Platform 101
Getting Started with SimGrid Platforms

Da SimGrid Team

April 17, 2016
Outline

- The Network Representation Issue

- XML Based Formalism for Platform Description
  - The XML Approach
  - Specifying Host
  - Specifying inter-host network connections
  - Compacting the XML platform Description
  - Autonomous systems
  - Setting Properties
  - Examples of use

- Lua Based Formalism For Platform Description
  - Specifying Platform Element
  - Examples of use
  - Deploy Application
**Network Communication Models**

**Packet-level simulation** full simulation of the whole protocol stack
- Main approach in networking community (NS, GTneTS, OmNet++, ...)  
- Complex models \(\sim\) hard to instantiate and inherently slow  
- Beware of simplistic packet-level simulation (GridSim: slow but not accurate)

Along the same lines: Weaver and MsKee, *Are Cycle Accurate Simulations a Waste of Time?*  
Proc. of the Workshop on Duplicating, Deconstruction and Debunking, 2008

**Delay-based models** The simplest ones, often favored in P2P community
- **Constant time:** communication time is constant or statistically distributed  
  \(\sim\) \(\Theta(1)\) footprint and \(O(1)\) computation  
- **Coordinates:** geographic proximity  
  \(\sim\) \(\Theta(N)\) footprint and \(O(1)\) computation  
- **LogP:** more detailed model  
  \(\sim\) \(\Theta(N)\) footprint and \(O(1)\) computation

Scalable models, but ignore network congestion
Flow-level models  A communication (flow) is simulated as a single entity:

\[ T_{i,j}(S) = L_{i,j} + \frac{S}{B_{i,j}}, \]

where

\[ \begin{cases} 
    S & \text{message size} \\
    L_{i,j} & \text{latency between } i \text{ and } j \\
    B_{i,j} & \text{bandwidth between } i \text{ and } j 
\end{cases} \]

Estimating \( B_{i,j} \) requires to account for interactions with other flows.
Flow-level models  A communication (flow) is simulated as a single entity:

\[ T_{i,j}(S) = L_{i,j} + S/B_{i,j}, \]

where

\[ \begin{cases} 
S & \text{message size} \\
L_{i,j} & \text{latency between } i \text{ and } j \\
B_{i,j} & \text{bandwidth between } i \text{ and } j 
\end{cases} \]

Estimating \( B_{i,j} \) requires to account for interactions with other flows.

Assume steady-state and share bandwidth every time a new flow appears or disappears.

**Setting**  a set of flows \( \mathcal{F} \) and a set of links \( \mathcal{L} \)

**Constraints**  For all link \( j \):

\[ \sum_{i} \varrho_i \leq C_j \]

if flow \( i \) uses link \( j \)
Network Communication Models (cont’d)

**Flow-level models** A communication (flow) is simulated as a single entity:

\[ T_{i,j}(S) = L_{i,j} + \frac{S}{B_{i,j}}, \]

where

- \( S \): message size
- \( L_{i,j} \): latency between \( i \) and \( j \)
- \( B_{i,j} \): bandwidth between \( i \) and \( j \)

Estimating \( B_{i,j} \) requires to account for interactions with other flows.

Assume steady-state and **share bandwidth** every time a new flow appears or disappears.

**Setting** a set of flows \( F \) and a set of links \( L \)

**Constraints** For all link \( j \):

\[ \sum_{i \text{ flow uses link } j} \varrho_i \leq C_j \]

if flow \( i \) uses link \( j \)

**Objective function**

- Max-Min \( \max(\min(\varrho_i)) \)
- or other fancy objectives
  - e.g., Reno \( \sim \max(\sum \log(\varrho_i)) \)
Wrap up on flow-level models

Such **fluid models can account** for TCP key characteristics

- slow-start
- flow-control limitation
- RTT-unfairness
- cross traffic interference

They are a very reasonable approximation for most LSDC systems

Yet, many people think they are too complex to scale.

Let’s prove them wrong! 😊
How to achieve scalability

Platform description

Main issues with topology
- description size, expressiveness
- memory footprint
- computation time

$N$ nodes and $E$ links

<table>
<thead>
<tr>
<th>Representation</th>
<th>Input</th>
<th>Footprint</th>
<th>Parsing</th>
<th>Lookup</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How to achieve scalability

Platform description

Main issues with topology
▶ description size, expressiveness
▶ memory footprint
▶ computation time

Classical network representation

1. Flat representation
   5000 hosts doesn’t fit in 4Gb!

\[ \begin{array}{|c|c|c|c|}
\hline
\text{Representation} & \text{Input} & \text{Footprint} & \text{Parsing} & \text{Lookup} \\
\hline
\text{Flat} & N^2 & N^2 & N^2 & 1 \\
\hline
\end{array} \]
How to achieve scalability

Platform description

Main issues with topology
- description size, expressiveness
- memory footprint
- computation time

Classical network representation
1. Flat representation
   5000 hosts doesn’t fit in 4Gb!
2. Graph representation assuming shortest path routing

<table>
<thead>
<tr>
<th>Representation</th>
<th>Input</th>
<th>Footprint</th>
<th>Parsing</th>
<th>Lookup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dijsktra</td>
<td>$N + E$</td>
<td>$E + N \log N$</td>
<td>$N + E$</td>
<td>$E + N \log N$</td>
</tr>
<tr>
<td>Floyd</td>
<td>$N + E$</td>
<td>$N^2$</td>
<td>$N^3$</td>
<td>1</td>
</tr>
</tbody>
</table>
How to achieve scalability

Platform description

Main issues with topology

- description size, expressiveness
- memory footprint
- computation time

Classical network representation

1. Flat representation
   - 5000 hosts doesn’t fit in 4Gb!
2. Graph representation assuming shortest path routing
3. Special class of structures (star, cloud, ...)

<table>
<thead>
<tr>
<th>Representation</th>
<th>Input</th>
<th>Footprint</th>
<th>Parsing</th>
<th>Lookup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star</td>
<td>1</td>
<td>$N$</td>
<td>$N$</td>
<td>1</td>
</tr>
<tr>
<td>Cloud</td>
<td>$N$</td>
<td>$N$</td>
<td>$N$</td>
<td>1</td>
</tr>
</tbody>
</table>

$N$ nodes and $E$ links
Our proposal

Every such representation has drawbacks and advantages
Let’s build on the fact that most networks are mostly hierarchical

1. Hierarchical organization in AS
   $\leadsto$ cuts down complexity
   $\leadsto$ recursive routing
2. Efficient representation of classical structures
3. Allow bypass at any level
Step by step routing

1. \( \text{SRC} \rightarrow AS_{t_0} \rightarrow \cdots \rightarrow AS_{t_n} \rightarrow AS_{\text{common}} \)
2. \( \text{DST} \rightarrow AS_{r_0} \rightarrow \cdots \rightarrow AS_{r_m} \rightarrow AS_{\text{common}} \)
3. \( \text{get\_route}(AS_{t_n}, AS_{r_m}) \) from = \( GW_{\text{SRC}} \) to = \( GW_{\text{DST}} \) links = \( L_0 \cdots L_k \)
4. \( \text{get\_route}(\text{SRC}, GW_{\text{SRC}}) \) from = \( \text{SRC} \) to = \( GW_{\text{SRC}} \) links = \( L_{s_0} \cdots L_{s_x} \)
5. \( \text{get\_route}(GW_{\text{DST}}, \text{DST}) \) from = \( GW_{\text{DST}} \) to = \( \text{DST} \) links = \( L_{d_0} \cdots L_{d_z} \)
Outline

- The Network Representation Issue

- XML Based Formalism for Platform Description
  - The XML Approach
  - Specifying Host
  - Specifying inter-host network connections
  - Compacting the XML platform Description
  - Autonomous systems
  - Setting Properties
  - Examples of use

- Lua Based Formalism For Platform Description
  - Specifying Platform Element
  - Examples of use
  - Deploy Application
Outline

- The Network Representation Issue

- XML Based Formalism for Platform Description
  - The XML Approach
  - Specifying Host
  - Specifying inter-host network connections
  - Compacting the XML platform Description
  - Autonomous systems
  - Setting Properties
  - Examples of use

- Lua Based Formalism For Platform Description
  - Specifying Platform Element
  - Examples of use
  - Deploy Application
XML Platforms

platform.xml

```xml
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM "surfxml.dtd">
<platform version="2">
<AS id="AS0" routing="Full">
  <host name="host1" speed="1E8"/>
  <host name="host2" speed="1E8"/>
  <link name="link1" bandwidth="1E6" latency="1E-2"/>
  <route src="host1" dst="host2">
    <link:ctn id="link1"/>
  </route>
</AS>
...
</platform>
```

- Introduced since version 3 (released in 2005)
- Separate the Application Scenario
- FleXML based Mechanism
- SAX Approach (Callbacks)
Outline

- The Network Representation Issue

- XML Based Formalism for Platform Description
  - The XML Approach
  - Specifying Host
  - Specifying inter-host network connections
  - Compacting the XML platform Description
  - Autonomous systems
  - Setting Properties
  - Examples of use

- Lua Based Formalism For Platform Description
  - Specifying Platform Element
  - Examples of use
  - Deploy Application
Specifying Host

<Host> Tag

<host id="host_id"
    speed="500000000"
    [availability_file="hostAvail.trace"]
    [state_file="hostState.trace"] />

- **id**: Host Identifier
- **speed**: Host Computational Power in Flops
- **availability_file**: Trace file giving the availability variation
- **state_file**: Trace file modeling the failures
Expressing dynamicity

Adding a availability trace file

```xml
:host id="Bob"
  speed="500Mf"
  availability_file="bobAvail.trace" />
```

Example of "bobAvail.trace" file

PERIODICITY 3.0
0.0 1.0
11.0 0.5
20.0 0.8

time=0 : The host delivers 500 Mflop/s
time=11 : Delivers half that is 250 Mflop/s until time 20.0
time=20 : Starts delivering 80% of its power, that is 400 Mflops/s
time=23 (20+Periodicity): Loops the beginning and delivers again 500 Mflops/s
time=34 (23+11): Delivers 250 Mflop/s, and so on.

Adding a state trace file

```xml
:host id="Bob"
  speed="500Mf"
  state_file="bobState.trace" />
```

Example of "bobState.trace" file

PERIODICITY 5
0 1
5 0

time=0 : The host is up and running
time=5 : The host is shut down
time=10 (5+Periodicity): The host boots up again, and so on.
Outline

- The Network Representation Issue

- XML Based Formalism for Platform Description
  - The XML Approach
  - Specifying Host
  - Specifying inter-host network connections
  - Compacting the XML platform Description
  - Autonomous systems
  - Setting Properties
  - Examples of use

- Lua Based Formalism For Platform Description
  - Specifying Platform Element
  - Examples of use
  - Deploy Application
Declaring Network Links

### <link> Tag

```xml
<link id="link_id"
  bandwidth="125000000"
  latency="5E-5" [sharing_policy="SHARED"] />
```

- **id**: Link Identifier
- **bandwidth**: Link bandwidth in bytes/s
- **latency**: Link latency in seconds
- **sharing_policy**:
  - **SHARED** (by default) ⇒ if more than one flow go through a link, each get an equal share of the available bandwidth
  - **FATPIPE** ⇒ each flow going through this link will get all available bandwidth, whatever the number of flows (this allows to describe switches or Intern backbones)
Expressing dynamicity

Adding an availability trace file

Example of "link1.bw" file
PERIODICITY 12.0
4.0 40000000
8.0 60000000

Example of "link1.lat" file
PERIODICITY 5.0
1.0 0.001
2.0 0.01
3.0 0.001

- It is possible to declare links whose state, bandwidth or latency change over time
- In this case, the bandwidth and latency are respectively replaced by the bandwidth_file and latency_file attributes in the corresponding text files
Declaring routes

<route> Tag

<host id="Bob" power="100000000"/>
<host id="Alice" power="500000000"/>
<link id="Link1" bandwidth="125000000" latency="5E-5"/>

<route src="Bob" dest="Alice"> <link ctn id="Link1"/> </route>
<route src="Alice" dest="Bob"> <link ctn id="Link1"/> </route>
Expressing multi-hop routes

Multi-hop routes and asymmetry

```xml
<host id="BOB" power="100000000"/>
<host id="ALICE" power="50000000"/>

<link id="LINK_BOB" bandwidth="125000000" latency="5E-5"/>
<link id="LINK_ALICE" bandwidth="125000000" latency="5E-5"/>
<link id="SWITCH" bandwidth="125000000" latency="5E-5"
sharing_policy="FATPIPE"/>

<route src="BOB" dest="ALICE">
    <link_ctn id="LINK_BOB"/>
    <link_ctn id="SWITCH"/>
    <link_ctn id="LINK_ALICE"/>
</route>

<route src="ALICE" dest="BOB">
    <link_ctn id="LINK_ALICE"/>
    <link_ctn id="SWITCH"/>
    <link_ctn id="LINK_BOB"/>
</route>
```
Specifying routers

A router is like a host except it is invisible from the user level.

**<router> Tag**

```
<router id="R1">
<router id="R2">
```

**Using it**

```
<route src="A" dest="R1">
  <link_ctn id="Link1"/>
</route>

<route src="R1" dest="B">
  <link_ctn id="Link2"/>
</route>

<route src="R1" dest="C">
  <link_ctn id="Link3"/>
</route>
```

⇒/examples/msg/small_platform_with_routers.xml
Outline

- The Network Representation Issue

- **XML Based Formalism for Platform Description**
  - The XML Approach
  - Specifying Host
  - Specifying inter-host network connections
  - Compacting the XML platform Description
  - Autonomous systems
  - Setting Properties
  - Examples of use

- **Lua Based Formalism For Platform Description**
  - Specifying Platform Element
  - Examples of use
  - Deploy Application
A cluster is actually expended as an AS with a special type of routing...
Outline

- The Network Representation Issue

- XML Based Formalism for Platform Description
  - The XML Approach
  - Specifying Host
  - Specifying inter-host network connections
  - Compacting the XML platform Description
  - Autonomous systems
  - Setting Properties
  - Examples of use

- Lua Based Formalism For Platform Description
  - Specifying Platform Element
  - Examples of use
  - Deploy Application
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM "http://simgrid.gforge.inria.fr/simgrid.dtd">
<platform version="3">
  <AS id="AS0" routing="Full">
    <host id="Tremblay" power="98095000"/>
    <host id="Jupiter" power="76296000"/>
    ...
    <link id="6" bandwidth="41279125" latency="5.9904e-05"/>
    <link id="11" bandwidth="252750" latency="0.00570455"/>
    <link id="3" bandwidth="34285625" latency="0.000514433"/>
    <link id="7" bandwidth="11618875" latency="0.00018998"/>
    <link id="9" bandwidth="7209750" latency="0.001461517"/>
    ...
    <route src="Tremblay" dst="Fafard">
      <link_ctn id="4"/>
      <link_ctn id="3"/>
      <link_ctn id="2"/>
      <link_ctn id="0"/>
      <link_ctn id="1"/>
      ...
    </route>
    <route src="Tremblay" dst="Ginette">
      <link_ctn id="4"/>
      <link_ctn id="3"/>
      <link_ctn id="5"/>
    </route>
    <route src="Tremblay" dst="Bourassa">
      <link_ctn id="4"/>
      <link_ctn id="3"/>
      <link_ctn id="2"/>
      <link_ctn id="0"/>
      <link_ctn id="1"/>
      ...
    </route>
  </AS>
</platform>
Hierarchy of AS

Cluster with cabinets: platforms/griffon.xml

```xml
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM "http://simgrid.gforge.inria.fr/simgrid.dtd">
<platform version="3">
  <AS id="AS-griffon" routing="Full">
    <cluster id="griffon_cluster_cabinet1" prefix="griffon-" suffix=".nancy.grid5000.fr"
      radical="1-29,58,59,60" power="286087" bw="1.25e8" lat="2.4e-5"
      bb_bw="1.25e9" bb_lat="0" sharing_policy="FULLDUPLEX" bb_sharing_policy="SHARED"/>
    <cluster id="griffon_cluster_cabinet2" prefix="griffon-" suffix=".nancy.grid5000.fr"
      radical="30-57" power="286087" bw="1.25e8" lat="2.4e-5"
      bb_bw="1.25e9" bb_lat="0" sharing_policy="FULLDUPLEX" bb_sharing_policy="SHARED"/>
    <cluster id="griffon_cluster_cabinet3" prefix="griffon-" suffix=".nancy.grid5000.fr"
      radical="61-92" power="286087" bw="1.25e8" lat="2.4e-5"
      bb_bw="1.25e9" bb_lat="0" sharing_policy="FULLDUPLEX" bb_sharing_policy="SHARED"/>
    <link id="backbone" bandwidth="1.25e9" latency="2.4e-5" sharing_policy="SHARED"/>
    <ASroute src="griffon_cluster_cabinet1" dst="griffon_cluster_cabinet2"
      gw_src="griffon-griffon_cluster_cabinet1_router.nancy.grid5000.fr"
      gw_dst="griffon-griffon_cluster_cabinet2_router.nancy.grid5000.fr"
      symmetrical="YES">
      <link ctn id="backbone"/>
    </ASroute>
    <ASroute src="griffon_cluster_cabinet2" dst="griffon_cluster_cabinet3"/>
    ...
    </ASroute>
    <ASroute src="griffon_cluster_cabinet2" dst="griffon_cluster_cabinet3"/>
    ...
    </ASroute>
  </AS>
</platform>
```
Hierarchy of AS 2

A “Cloud” platform

<config id="General">
<prop id="network/coordinates" value="yes"></prop>
</config>

<AS id="AS0" routing="Vivaldi">
  <AS id="AS1_dc1" routing="RuleBased">
    <cluster id="AS1_cb1" prefix="cb1-" suffix=".dc1.acloud.com" radical="1-40" power="5.2297E9">
    <cluster id="AS1_cb2" prefix="cb2-" suffix=".dc1.acloud.com" radical="1-50" power="8.8925E9">
    <cluster id="AS1_cb3" prefix="cb3-" suffix=".dc1.acloud.com" radical="1-30" power="13.357E9">
    <AS id="gw_AS1_dc1" routing="Floyd">
      <ASroute src="AS1_cb(.*)" dst="AS1_cb(.*)" gw_src="cb$1src-AS1_cb$1src_router.dc1.acloud.com" gw_dst="cb$1dst-AS1_cb$1dst_router.dc1.acloud.com" symmetrical="YES">
        <link_ctn id="link_dc1_cb$1src"/>
        <link_ctn id="link_dc1_cb$1dst"/>
      </ASroute>
    </AS>
  </cluster>
  </cluster>
</AS>

<AS id="AS2_dc2" routing="RuleBased">
...
</AS>

<!-- internal routes between clusters -->
<ASroute src="AS3_cb(.*)" dst="AS3_cb(.*)" gw_src="cb$1src-AS3_cb$1src_router.dc3.acloud.com" gw_dst="cb$1dst-AS3_cb$1dst_router.dc3.acloud.com">
  <link_ctn id="link_dc3_cb$1src"/>
  <link_ctn id="link_dc3_cb$1dst"/>
</ASroute>
Routing types

- Full
- Floyd
- Dijkstra
- Dijkstra / cache
- Rulebased
- Cluster
- Vivaldi

We’ll keep on adding new constructs.

Peer

```xml
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM "http://simgrid.gforge.inria.fr/simgrid.dtd">
<platform version="3">

<config id="General">
  <prop id="network/coordinates" value="yes"></prop>
</config>

<AS id="AS0" routing="Vivaldi">
  <peer id="peer-0" coordinates="173.0 96.8 0.1" power="730000000.0"
       bw_in="13380000" bw_out="1024000" lat="5E-4"/>

  ....
</AS>
</platform>
```
Outline

- The Network Representation Issue

- **XML Based Formalism for Platform Description**
  - The XML Approach
  - Specifying Host
  - Specifying inter-host network connections
  - Compacting the XML platform Description
  - Autonomous systems
  - Setting Properties
  - Examples of use

- **Lua Based Formalism For Platform Description**
  - Specifying Platform Element
  - Examples of use
  - Deploy Application
Attaching properties to elements

Adding properties to Host

```
<host id="BOB" power="500000000">
  <prop id="memory" value="100000000"/>
  <prop id="desk" value="80E9"/>
  <prop id="OS" value="Linux 2.6.22-14"/>
</host>
```

Adding properties to Link

```
<link id="l1" bandwidth="125000000" latency="0.000100">
  <prop id="type" value="Ethernet"/>
</link>
```

⇒/examples/platforms/prop.xml
Retrieving values

SimDag interface

```c
xbt_dict_t SD_link_get_properties(SD_link_t link);
const char* SD_link_get_property_value(SD_link_t link, const char* name);
```

```c
xbt_dict_t SD_get_workstation_properties(SD_workstation_t workstation);
const char* SD_workstation_get_property_value(SD_workstation_t workstation, const char* name);
```

MSG interface

```c
xbt_dict_t MSG_host_get_properties(m_host_t host);
const char* MSG_host_get_property_value(m_host_t host, const char* name);
xbt_dict_t MSG_process_get_properties(m_process_t process);
const char* MSG_process_get_property_value(m_process_t process, const char* name);
```
Outline

- The Network Representation Issue

- XML Based Formalism for Platform Description
  - The XML Approach
  - Specifying Host
  - Specifying inter-host network connections
  - Compacting the XML platform Description
  - Autonomous systems
  - Setting Properties
  - Examples of use

- Lua Based Formalism For Platform Description
  - Specifying Platform Element
  - Examples of use
  - Deploy Application
Example of use

Where to find XML platform examples?
⇒ <simgrid_dir>/examples/platforms
⇒ <simgrid_dir>/examples/msg

Where to find XML platform generators for SimGrid?
⇒ <simgrid_dir>/contrib/trunk/platform_generation
⇒ <simgrid_dir>/contrib/trunk/VisualGrid
Outline

- The Network Representation Issue

- XML Based Formalism for Platform Description
  - The XML Approach
  - Specifying Host
  - Specifying inter-host network connections
  - Compacting the XML platform Description
  - Autonomous systems
  - Setting Properties
  - Examples of use

- Lua Based Formalism For Platform Description
  - Specifying Platform Element
  - Examples of use
  - Deploy Application
Outline

- The Network Representation Issue

- XML Based Formalism for Platform Description
  - The XML Approach
  - Specifying Host
  - Specifying inter-host network connections
  - Compacting the XML platform Description
  - Autonomous systems
  - Setting Properties
  - Examples of use

- Lua Based Formalism For Platform Description
  - Specifying Platform Element
  - Examples of use
  - Deploy Application
Lua Based Formalism For Platform Description

**lua Platforms**

```lua
platform.lua

require "simgrid"
simgrid.AS.new{id="AS0", mode="Full"};

simgrid.Host.new{id="Tremblay", power=98095000};
simgrid.Host.new{id="Jupiter", power=76296000};
...
for i=10,0,-1 do
    simgrid.Link.new{id=i, bandwidth=252750 + i*768, latency=0.000270544+i*0.087};

simgrid.Route.new("Tremblay","Jupiter","{1}");

simgrid.Route.new("Tremblay","Fafard", "0","1","2","3","4","8");
...
simgrid.msg_register_platform();
```

- lua console application ➔ scripting language
- Using loops and conditional instructions
- Simple and lightweight edition
- Good performances when interconnecting with C Code.
Specifying element in lua script

Specifying host

```lua
simgrid.Host.new{id="Tremblay", power=98095000};
```

Specifying link

```lua
simgrid.Link.new{id="3", bandwidth=98095000, latency=5E-5};
```

Specifying route

```lua
simgrid.Route.new{"Tremblay","Ginette","3","4","5"};
```

Register platform

```lua
[MSG] simgrid.msg_register_platform();
[SimDAG] simgrid.sg_register_platform();
```
Outline

- The Network Representation Issue

- XML Based Formalism for Platform Description
  - The XML Approach
  - Specifying Host
  - Specifying inter-host network connections
  - Compacting the XML platform Description
  - Autonomous systems
  - Setting Properties
  - Examples of use

- Lua Based Formalism For Platform Description
  - Specifying Platform Element
  - Examples of use
  - Deploy Application
Example of use

- Where to find lua console examples?
  `<simgrid_dir>/examples/msg/masterslave`
  `<simgrid_dir>/examples/simdag`

- Where to find lua script examples?
  `<simgrid_dir>/examples/lua/masterslave_bypass.lua`
  `<simgrid_dir>/examples/msg/masterslave/platform_script.lua`
  `<simgrid_dir>/examples/simdag/platform_script.lua`
Outline

- The Network Representation Issue

- XML Based Formalism for Platform Description
  - The XML Approach
  - Specifying Host
  - Specifying inter-host network connections
  - Compacting the XML platform Description
  - Autonomous systems
  - Setting Properties
  - Examples of use

- Lua Based Formalism For Platform Description
  - Specifying Platform Element
  - Examples of use
  - Deploy Application
## Deploy application in lua script

### Set function to process

```lua
simgrid.Host.setFunction("Tremblay","master",{"20","550000","1000","4"});
simgrid.Host.setFunction("Jupiter","slave",{"1"});
```

### Register application

```lua
[MSG] simgrid.msg_register_platform();
[SimDAG] simgrid.sg_register_platform();
```