Expressing meaningful processing requirements among HeTeRoGEneOus nodes in an active network

Virginie Galtier, Kevin Mills, Yannick Carlinet,
Stefan Leigh, Andrew Rukhin

National Institute of Standards and Technology

http://w3.antd.nist.gov/active-nets

Outline of the presentation

- Problem:
  - Context: What are active nets? What are they for?
  - Why is it interesting to know the CPU resources requirement of an active application (AA)?
  - What are the sources of variability in the execution time of an AA?
- Proposed solution:
  - Two models to characterize the processing requirements of an application on any active node
  - A mechanism to scale the models from one node to a different one
- Discussion and future work

Active networks overview

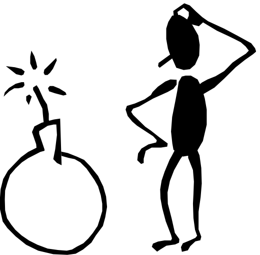
- Active packets carry not only data but also the code to process them which is executed at active nodes.

- Example: an application that sends MPEG packets can specify an intelligent dropping algorithm to be applied at intermediate nodes if congestion is detected.

- Advantage: fast and easy deployment of customized network services.

Why is it important to know the CPU resource requirements of an active application?

- Implication: in an active net the processing requirements can vary a lot from packet to packet .
- Without modeling, prediction, measurement and control, 3 threats:
  - a packet may consume excessive CPU time at a node, causing the node to deny services to other packets,
  - an active node may be unable to schedule its resources to meet the performance requirements of packets,
  - an active packet may be unable to select a path that can meet its performance requirements.



Existing control solutions

- A limit fixed by each node, the same for all packets.
- A time-to-live for the packet fixed by the application, the same for all nodes.

- Limitations with these solutions:
  - How to choose the limit?
  - This avoids major problems but doesn’t permit optimum management.
  - Because: all applications are treated the same way.

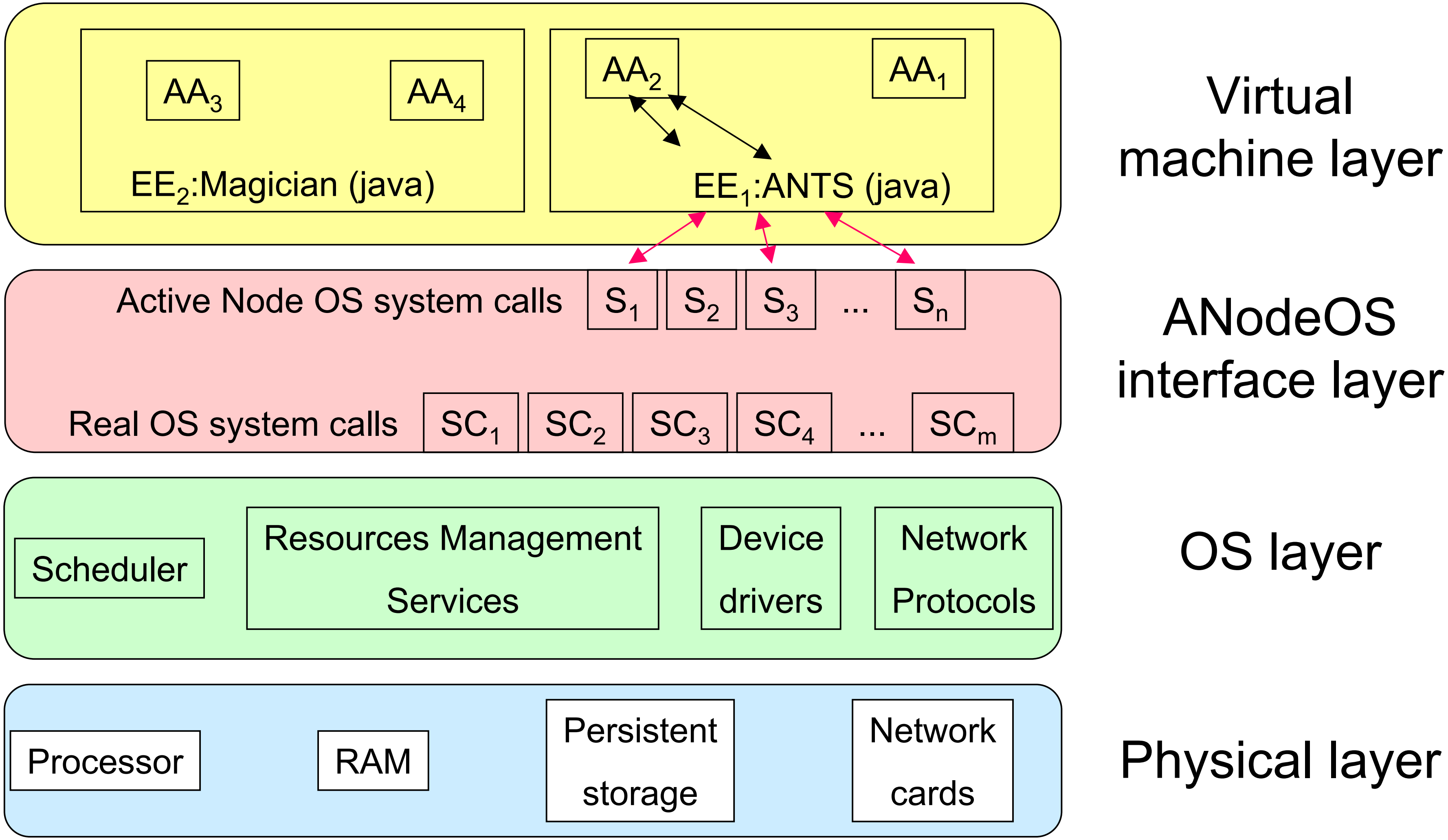
Necessity of modeling CPU requirements

- Idea to overcome these limitations: measure the CPU requirements of a packet once, and have the packet transport this information along with its data and code.
- Problem: there is no unit to measure CPU requirement that can be understood by all active nodes.

HEteROGeNOus

- It's necessary to have a model which captures all sources of variability and which can be translated on every node into a meaningful measure.

Sources of variability in processing time



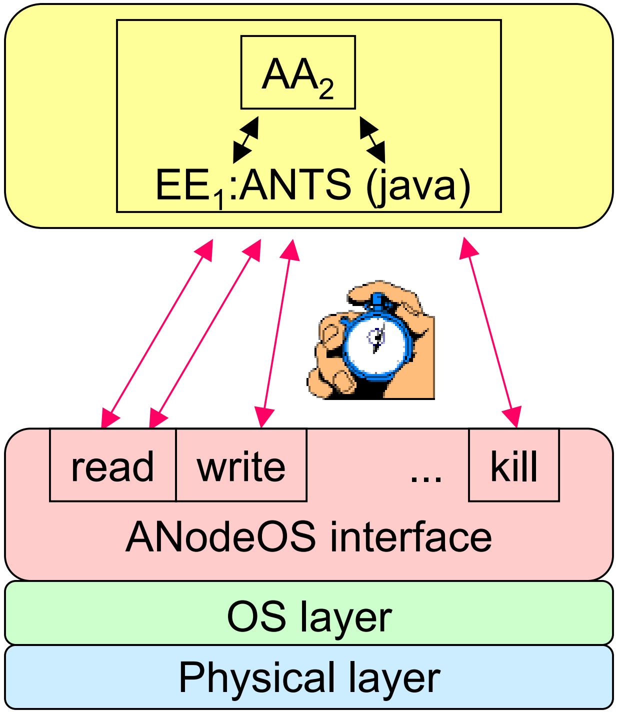
Modeling active applications: trace

Active Node OS
System calls
Monitoring

→

Execution trace

series of CPU time stamped system calls and transitions



```
…
begin, user (4 cc), read (20 cc), user
(18 cc), write(56 cc), user (5 cc), end

begin, user (2 cc), read (21 cc), user
(18 cc), kill (6 cc), user (8 cc), end

begin, user (2 cc), read (15 cc), user
(8 cc), kill (5 cc), user (9 cc), end

begin, user (5 cc), read (20 cc), user
(18 cc), write(53 cc), user (5 cc), end

begin, user (2 cc), read (18 cc), user
(17 cc), kill (20 cc), user (8 cc), end
…
```

Modeling active applications: model M1

Execution trace → Model M1

(suited for ANTS applications)

```
…
begin, user (4 cc), read (20 cc),
user (18 cc), write(56 cc), user
(5 cc), end

begin, user (2 cc), read (21 cc),
user (18 cc), kill (6 cc), user
(8 cc), end

begin, user (2 cc), read (15 cc),
user (8 cc), kill (5 cc), user (9
cc), end

begin, user (5 cc), read (20 cc),
user (18 cc), write(53 cc), user
(5 cc), end

begin, user (2 cc), read (18 cc),
user (17 cc), kill (20 cc), user
(8 cc), end
…
```

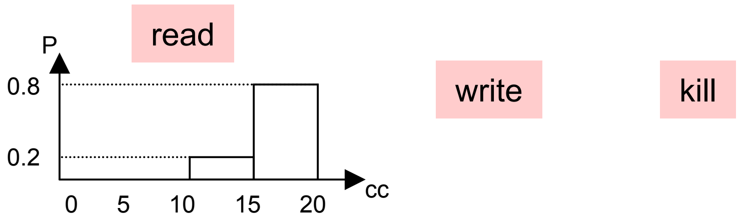
Scenario A:

sequence = “read-write”,
probability = 2/5

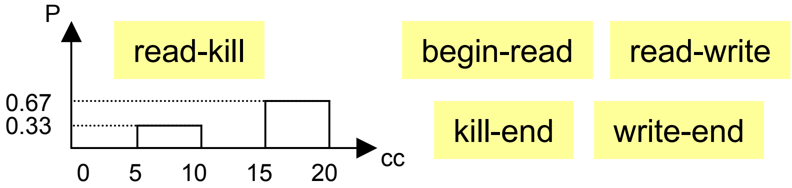
Scenario B:

sequence = “read-kill”,
probability = 3/5

Distributions of CPU time in system calls :



Distributions of CPU time between system calls :



Modeling active applications: model M2

Execution trace → Model M2

(suited for Magician applications)

```
…
begin, user (4 cc), read (20 cc),
user (18 cc), write(56 cc), user
(5 cc), end

begin, user (2 cc), read (21 cc),
user (18 cc), kill (6 cc), user
(8 cc), end

begin, user (2 cc), read (15 cc),
user (8 cc), kill (5 cc), user (9
cc), end

begin, user (5 cc), read (20 cc),
user (18 cc), write(53 cc), user
(5 cc), end

begin, user (2 cc), read (18 cc),
user (17 cc), kill (20 cc), user
(8 cc), end
…
```

Scenario A:

sequence = `begin, user (4,5 cc), read (20 cc), user (18 cc), write (54,5 cc), user (5 cc), end`

probability = 2/5

Scenario B:

sequence = `begin, user (2 cc), read (18 cc), user (14.33 cc), kill (10.33 cc), user (8.33 cc), end`

probability = 3/5

Predicting CPU requirements

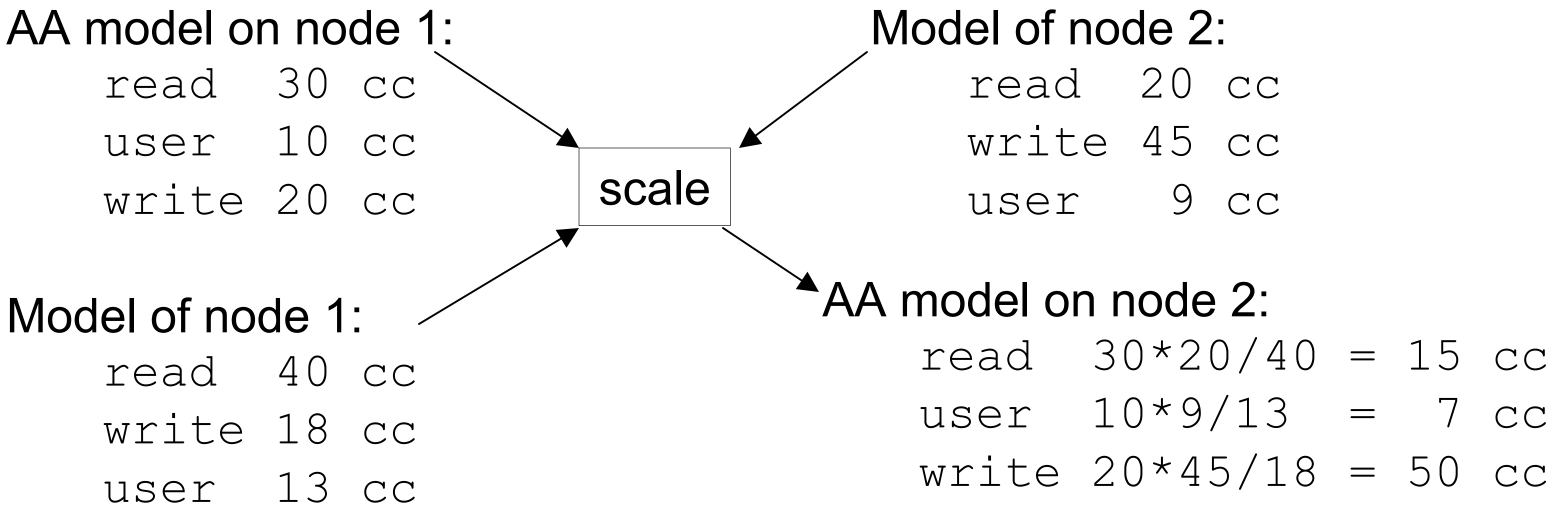
- A node needs to predict not only the average CPU time required to execute a packet but also the high percentiles (example : 95% of executions are expected to complete within 70 cc).

- Model M1: simulation
- Model M2: analytical computation

Active Network Platform	Active Application	average absolute deviation of predictions from reality (%)							
		M1, 100 bins, 20000 rep		M1, 50 bins, 20000 rep		M1, 50 bins, 500 rep		M2	
		mean	high perc.	mean	high perc.	mean	high perc.	mean	high perc.
ANTS	ping	0.859	0.9	0.643	1.622	2.696	9.8	0.028	16
	multicast	0.398	1.94	0.351	3.002	4.913	15.93	0.001	18
magician	ping	0.296	49	0.193	43			0.006	18
	route	0.991	20	0.211	19			0.001	23

Overcoming node heterogeneity: node model

- Node model:
  - a system benchmark program ⌫ for each system call, average system
  - for each EE, a user benchmark program ⌫ average time spent in the EE between system calls



AA model on node 1:

```
read   30 cc
user   10 cc
write  20 cc
```

Model of node 2:

```
read   20 cc
write  45 cc
user    9 cc
```

scale

Model of node 1:

```
read   40 cc
write  18 cc
user   13 cc
```

AA model on node 2:

```
read   30*20/40 = 15 cc
user   10*9/13  =  7 cc
write  20*45/18 = 50 cc
```

- To scale: a reference node model known by all other active nodes

Overcoming node heterogeneity: results

Platform	Application	node 1	node 2	mean	high perc.
ANTS	Ping	Daisy	Blue	2.78	4.91
		Daisy	Sloth	4.55	11.05
		Blue	Daisy	3.63	5.64
		Sloth	Blue	7.69	8.33
	Multicast	Daisy	Blue	0.32	7.29
		Blue	Daisy	3.15	11.79
		Sloth	Daisy	23.38	15.7
Magician	Ping	Blue	Daisy	11.49	20.03
		Blue	Sloth	8.01	5.2
		Daisy	Blue	7.3	37.92
	Route	Blue	Daisy	2.23	19.23
		Daisy	Blue	1.59	34.54
		Sloth	Blue	19.04	44.3

Limitations of our models

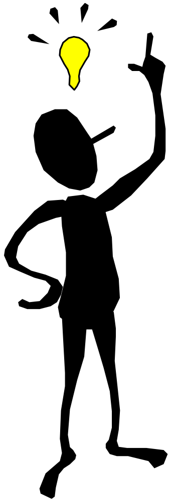
- Models can be large: O(number of scenarios, number of bins, distributions of the times).
- Simulation can be resource and time consuming: O(number of repetitions, size of the model).

- Trace-based models might represent probabilities not met in reality, if the scenario mix used to generate the traces does not represent the scenario mix actually seen on the nodes.
- Application behavior, such as looping, may depend on conditions at network nodes, and these conditions can be difficult to predict when generating the original traces.

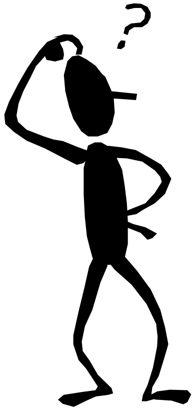
Future work

- Increase the test bed size (more nodes, more platforms, more applications)
- Investigate new models (your ideas are welcome!)
  - e.g., parameterize paths for loops
- Investigate an “Active” model:
  - gains experience as it travels through the net,
  - continuously evaluate which of the available co-existing models or prediction systems is the most accurate to return the prediction.
- Integrate our models with GE network-resource prediction system.

Your turn...



Questions, suggestions…



http://w3.antd.nist.gov/active-nets