Spiking Neuron Networks Distributed Event-Driven Simulations with DAMNED

Anthony Mouraud

FACETS CodeJam Workshop

June 2010, 23^{rd}

UNIVERSITÉ LUMIÈRE LYON 2

LIRIS

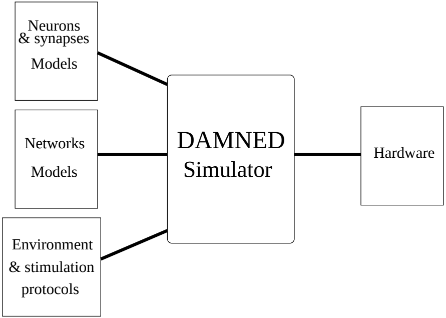
Simulation framework

- **Distributed** : allow large scale simulations
- **Event-driven** : sparse activities
- **Multithreaded** : computations and communications overlap

DAMNED :
D*istributed* **A***nd* **M***ultithreaded* **N***eural* **E***vent-***D***riven* *simulator*

[PDCN'06], [NEUROCOMP'06]

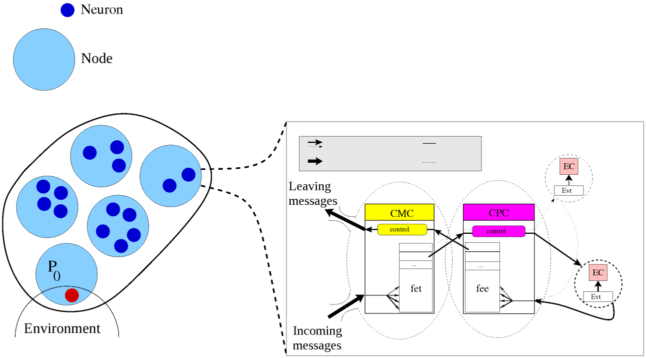
Simulation framework



DAMNED :
***D**istributed **A**nd **M**ultithreaded **N**eural **E**vent-**D**riven simulator*

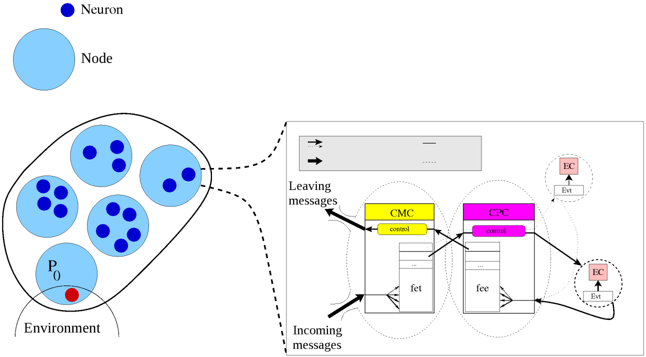
DAMNED concepts

Basic element : Event-driven Cell (EC)



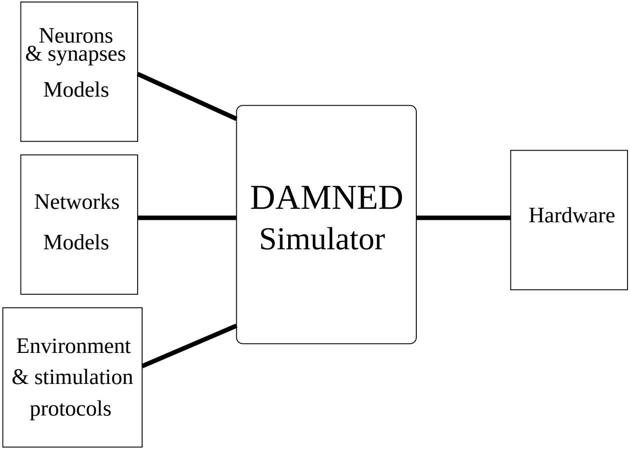
DAMNED concepts

Basic element : Event-driven Cell (EC)



- Decentralized Global Virtual Time handling
- Mutexes on shared datas structures

Simulation framework



Neurons
& synapses
Models

Networks
Models

Environment
& stimulation
protocols

DAMNED
Simulator

Hardware

Neuron models definition

C++ classes inheritance

- Neuron ← ObjetEvenementiel (EC)
- Synapses handled by neuron model
- Few methods definition

Neuron models definition

C++ classes inheritance

- Neuron ← ObjetEvenementiel (EC)
- Synapses handled by neuron model
- Few methods definition

```
class Neurone :   public ObjetEvenementiel{
public :
    Neurone();
    ~Neurone();
    Evenement* runRecepEvt(Evenement*);
    void ajouterConnexion(ObjetEvenementiel*);
    unsigned int getDelai(const Point4D&);
}
```

Event PA

date	n source

Neuron models definition

C++ classes inheritance

- Neuron ← ObjetEvenementiel (EC)
- Synapses handled by neuron model
- Few methods definition

- Portability
- Very few constraints on models

Event-driven constraint

- Irregular updates

  ⟶ Predictions

  ⟶ Coupled ED

Synaptic connexions

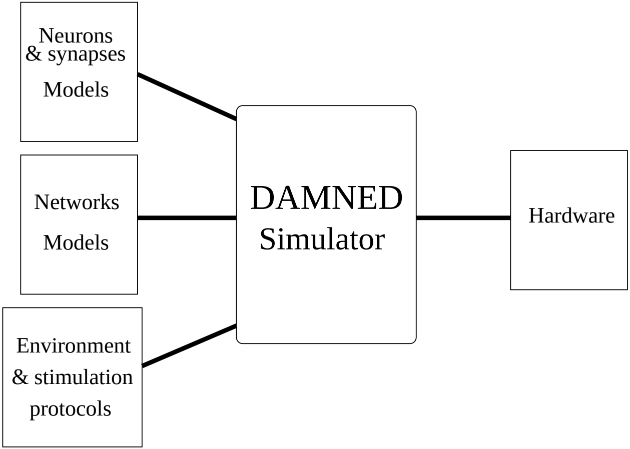
Weights

- Post-synaptic handling
- Unsupervised local learning processes

Delays

- Applied post-synaptically
- Delay learning / shift

Simulation framework



Neurons & synapses

Models

Networks

Models

Environment & stimulation protocols

DAMNED Simulator

Hardware

Network models

Network

- Populations set
- Projections set

Network models

Network

- Populations set defining :
  - Neuron model
  - size
  - name, etc
- Projections set

Inside API :

Network net = createNewNetwork() ;
net.setNbPop(N) ;

net.setNomPop(1, "*E*xcit 2D MAP") ;
net.setNeuronModel(1, "*N*euronTypeT") ;
net.setTaillePop(1, 100) ;
...

Network models

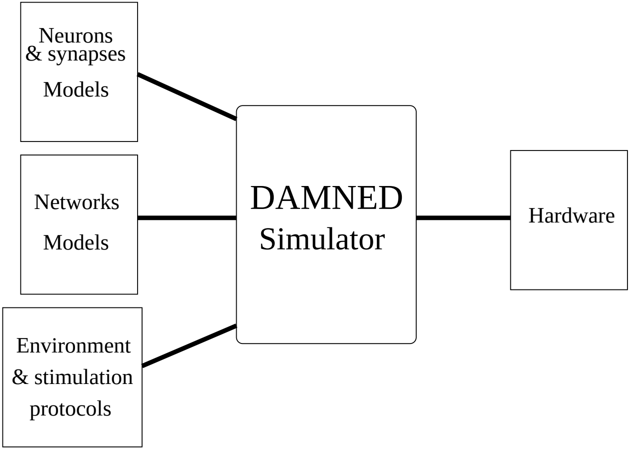
Network

- Populations set
- Projections set defining :
  - Projection
  - Weight
  - Specific parameters

Inside API :

```
int aproj = net.addProjection( 1, 2, ”SomeProjType” ) ;
net.setPoidsProj(aproj, 12) ;
net.setParamSpec(aproj, ”paramName”, 0.5 ) ;
```

Simulation framework



Environnement models

Particular role : online interactions

- Applies stimulations (through input cells)
- Gets output activities (through output cells)

- May act outside simulation (e.g. robot command)

Environment is a full process

- A Thread running during simulation
- A specific (EC) : CPC → **run()** method definition

Environnement models

Particular role : online interactions

- Applies stimulations (through input cells)
- Gets output activities (through output cells)
- May act outside simulation (e.g. robot command)

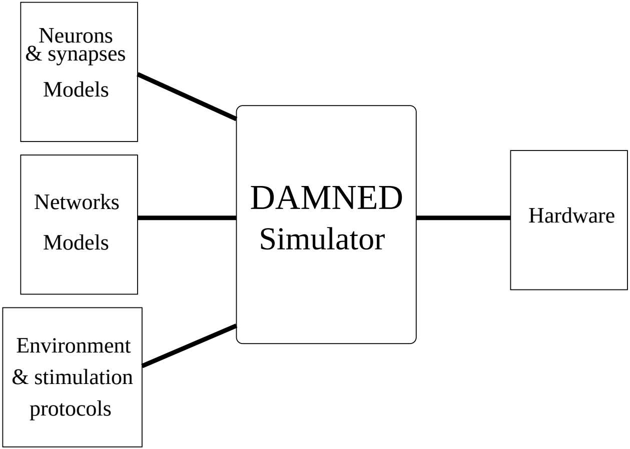
Environment is a full process

- A Thread running during simulation
- A specific (EC) : CPC → **run()** method definition

Inside API :

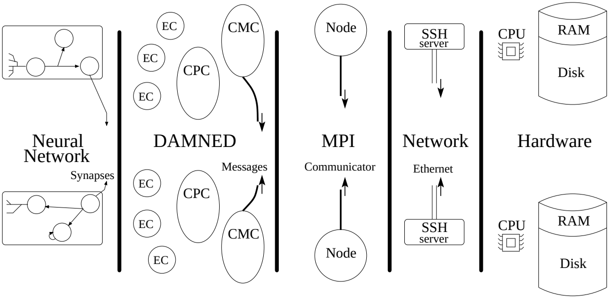
setEnvModel("*M*odEnvironment") ;
net.setNbPopIn(2) ;
net.setNbPopOut(2) ;

Simulation framework

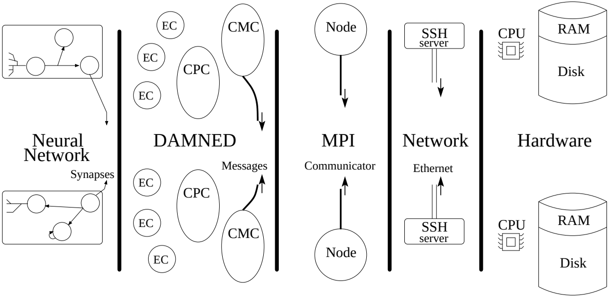


Distributed hardware

- Mono or multi-core stations
- LAN, Clusters, Grids
- Parallel machines



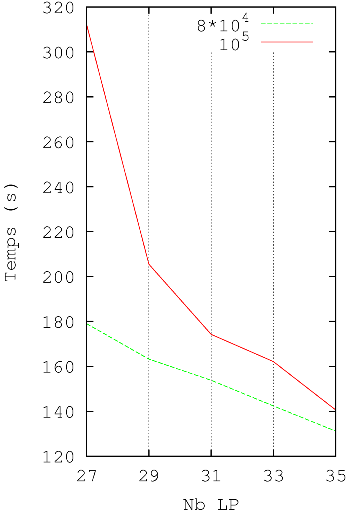
Distributed hardware



Inside API :

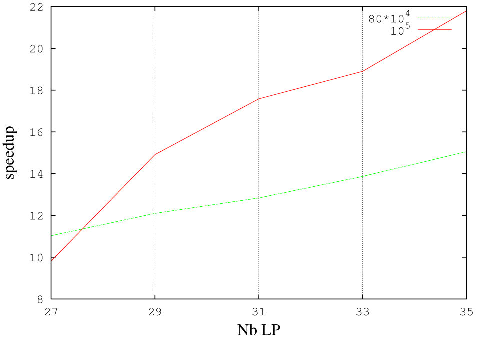
```
setNbNodes( 3 ) ;
defHardware( "FicHosts" ) ;
addMappingRange( 0, 0, 300 ) ;
addMappingRange( 1, 301, 2000 ) ;
addMappingRange( 2, 2001, 4999 ) ;
```

Large scale networks : execution time



- Stations : bi-cores 2Ghz, 1Go Ram
- Neuron model : LIF
- Nb neur. : $8 * 10^4$ et 10^5
- Nb syn. : $80 * 10^6$ et $100 * 10^6$
- Duration : 1 s
- Stim. freq. : 600 Hz
- Average activity : 1 Hz

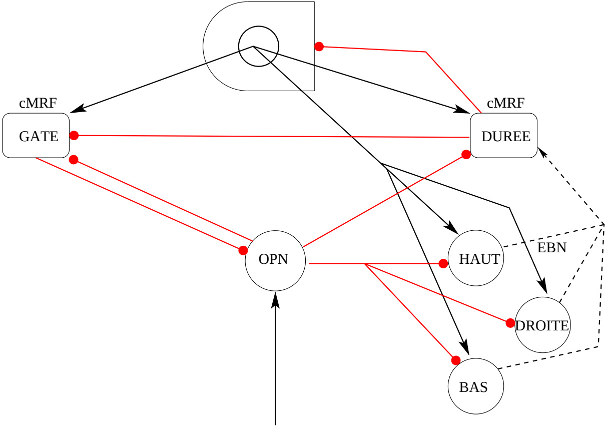
Large scale networks : speedups



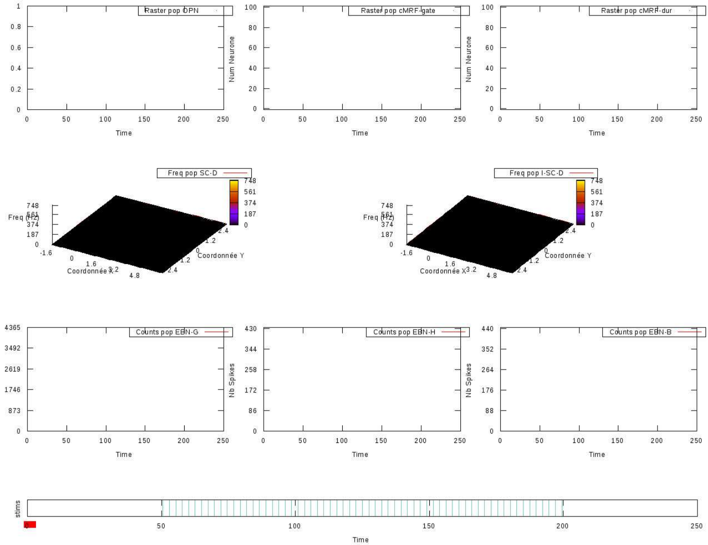
- Speedup : $\frac{T\ Sequential}{T\ Parallel}$
- Extrapolated sequential time

[EANN'09]

Saccade Burst Generator



Modèle élaboré



Simulator dependencies

- make
- C++ compiler
- libdl (dynamic library load)
- Posix threads
- MPI (MPICH2)
- ssh

Plots : imagemagick (convert tool)

The end

Thanks for your attention

http ://sourceforge.net/projects/damned