation applicability limits and mapping algorithm is challenging. The first issue is how to keep *global synchronized scenario* with a huge number of events to be processed. Real time simulation have to keep event processing synchronized with the real clock. If the amount of events exceeds the processor capability, the global synchronization cannot be maintained. *Grid infrastructure impact* is another issue to solve. We have to check if the Grid topology or the bandwidth did affect the simulation scenario or not. After defining the operational zone or 'applicability limit', we have to define a mapping algorithm that respects this limit and splits the global scenario depending on its complexity. So it is necessary to calculate the processing load and to find the minimal number of grid nodes to build a safe distributed simulation.

In this work, we have chosen to use the new ns-3 network simulator mainly because it supports real time features and it is able to run in emulation mode where packets can be exchanged between simulated nodes and other nodes in the network. We selected the Grid5000 environment because it offers good network performance and full node management. Our main contribution consisted in 1) showing the feasibility of simulation scenarios distribution over this environment and 2) finding the applicability limits which represent the safe simulation zone,

and defining a mapping algorithm to split a global scenario on independent parts, and 3) developing a mapping algorithm respecting those limits.

The rest of this paper is organized as follows. Section 2 describes our distributed simulation environment including the ns-3 network simulator and the *Grid5000* grid network. Section 3 presents technical steps to run a distributed simulation. Section 4 defines the applicability limits and verifies the validity of the results given by our experimentation method by comparing them with one-node simulation. Then, it presents the mapping algorithm. Finally, section 5 concludes the paper.

2 Hybrid Experimental Environment

Our hybrid experimental environment includes both a network simulator and a grid environment. In this section, we discuss why we have chosen the ns-3 network simulator and the *Grid5000* grid network.

2.1 The ns-3 Network Simulator

We have chosen the discrete event ns-3 simulator because we needed a network simulator with real time simulation and emulation features. ns-3 includes a real time scheduler that makes easier "simulation-in-the-loop" use cases for interacting with real systems. For example, users can emit and receive ns-3-generated packets on real network devices, and ns-3 can serve as an interconnection framework to add the effects on real links between virtual machines. This feature is important to be able to connect simulators installed on different machines (e.g. grid nodes). Furthermore, the Direct Code Execution (DCE) feature of ns-3 allows to run simulations with unmodified applications or the entire Linux kernel networking stack. The latter option allows to run network scenarios with real applications.

ns-3 is the new version of the ns-2[2] network simulator, rewritten from scratch[1]. It is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. ns-3 is free software, licensed under the GNU GPLv2 license, and is publicly available for research, development, and use.

As shown in figure 1, each machine of the grid network will run a different scenario. Each scenario will include a node with an *emulated device* as link to the real network devices. This node will be able to send/receive data from the network.

2.2 The Grid-5000 Grid Environment

The selected Grid network must provide a closed environment and must provide full control on the used nodes. Grid5000[4] is an example of grid network that fulfils these needs. It is a infrastructure distributed in 9 interconnected sites in France. In each site, a gigabyte switched Ethernet network is built from at least 2

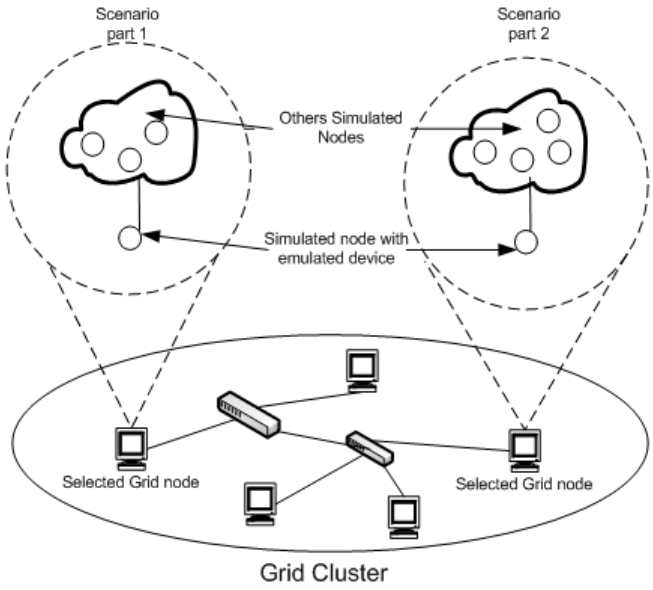

Fig. 1. Example of using ns-3 emulation mode over a Grid Network

clusters. Each user is provided an account with ssh access to get in the platform. To be able to launch experimentation, users have to reserve a number of nodes. Any operating system can be installed in the reserved nodes.

Platforms like *PlanetLab* also make it possible to run a real code on physical testbeds. Here the choice of a Grid network offers better flexibility and the possibility to fully control the experimental environment without any external traffic interaction. Also, we can easily manipulate node parameters and debug any feature in the simulated scenario.

3 Hybrid Experimentation Setup

In this section, we explain how to set up an hybrid experimentation with ns-3 over Grid5000. After node reservation and operating system configuration under grid5000, the following steps are necessary:

1. Configure the simulation scenario.
2. Synchronize scenario start-up in each grid node.

3.1 Configuration of the Simulation Scenario

To be able to use the *emulation mode* in any ns-3 script scenario, we have to configure an emulation device and assign it to at least one Grid node. This configuration is done as shown in the script listed below. It ensures that the

simulation scenario can control transmission of the chosen physical interface. Then, the emulated device has to be attached with at least one node in our scenario. As shown in figure 1, the chosen node must be a router to be able to forward traffic to other simulated nodes.

To allow communication with real device, the simulator must be run using the *real time scheduler*. Also, as mentioned in the following ns-3 code, we have to fix the *hard limit*, which stands for the maximum time difference allowed between the real time and the simulated time. On scheduling simulation events, the simulator has to be as fast as the real clock. If the difference between the two times exceeds the hard limit, the simulator will abort. As we will show later in the paper, the *hard limit* is used to compute the real time processing limit.

The ns-3 code for emulation mode configuration

```
   /* Emulation parameters */
  std::string deviceName ("eth0s");
  std::string encapMode ("Dix");
  CommandLine cmd;
  cmd.AddValue("deviceName", "device name", deviceName);
  cmd.AddValue("encapsulationMode", "encapsulation mode of emu
   device ("Dix" [default] or "Llc")", encapMode);
  cmd.Parse (argc, argv);
/* Using the real time scheduler */
  GlobalValue::Bind ("SimulatorImplementationType",StringValue
   ("ns3::RealtimeSimulatorImpl"));
/* define the hard limit */
  Config::SetDefault ("ns3::RealtimeSimulatorImpl::
  SynchronizationMode",StringValue("HardLimit"));
  Config::SetDefault ("ns3::RealtimeSimulatorImpl::HardLimit"
  ,TimeValue(Seconds(0.2)));

/* Create the emulated device */
  EmuHelper emu;
  emu.SetAttribute ("DeviceName", StringValue (deviceName));
  emu.SetAttribute ("EncapsulationMode",
   StringValue (encapMode));

  Ptr<EmuNetDevice> device = CreateObject<EmuNetDevice> ();
  std::string adr="00:14:4f:ca:97:78";
  device->SetAttribute("Address",Mac48AddressValue
  (Mac48Address::  Mac48Address(adr.c_str())));
  device->SetAttribute ("DeviceName", StringValue (deviceName));
  device->SetAttribute ("EncapsulationMode",
  StringValue (encapMode));
  Ptr<Queue> queue = CreateObject<DropTailQueue> ();
  device->SetQueue (queue);
  node->AddDevice (device);
```

3.2 Time Synchronization

As mentioned previously, in our experimentation each grid node runs separately one piece of a large scenario. Once the scenario is configured in each grid node, the simulation start-up time has to be set up. We use NTP to synchronize the clock of the different Grid5000 nodes and the Cron job scheduler utility to automate the simulation start-up time in the different nodes.

4 Operating Limits and Mapping Algorithm

In this section, we will study the validity of our approach and in particular, the operating limits. It is important to identify the conditions to satisfy in order to create the mapping algorithm and conduct a valid hybrid experimentation. Also, we will verify that our approach provides the same results as the experimentation scenario was run over one single node.

4.1 Study of the operating limit

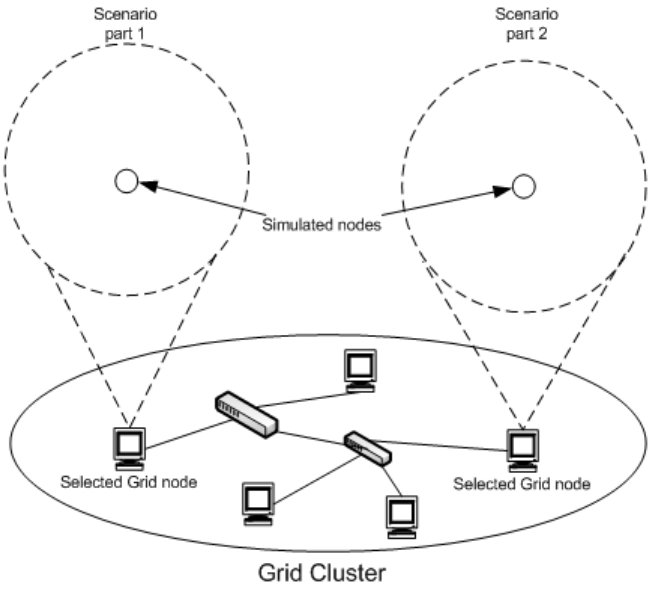

Fig. 2. Scenario : Study of the Grid5000 bandwidth impact

Impact of Grid5000 Bandwidth First limit to study is the impact of the Grid5000 infrastructure on the conducted simulation. To verify that the infrastructure bandwidth will not affect the simulation results, we set-up the simple experimental scenario shown in figure 2. Basically, two simulated nodes exchange UDP traffic through the Grid5000 network. We increase the sending data rate and then measure the receiving data rate. We expect to find the same data rate in both cases to ensure that the Grid5000 infrastructure is not a bottleneck.

Results are given in figure 3. As expected, we obtain the same traffic at the receiver in both cases. The available bandwidth provided by Grid5000 is 1 Gbps. So, the infrastructure will not affect the results if we ensure that the data sending rate does not exceed 1Gbps.

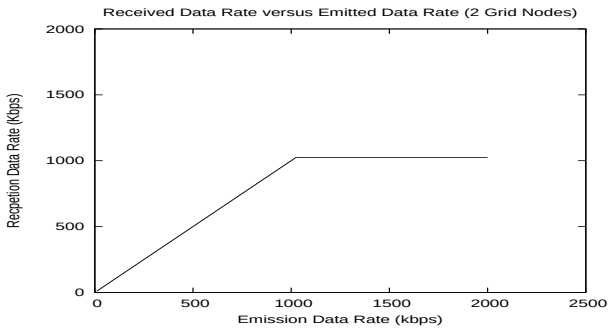

Fig. 3. Impact of the Grid5000 bandwidth on the receiving data rate

Real Time Processing Limit In order to detect the real time processing limit, we increase the number of processed events until reaching the hard limit. The hard limit is reached when the scheduler is no longer able to keep up with real time clock. As shown in figure 4, a distributed scenario is run over two grid nodes. Then, we increase the exchanged traffic by increasing the number of simulated nodes. The simulation is stopped after using 66 simulated nodes. Then, we estimate the number of processed events to compute the real time processing limit. We can estimate the number of the processed events by counting the number of processed bytes per second, as proposed in [9]. In our case this limit

is 13 Mbps. In other words, the amount of traffic processed every second by any grid node must be less than 13 Mbps to be able to run *real time* simulation.

4.2 Performance Study

In order to evaluate the efficiency of the results given by our hybrid approach, we compare the results obtained using the hybrid experimentation with results obtained by a simulation on one single node. Then we increase the simulation complexity by increasing the number of nodes in the scenario and the amount of the traffic exchanged. We measure the *average received data rate* to compare hybrid results with single node results. In our experimentation we use the following number of grid nodes: 2,4 and 8. In each scenario, a TCP Traffic is set up between each simulated nodes (sender and receiver located in two different grid nodes). We fix the throughput in the sending node at 200 kbps and the scenario time to 200 seconds. Then we increase the number of simulated nodes. The same scenarios are executed both in the hybrid environment and in one single node.

As shown in figures 4, 5 and 6, we measure the receiving data rate versus the number of couples (sender, receiver) referenced as the Flow Number. Results are given for the hybrid simulation and the one single node simulation. Clearly, the two kinds of experimentation provide exactly the same results. Despite the variation of the number of simulated nodes, the receiving data rate measured is the same. Also, we can check that the number of grid nodes used in the simulation do not affect the accuracy of the experimentation results.

Moreover, as we see in figure 6, we are able to simulate 166 couples of (sender, receiver) by using 8 grid nodes. Therefore, we show that by increasing the number of grid nodes, it is possible to increase the scalability of the hybrid experimentation platform.

Figures 7, 8 illustrate the execution time of the 2 scenarios (4 and 8 Grid nodes). We can see that for the hybrid experimentation we run a*real time* simulation. Then, the execution time will be always the same and equal to the time fixed for the scenario (200 seconds). But in the single node simulation the scheduler is not *a real time* scheduler but *a discrete event* one. We can see that as the scenario becomes more complex, more time is needed to complete the simulation. However, in the hybrid experimentation, we will never exceed the time fixed in the scenario. So, the hybrid experimentation will be more adequate with complex scenarios.

4.3 Mapping Algorithm

In order to use the hybrid evaluation approach in network experimentation, it is necessary to define a mapping algorithm to automate the conversion from a scenario designed to be simulated over one single node to a full distributed scenario. The mapping algorithm has to respect the operational limits of the system, and more precisely the real time processing limit. The objective is then to respect the hard limit of the simulator in order to keep a real time simulation. The mapping algorithm needs to estimate the overall load of the scenario to

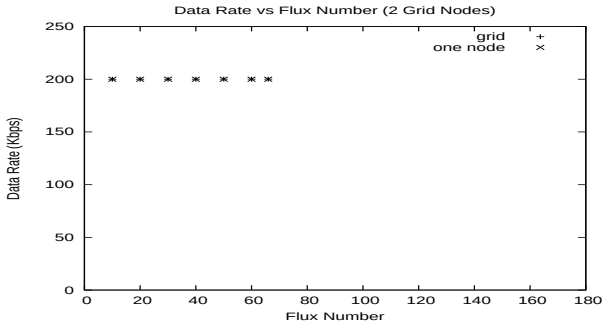

Fig. 4. Received Data Rate versus node number with 2 Grid nodes

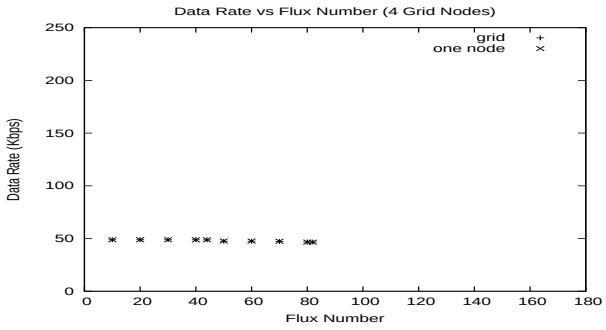

Fig. 5. Received Data Rate versus node number with 4 Grid nodes

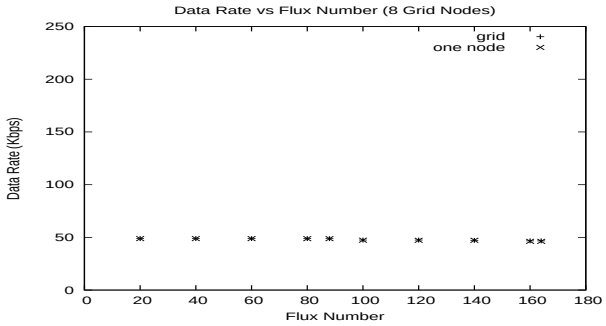

Fig. 6. Received Data Rate versus node number with 8 Grid nodes

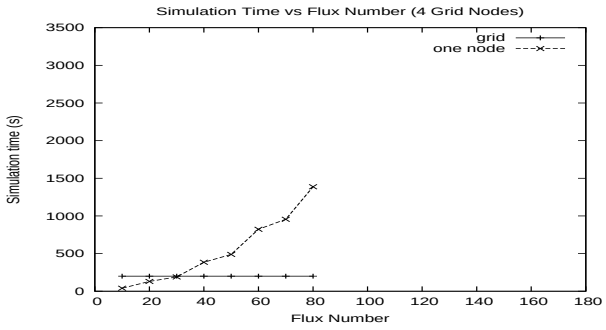

Fig. 7. Simulation execution time with 4 Grid nodes

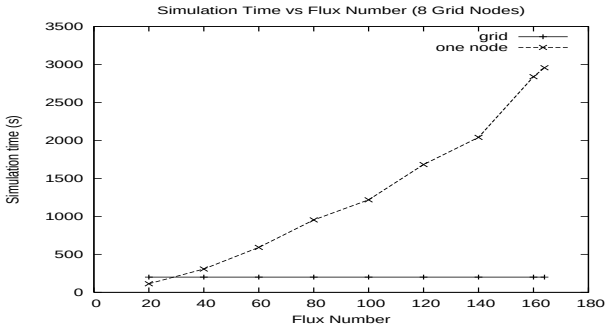

Fig. 8. Simulation execution time with 8 Grid nodes

find the adequate number of grid nodes needed for the distributed scenario. The mapping algorithm follows the following steps:

1. Calculate the number of the processed events in the global scenario. It will be estimated by the number of processed bytes (i.e. the limit must be under 13 Mbps for Grid5000). Then, compute the minimal number of grid nodes required.
2. Study the topology of the scenario and try to group linked simulated nodes in the same grid node respecting the real time limit.
3. Configure links between grid nodes based on the scenario topology.

In the following, we provide the pseudo code of the mapping algorithm.

Pseudo code of the Mapping Algorithm

```
var
 CumuLoad (an integer storing the cumulated
  load of the scenario part)
 NodeList (a list of all nodes in the global scenario)
begin
  CumuLoad:=Initiate();
  for node in NodeList do
   CumuLoad:=CumuLoad+CalculateLoad(node);
   if (CumuLoad>LoadLimit) then
   NewPart:=GenerateNewSimPart();
```

```
    CumuLoad:=Initiate();
    Store(NewPart);
    else
    node=NextNode;
   end
  end.
```

5 Conclusion

In this paper we presented an hybrid approach for the evaluation of networking protocols based on the ns-3 network simulator and a Grid testbed. We studied the feasibility of the approach and illustrate its performance with a simple use case. The evaluation shows that our proposition can be used with larger and more complex scenarios. The scalability of the simulation is limited by processor speed and memory capacities of the simulation node. By dividing the scenario over a number of grid machines, we show that the scalability of the evaluation platform can be increased.

References

1. ns-3: The network simulator ns-3, "http://www.nsnam.org/".
2. NS2: The network simulator ns-2, "http://www.isi.edu/nsnam/ns/".
3. Grid5000: Grid5000.fr, "https://www.grid5000.fr/mediawiki/index.php/Grid5000:Home".
4. J.Liu: Immersive real-time large-scale network simulation: a research summary. Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS'08) NSF NGS Workshop, Miami, Florida, April 13-14, 2008.
5. B. White et al: An integrated experimental environment for distributed systems and networks. OSDI, 2002.
6. L. Peterson et al: A blueprint for introducing disruptive technology into the Internet. HotNets-I, 2002.
7. S.Jansen: Simulation with Real World Network Stacks. Winter Simulation Conference 2005.
8. M. Lacage: Outils d'expérimetation pour la recherche dans les reseaux. "http://www-sop.inria.fr/members/Mathieu.Lacage/thesis.pdf". 2010