Using the Twickel Cluster to scale up simulation

Martijn van Eenennaam

DACS Colloquium, June 15th 2012

e.m.vaneenennaam@ewi.utwente.nl
Design and Analysis of Communication Systems group
University of Twente – The Netherlands
Did you know...

FMT, SE and DACS have a Compute Cluster?
Twickel estate, near Delden, The Netherlands

Martijn van Eenennaam
1. How I use the cluster

2. How the cluster could aid in your research
1. How I use the cluster

2. How the cluster could aid in your research
A couple machines for running VMs
- 25 nodes in a compute cluster (2 quad-core Xeon CPUs, 16/24GB)
- 4 machines with lots of memory (Xeon/Opteron, with 64 – 144GB)

Fileservers
Database server
Inventory

- A couple machines for running VMs
- 25 nodes in a compute cluster (2 quad-core Xeon CPUs, 16/24GB)
- 4 machines with lots of memory (Xeon/Opteron, with 64 – 144GB)
- Fileservers
- Database server
A couple machines for running VMs
25 nodes in a compute cluster
(2 quad-core Xeon CPUs, 16/24GB)
4 machines with lots of memory
(Xeon/Opteron, with 64 – 144GB)
Fileservers
Database server
Inventory

- A couple machines for running VMs
- 25 nodes in a compute cluster (2 quad-core Xeon CPUs, 16/24GB)
- 4 machines with lots of memory (Xeon/Opteron, with 64 – 144GB)
- Fileservers
  - Database server
Inventory

- A couple machines for running VMs
- 25 nodes in a compute cluster (2 quad-core Xeon CPUs, 16/24GB)
- 4 machines with lots of memory (Xeon/Opteron, with 64 – 144GB)
- Fileservers
- Database server
Why use it?

- Over 250 CPUs
- More memory than your desktop
- Compute for days – still continue working
- Don’t fry your laptop (or overheat your office!)
Why use it?

- Over 250 CPUs
- More memory than your desktop
- Compute for days – still continue working
- Don’t fry your laptop (or overheat your office!)
Why use it?

- Over 250 CPUs
- More memory than your desktop
- Compute for days – still continue working
- Don’t fry your laptop (or overheat your office!)
Why use it?

- Over 250 CPUs
- More memory than your desktop
- Compute for days – still continue working
- Don’t fry your laptop
  (or overheat your office!)
Why use it?

- Over 250 CPUs
- More memory than your desktop
- Compute for days – still continue working
- Don’t fry your laptop (or overheat your office!)
Where to begin

1. Get an account  
   (then simply use your EWI account)

2. SSH to \textit{weldam.ewi.utwente.nl} - the head node.

Once on \textit{weldam}:

- 2 quad-core Xeon CPUs / 64GB RAM → \textit{compute}
- SSH to \textit{big3} or \textit{big4} → \textit{compute++}
- Headnode of the cluster → \textit{compute+++}
Where to begin

1. Get an account
   (then simply use your EWI account)
2. SSH to `weldam.ewi.utwente.nl`
   - the head node.

Once on `weldam`:
- 2 quad-core Xeon CPUs / 64GB RAM → `compute`
- SSH to `big3` or `big4` → `compute++`
- Headnode of the cluster → `compute+++`
- All machines mount the same filesystem:
  /home/dacs/[username]
  (this is *not* your EWI homedir as found on *demeter*)

- All machines have same environment variables:
  
  ```
  export PATH=$PATH:/home/dacs/eenenna/bin
  (i.e. your .bashrc is global)
  ```

- Now Running OpenSUSE, migration to Scientific Linux (Fermilab, CERN)

- Cluster accessible through Torque/MAUI scheduling
  and execution system (used by many supercomputers worldwide)
All machines mount the same filesystem:
/home/dacs/[username]
(this is *not* your EWI homedir as found on demeter!)

All machines have same environment variables:
export PATH=$PATH:/home/dacs/eenenna/bin
(i.e. your .bashrc is global)

Now Running OpenSUSE, migration to Scientific Linux (Fermilab, CERN)

Cluster accessible through Torque/MAUI scheduling and execution system (used by many supercomputers worldwide)
A heap of computers — and then?
Use case: VANET simulations

Martijn van Eenennaam

Tools: Twickel Cluster
Use case: VANET simulations

- Communication range (~200m)
- Max. automotive radar range (~150m)
- Minimum follow distance (~20m, 0.6s@33m/s)

Object ID, latitude, longitude, velocity, acceleration, width, length, intent

Traffic flow

Martijn van Eenennaam

Tools: Twickel Cluster

13 of 43
OMNeT++ discrete event simulation platform
http://www.omnetpp.org
RNG: Mersenne Twister generates (pseudo) random numbers

```c
scheduleAt(simTime() + exponential(1.0/lambda));
```

Event queue:

```c
initialize();
while(nextEvent){
    process(event at simTime());
}
finish();
```
Modularity

- Core + modules, channels and messages: C++
- Composition by means of a high-level language (NED)
Eclipse integration

```plaintext
simple Node {
    gates:
        inout port;
}

network ExampleNetworkModule extends CallCenter {
    parameters:
        int a;
    types: // defines locally used ch
        channel PPPLink {
            parameters:
                @display(c="red");
                double delay = 10us;
                undefinedError = 23;
        };
    module HostX extends Host {
        double linkspeed;
    };
    submodules:
        host[100]: HostX;
    connections:
        host[0].pppPort++ <-> PPPLink <-> host[1].pppPort++;
}
```
MiXiM: node mobility and propagation of radio waves

http://mixim.sourceforge.net

In this section we describe the general structure of the MiXiM simulation. MiXiM is a blackboard-based object-oriented simulation framework designed for the simulation of wireless networks. The framework consists of several layers, each with its own responsibilities.

**Connection Manager**: Manages different communication protocols and provides a general interface for collecting statistical data. The connection manager is responsible for handling the Address Resolution Protocol (ARP), which is used for energy management.
In this section we describe the general structure of the MiXiM framework and its constituent components.

The MiXiM network consists of nodes and NICs. Each node has a structure that includes an application layer, a utility layer, an ARP layer, a battery layer, and a mobility layer. The NIC structure includes a net layer, an NIC layer, a phy layer, and a phy layer.

The net layer is responsible for the network layer and manages access points (APs) and terminals. The NIC layer handles the MAC (Medium Access Control) layer design, which is usually tightly coupled and very specific for the physical layer. The physical layer is the place to take care of the reception and collision handling and is described in Section 4.3 in detail. Adjacent to the physical layer is the MAC layer, which is responsible for controlling the MAC addresses and is described in detail in Section 4.2.2.

The utility module maintains parameters that need to be accessed by more than one module within a node. One example is the position of a node, which is used to collect global parameters like the dimensions of the network objects. Objects influence the radio propagation of signals and the position of all nodes and can query object positions from the object manager.

The utility module can be seen in Figure 2(a). It is responsible for different frequency ranges such as radio and ultrasound. The wireless network interface card (NIC) module (Figure 2(b)) provides a general interface for collecting statistical data.

In general, MiXiM supports multiple connection managers, responsible for different frequency ranges such as radio and ultrasound. We thus can specify, such as Access Points (APs) and terminals. Additionally, a node can have different communication capabilities, e.g. bluetooth, GSM and ultrasound.

Details on mobility modeling and the core implementation of MiXiM are shortly described in Section 5. After this we conclude our work and give an outlook on the future of MiXiM in Section 6.
Wired connections: static, point-to-point

Wireless connections: broadcast by default

ConnectionManager:

1. Connect those modules which are in each other’s interference range based on attenuation of signals

2. Each node performs BER and PER calculations

While nodes move around, ConnectionManager updates the connections between modules.
Wired connections: static, point-to-point

Wireless connections: broadcast by default

**ConnectionManager:**

1. Connect those modules which are in each other’s interference range based on attenuation of signals

2. Each node performs BER and PER calculations

While nodes move around, ConnectionManager updates the connections between modules.
DACS MiXiM Extension to MiXiM for VANET simulations
- **IDMMobilityModel** - simple vehicle-following model (Treiber et al.)
- **BeaconAppl** impl with logging facilities
- **BeaconNetwLayer** (deterministic timer, logging)
  - **JitterBeaconNetwLayer** (Uniform Jitter generator)
  - **PoissonBeaconNetwLayer** (Poisson generator)
  - **SnifferBeaconNetwLayer** (sniffer, does not generate)
  - **CCHBeaconNetwLayer** (IEEE 1609.4 channel-switching)
- ... 
- **IEEE 802.11p MAC** (EDCA) and PHY with OFDM BER and Frame Capture
- Port of database support from old OMNeT++ version
- **IDMMobilityModel** - simple vehicle-following model (Treiber et al.)
- **BeaconAppl** impl with logging facilities
  - **BeaconNetwLayer** (deterministic timer, logging)
    - **JitterBeaconNetwLayer** (Uniform Jitter generator)
    - **PoissonBeaconNetwLayer** (Poisson generator)
    - **SnifferBeaconNetwLayer** (sniffer, does not generate)
    - **CCHBeaconNetwLayer** (IEEE 1609.4 channel-switching)
  - ...
- **IEEE 802.11p MAC** (EDCA) and PHY with OFDM BER and Frame Capture
- Port of database support from old OMNeT++ version
• **IDMMobilityModel** - simple vehicle-following model (Treiber et al.)
• **BeaconAppl impl** with logging facilities
• **BeaconNetwLayer** (deterministic timer, logging)
  • **JitterBeaconNetwLayer** (Uniform Jitter generator)
  • **PoissonBeaconNetwLayer** (Poisson generator)
  • **SnifferBeaconNetwLayer** (sniffer, does not generate)
  • **CCHBeaconNetwLayer** (IEEE 1609.4 channel-switching)
  • ...
• **IEEE 802.11p MAC** (EDCA) and PHY with OFDM BER and Frame Capture
• Port of database support from old OMNeT++ version
- IDMMobilityModel - simple vehicle-following model (Treiber et al.)
- BeaconAppl impl with logging facilities
- BeaconNetwLayer (deterministic timer, logging)
  - JitterBeaconNetwLayer (Uniform Jitter generator)
  - PoissonBeaconNetwLayer (Poisson generator)
  - SnifferBeaconNetwLayer (sniffer, does not generate)
  - CCHBeaconNetwLayer (IEEE 1609.4 channel-switching)
- ... 
- IEEE 802.11p MAC (EDCA) and PHY with OFDM BER and Frame Capture
- Port of database support from old OMNeT++ version
- **IDMMobilityModel** - simple vehicle-following model (Treiber et al.)

- **BeaconAppl** impl with logging facilities

- **BeaconNetwLayer** (deterministic timer, logging)
  - **JitterBeaconNetwLayer** (Uniform Jitter generator)
  - **PoissonBeaconNetwLayer** (Poisson generator)
  - **SnifferBeaconNetwLayer** (sniffer, does not generate)
  - **CCHBeaconNetwLayer** (IEEE 1609.4 channel-switching)

- ... 

- **IEEE 802.11p MAC** (EDCA) and PHY with OFDM BER and Frame Capture

- Port of database support from old OMNeT++ version
- **IDMMobilityModel** - simple vehicle-following model (Treiber et al.)
- **BeaconAppl impl** with logging facilities
- **BeaconNetwLayer** (deterministic timer, logging)
  - **JitterBeaconNetwLayer** (Uniform Jitter generator)
  - **PoissonBeaconNetwLayer** (Poisson generator)
  - **SnifferBeaconNetwLayer** (sniffer, does not generate)
  - **CCHBeaconNetwLayer** (IEEE 1609.4 channel-switching)
- ... 
- **IEEE 802.11p MAC** (EDCA) and PHY with OFDM BER and Frame Capture 
- Port of database support from old OMNeT++ version


van Eenennaam, E.M. (2009) microSlotted 1-Persistent Flooding in VANETs, W3 workshop, University of Twente.

Compare 4 systems, 6 values of $n$, 10 repetitions: 240 runs (and a run can take several hours)
OMNeT++ supports Parallel distributed simulation

Cluster supports MPI

But: all our replications are independent!

```cpp
for i = 0:239 {
    start run(i);
}
```
OMNeT++ supports Parallel distributed simulation

Cluster supports MPI

But: all our replications are independent!

for i = 0:239 {
    start run(i);
}
OMNeT++ supports Parallel distributed simulation

Cluster supports MPI

But: all our replications are independent!

```latex
for i = 0:239 {
    start run(i);
}
```
OMNeT++ supports Parallel distributed simulation

Cluster supports MPI

But: all our replications are independent!

```plaintext
for i = 0:239 {
    start run(i);
}
```
It’s simple:

1. Compile + test on **weldam**
2. Make a PBS file:

```bash
myExperiment.pbs

#!/bin/bash
#PBS -l nodes=10:ppn=8
#PBS -l pmem=2gb
#PBS -l walltime=336:00:00
cd /home/dacs/eenennaa/dacs\_mixim/systems/beaconing
opp_runall -j80 ./beacon_sim [ arguments ] -r 0..239
```

3. Send it to the cluster:

```
qsub myExperiment.pbs
```
It's simple:

1. Compile + test on `weldam`
2. Make a PBS file:

```
#!/bin/bash
PBS -l nodes=10:ppn=8
PBS -l pmem=2gb
PBS -l walltime=336:00:00

cd /home/dacs/eenennaa/dacs\_mixim/systems/beaconing
opp_runall -j80 ./beacon_sim [ arguments ] -r 0..239
```

3. Send it to the cluster:
   `qsub myExperiment.pbs`
It’s simple:

1. Compile + test on `weldam`
2. Make a PBS file:

```
#PBS -l nodes=10:ppn=8
#PBS -l pmem=2gb
#PBS -l walltime=336:00:00
cd /home/dacs/eenenna/dacs\_mixim/systems/beaconing
opp_runall -j80 ./beacon_sim [ arguments ] -r 0..239
```

3. Send it to the cluster:
   
   `qsub myExperiment.pbs`
Dacs/fmg/se Cluster Status

Thu Jun 14 12:16:29 2012

Nodes and Reservations

http://weldam.ewi.utwente.nl/cgi-bin/mauireswww-perl.pl
Log files...

Eventlogs, vector and scalar data
Log in the Dutch language:

*adjective* — heavy, plump, chubby.

- Take up lots of storage space
- Need to be parsed to make sense

```bash
awk '{if($2=="numTrans") print $3}' $SCALARFILE > /tmp/numTrans.dat
```

... takes a LOT of time

Solution: store output in database!

eerde.ewi.utwente.nl
Log files...

Log in the Dutch language:

*adjective* — heavy, plump, chubby.

→ Take up lots of storage space
→ Need to be parsed to make sense

```bash
awk '{if($2=="numTrans") print $3}' $SCALARFILE > /tmp/[...]
```

... takes a LOT of time

Solution: store output in database!

eerde.ewi.utwente.nl
Log in the Dutch language:

*adjective* — heavy, plump, chubby.

⁻→ Take up lots of storage space
⁻→ Need to be parsed to make sense

```
awk '{if($2=="numTrans") print $3}' $SCALARFILE > /tmp/numTrans.dat
```

... takes a LOT of time

Solution: store output in database!

eerde.ewi.utwente.nl
Again, we use OMNeT++’s modularity:

class ENVIR_API cMySQLOutputScalarManager : public cOutputScalarManager {}  

Direct output to DB in stead of a file!

Simple SQL query:

```
SELECT value FROM `scalar` WHERE runid IN (84,86,87,90,92,93,95,96,97,99) AND name = 'avg_mac_delay'
```
Again, we use OMNeT++’s modularity:

```cpp
class ENVIR_API cMySQLOutputScalarManager : public cOutputScalarManager {}
```

Direct output to DB in stead of a file!

Simple SQL query:

```sql
SELECT value FROM 'scalar' WHERE runid IN (84, 86, 87, 90, 92, 93, 95, 96, 97, 99) AND name = 'avg_mac_delay'
```
... and then use PHP and gnuplot to browse and plot results in real time:
Select a Scalar:

- `appl_beacons_received`
- `avg_contention_delay`
- `avg_end_to_end_delay`
- `avg_local_contention_delay`
- `avg_local_queueing_delay`
- `avg_mac_delay`
- `avg_queueing_delay`
- `avg_remainingbackoff`
- `avg_service_time`
- `backoffTx`
- `bcFreezesBO`
- `bcFreezesPBO`
- `beacons_from`
- `beacon_generation_rate`
- `completedContentionToTX`
- `directTx`
info | reception | delay | inter-arrival time | transmission back to scalars

Selected Scalar: **appl_beacons_received**

**Select varied iteration variable:**

- beaconLength{3200 }
- beaconNumber{15000 }
- freshnessBinNumber{100 }
- freshnessBinSize{500 }
- lambda{10 }
- numHosts{10 20 30 40 60 80 120 }
- queueLength{1 2 5 }
Selected Scalar: **appl_beacons_received**
Selected Varied intervar: **numHosts**

**Select operation:**

- calculate mean + 95% confidence intervals
- find max
- find min
- get distribution
Selected Scalar: `appl_beacons_received`
Selected Varied itervar: `numHosts`
Selected Operation: `avg`

using sniffer: 1

<table>
<thead>
<tr>
<th>numHosts</th>
<th>appl_beacons_received</th>
<th>95% CI</th>
<th>-</th>
<th>+</th>
<th>reps</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>19810.6</td>
<td>16.270334668961</td>
<td>19794.329665331</td>
<td>19826.870334669</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>38245.2</td>
<td>31.900170835906</td>
<td>38213.299829164</td>
<td>38277.100170836</td>
<td>10</td>
</tr>
<tr>
<td>30</td>
<td>51708.5</td>
<td>63.971245860933</td>
<td>51644.528754139</td>
<td>51772.471245861</td>
<td>10</td>
</tr>
<tr>
<td>40</td>
<td>52428.4</td>
<td>56.085842853968</td>
<td>52372.314157146</td>
<td>52484.485842854</td>
<td>10</td>
</tr>
<tr>
<td>60</td>
<td>32393</td>
<td>80.736466665318</td>
<td>32312.263533335</td>
<td>32473.736466665</td>
<td>10</td>
</tr>
<tr>
<td>80</td>
<td>21774.5</td>
<td>86.12531786206</td>
<td>21680.374682114</td>
<td>21860.625317886</td>
<td>10</td>
</tr>
<tr>
<td>120</td>
<td>14921.2</td>
<td>51.861608684652</td>
<td>14869.338391315</td>
<td>14973.061608685</td>
<td>10</td>
</tr>
</tbody>
</table>
Do you need to compute (a LOT)?
For a long time?
Need lots of memory?
Maybe Twickel can help you out!
OMNeT++
http://www.omnetpp.org
MiXiM
http://mixim.sourceforge.net
DACS_MiXiM
contact me (e.m.vaneenennaam@ewi.utwente.nl)
Twickel Wiki:
https://fmt.ewi.utwente.nl/redmine/projects/twickel
Cluster status:
http://weldam.ewi.utwente.nl/cgi-bin/mauireswww-per
Maintenance / accounts:
Contact Enno Oosterhuis
A PHP Interface to GNUPlot:
http://php-gnuplot.sourceforge.net
## Datasets in DB

<table>
<thead>
<tr>
<th>Table</th>
<th>Records</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>run</td>
<td>300</td>
<td>15.4 KiB</td>
</tr>
<tr>
<td>scalar</td>
<td>17,429,663</td>
<td>1.1 GiB</td>
</tr>
<tr>
<td>scenario</td>
<td>2,100</td>
<td>57.3 KiB</td>
</tr>
</tbody>
</table>
avg_mac_delay.dat

set terminal postscript eps enhanced
set key box
set xlabel "Number of vehicles"
set ylabel "Delay (s)"
plot "data.dat" using 1:2 title 'avg_mac_delay.dat' with linespoint

avg_mac_delay.m

simresn = [10 20 30 40 60 80 120 ]

avg_mac_delay10 = [0.0019955533931776 0.002561462737]
avg_mac_delay10CI = [4.7164625156547E-6 6.5876602784]

figure; hold on; axis('tight'),ylabel('avg_mac_delay'),xlabel('n');
plot(nodes(:,10),resCSMA(:,10,2),'LineWidth',2),
errorbar(simresn, avg_mac_delay10, avg_mac_delay10CI);
Velocity on a circular road