



# Les capteurs en ville

## Du maquettage à la génération de code

Bernard Pottier,  
Pierre Yves Lucas, Eloi Keita  
pottier@univ-brest.fr

*Université de Brest (UBO), France*  
*\*LabSTICC, UMR 3192*  
*Dept Informatique, Faculté des Sciences*

# Plan

1. Contexte : *Systemes pervasifs* LabSTICC
2. Une proposition de flot pour les WSN : NetGen
3. Outils : Forme textuelle du modèle topologique
4. Outils : Création interactive de réseaux
5. Suite du flot : brève présentation

# Systemes pervasifs, LabSTICC

- *Thème* de l'équipe Méthodes et Outils pour la Conception des Systemes
  - CAPNET : gestion de l'énergie dans les réseaux de capteurs, étude de cas (N.Julien)
  - RO (M. Sevaux) : optimisation pour les couvertures dans les réseaux de capteurs
  - Réalité Virtuelle augmentée et Mer (JP Diguët)
  - NetGen : flot de conception et outils associés (B.Pottier)

# Contributions proposées à RESSACS 2011

- Introduction aux outils de spécification NetGen
- Calculs et estimation de couvertures sur GIS  
(R. Herry, K. Ammouche, Y. Le Gall)
- Intégration logicielle des capteurs  
(Pierre-Yves Lucas)

# Proposition de flot de conception

**L'atelier NetGen propose un flot descendant inspiré de la conception matérielle:**

- **Etape 1:** conception, simulation, optimisation.
- **Etape 2 :** production du code pour les capteurs, tests et déploiement (Cf P.Y Lucas)

*Bouclages sur chacune des étapes et sur le flot complet.*

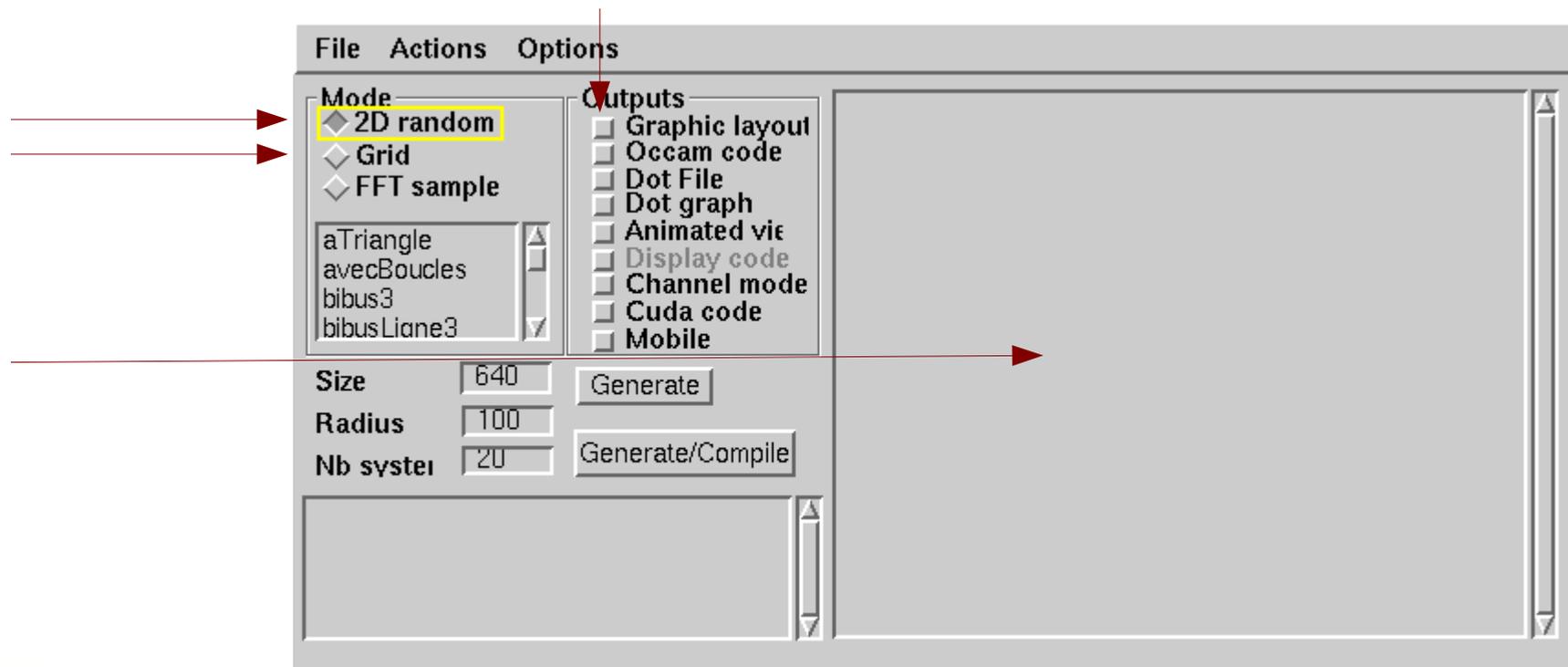
# Structuration des outils

- Outil de développement principal : Smalltalk
  - VisualWorks : prototypage rapide, constructions de modèles. Licence universitaire gratuite
- Pour l'expression et la simulation systèmes
  - Occam, compilateur de U. de Kent, libre
  - CUDA, compilateur et kit nvcc de Nvidia, licence
  - Graphviz, dot, etc..
- Matériel principal : msp430, mspgcc4
  - Études de capteurs et systèmes matériels

# Spécification géométrique (1)

- **Tableau de bord NetGen :**

- Générations de distributions aléatoires
- Réseaux réguliers
- Explications sur la forme textuelle du modèle :  $P_i \{ P_j P_k P_n \}$  Programme



# Spécification géométrique (2)

File Actions Options

Mode

- 2D random
- Grid
- FFT sample

aTriangle  
avecBoucles  
bibus3  
bibusLiane3

Size: 400  
Radius: 200  
Nb system: 5

Generate

Generate/Compile

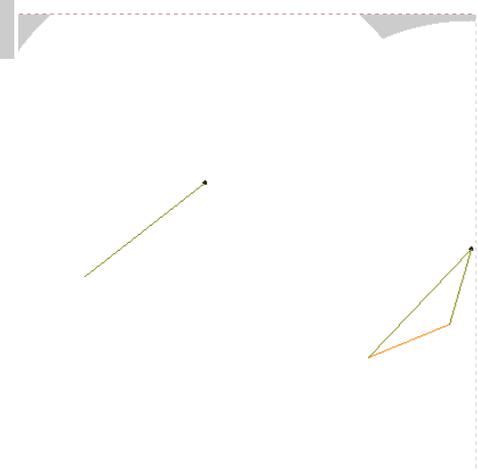
genRange200Points5

messages none defined.

P1 { P4, P5 } Node (377 @ 271) (200)  
P2 { P3 } Node (58 @ 229) (200)  
P3 { P2 } Node (163 @ 147) (200)  
P4 { P1, P5 } Node (306 @ 300) (200)  
P5 { P1, P4 } Node (396 @ 205) (200)

genRange200Points5  
processus : 5  
min fanout : 1

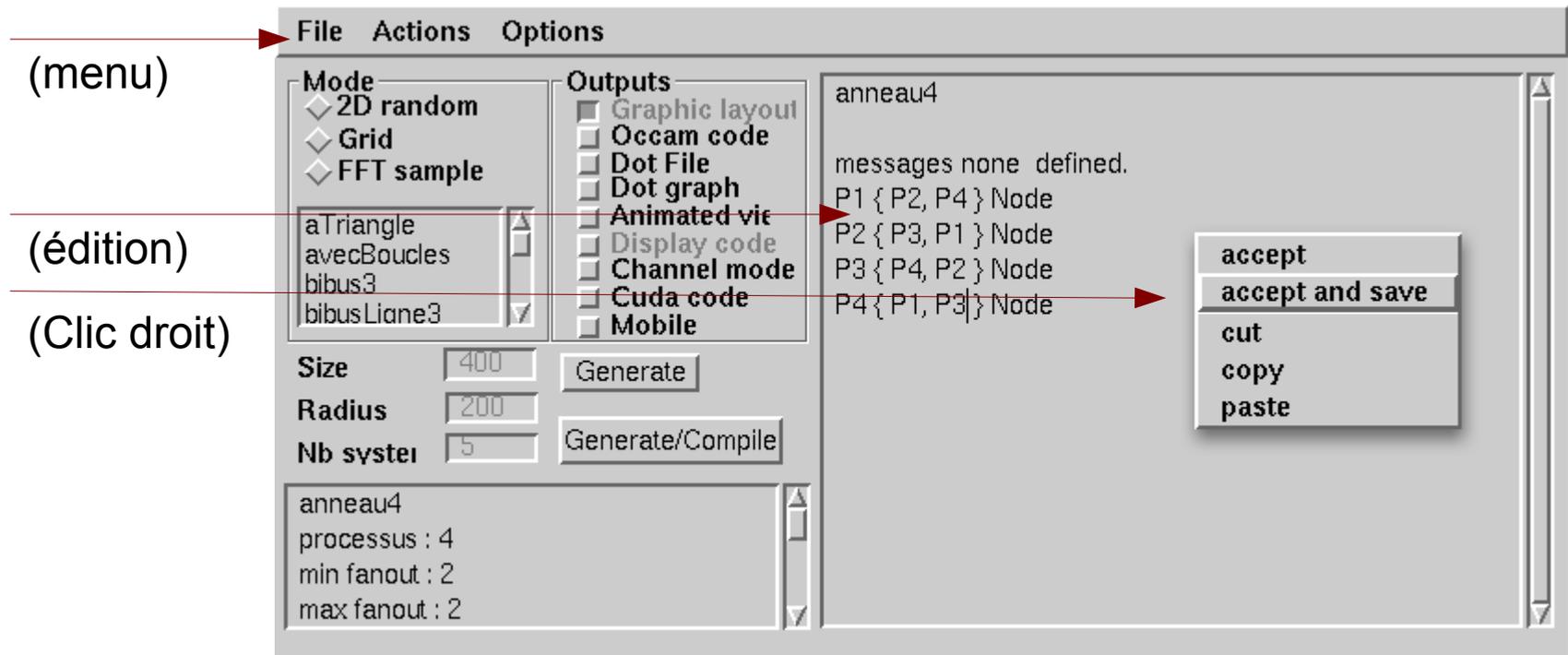
Modèle



# Spécification textuelle / fichiers

- **Editeur de spécification**

- Syntaxe spécification : titre messages ( $P_i \{ P_j P_k P_n \}$  Programme) ..
- *Load et save* : possibilité d'écrire vos propres générateurs.



# Spécifications régulières

- Grilles de connectivité variable

File Actions Options

Mode

- 2D random
- Grid**
- FFT sample

anneau4  
aTriangle  
avecBoucles  
bibus3

Outputs

- Graphic layout
- Occam code
- Dot File
- Dot graph
- Animated vie
- Display code
- Channel model
- Cuda code
- Mobile

Size: 200

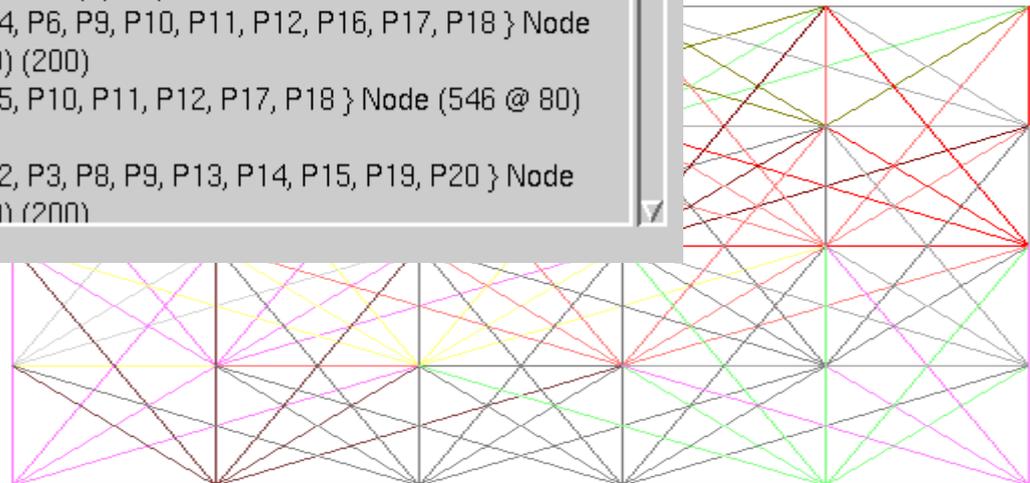
Radius: 200

Nb system: 30

Generate

Generate/Compile

```
gridRange200Points30
messages none defined.
P1 { P2, P3, P7, P8, P9, P13, P14 } Node (91 @ 80) (200)
P2 { P1, P3, P4, P7, P8, P9, P10, P13, P14, P15 } Node
(182 @ 80) (200)
P3 { P1, P2, P4, P5, P7, P8, P9, P10, P11, P14, P15, P16 }
Node (273 @ 80) (200)
P4 { P2, P3, P5, P6, P8, P9, P10, P11, P12, P15, P16, P17 }
Node (364 @ 80) (200)
P5 { P3, P4, P6, P9, P10, P11, P12, P16, P17, P18 } Node
(455 @ 80) (200)
P6 { P4, P5, P10, P11, P12, P17, P18 } Node (546 @ 80)
(200)
P7 { P1, P2, P3, P8, P9, P13, P14, P15, P19, P20 } Node
(91 @ 160) (200)
```



# Présentation graphe logique

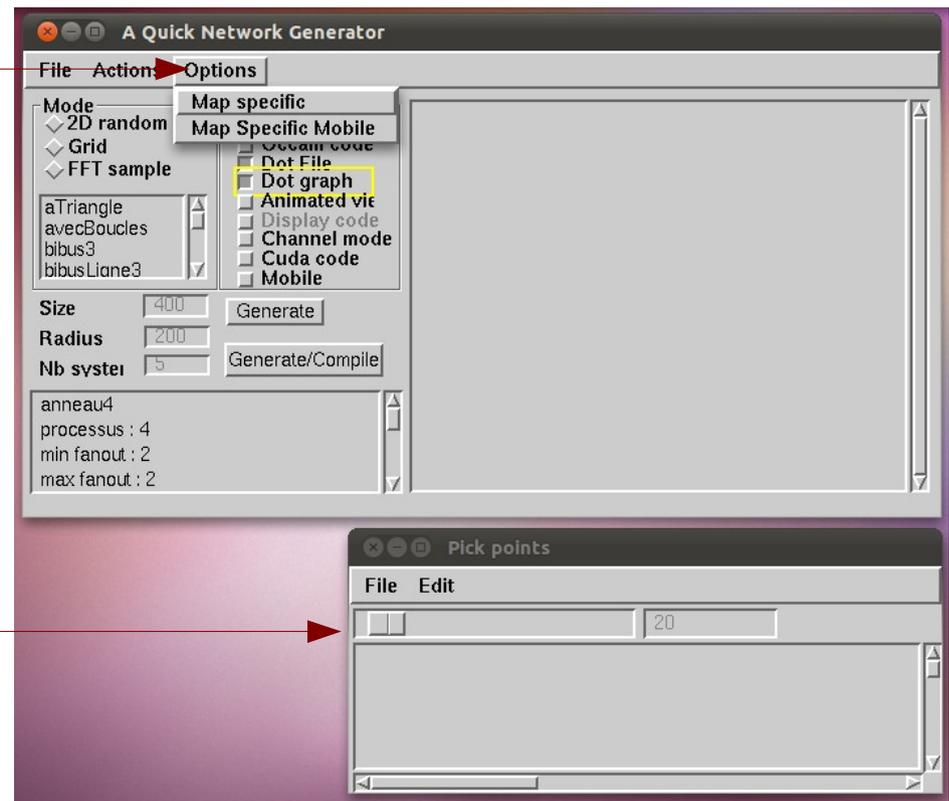
- Utilisation du logiciel graphviz
  - Ré-écriture du modèle en syntaxe *dot*
  - Si graphviz : production de présentations graphiques logiques

The screenshot shows the Graphviz software interface. The menu bar includes 'File', 'Actions', and 'Options'. The 'Mode' section has radio buttons for '2D random', 'Grid', and 'FFT sample'. The 'Outputs' section has checkboxes for 'Graphic layout', 'Occam code', 'Dot File', 'Dot graph', 'Animated vie', 'Display code', 'Channel mode', 'Cuda code', and 'Mobile'. The 'Size' is set to 400, 'Radius' to 200, and 'Nb system' to 5. The 'Generate' and 'Generate/Compile' buttons are visible. The main window displays the text 'anneau4' and 'messages none defined.' followed by node definitions: 'P1 { P2, P4 } Node', 'P2 { P3, P1 } Node', 'P3 { P4, P2 } Node', and 'P4 { P1, P3 } Node'. A context menu is open over the graph area, showing options: 'accept', 'accept and save', 'cut', 'copy', and 'paste'. Below the interface, a graph diagram shows four nodes (P1, P2, P3, P4) connected by directed edges with labels: P1.P2, P2.P3, P3.P4, P4.P3, P3.P2, P2.P1, P1.P4, and P4.P1.

# Saisie sur carte (1)

- **But : positionnement de capteurs sur un plan**
  - Accès fenêtre de saisie graphique

Menu option : map

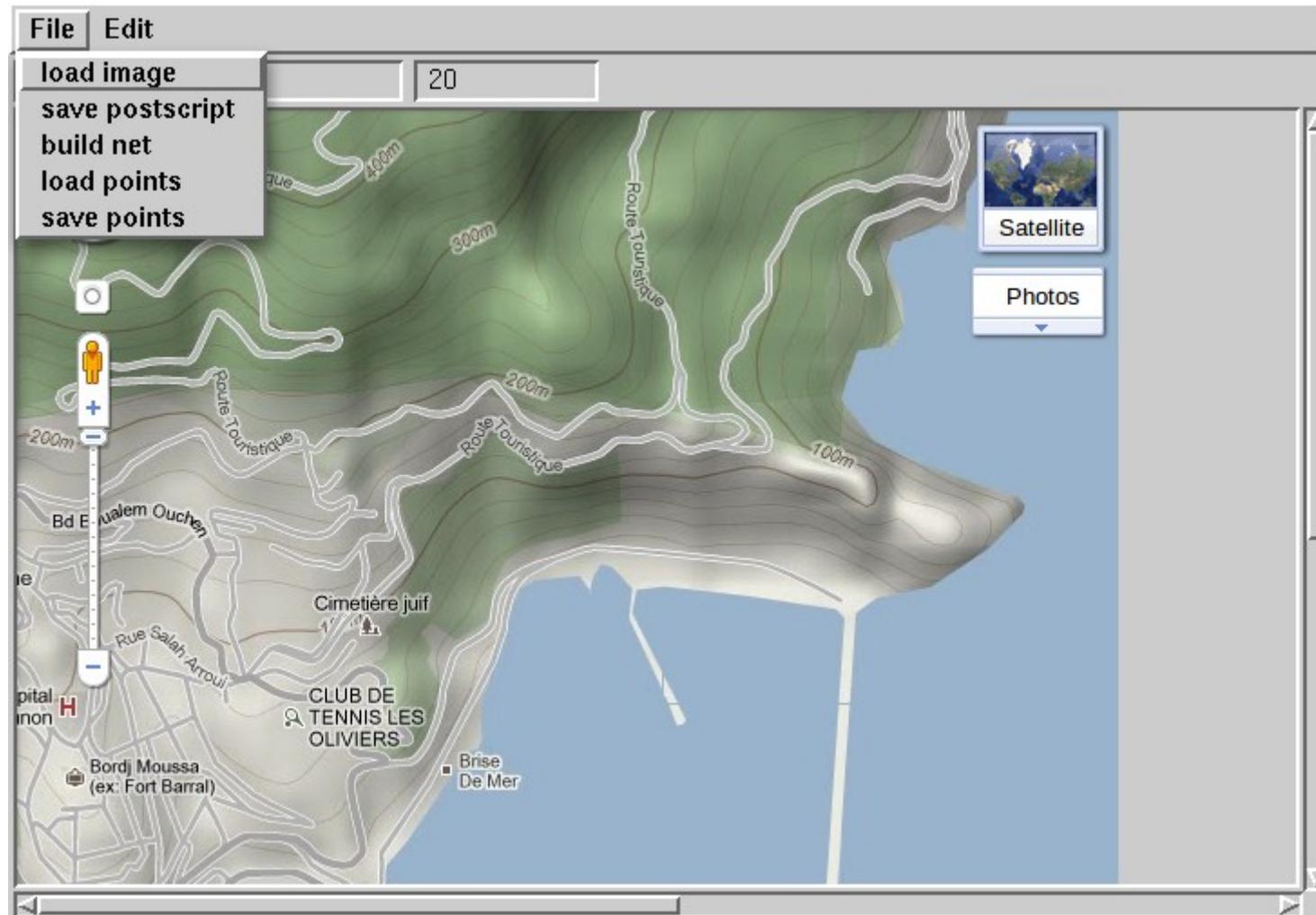


Fenetre graphique

# Saisie sur carte (2)

## Chargement d'une image (.png)

- Portail IGN (France), Google maps, etc... plans batiments



# Saisie sur carte (3)

## Positions des capteurs

- Selections de points
- Echelle en vue
- Dessin du réseau immédiat
- Portée des capteurs réglable
- *Ici : portées circulaires*



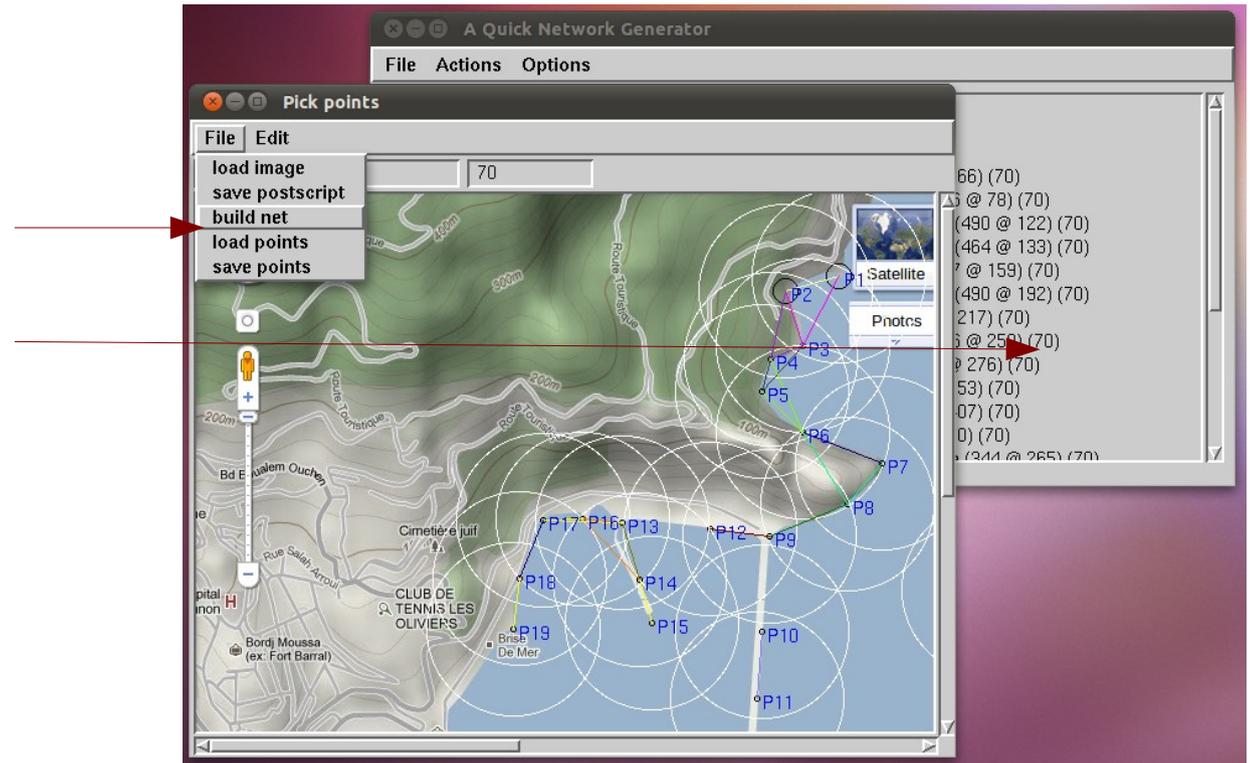
# Saisie sur carte (4)

## Production du modèle

- Transfert vers les outils du modèle
- Production d'un modèle annoté sur la carte

## Chemins de mobiles

- similaire





# Alternative : saisie par GPS

- **'Data logger' MerSea**
  - Micro controleur + GPS, boutons de saisie
  - Mesures de portée sans fil
  - *Promenade* sur site
  - Analyse en laboratoire



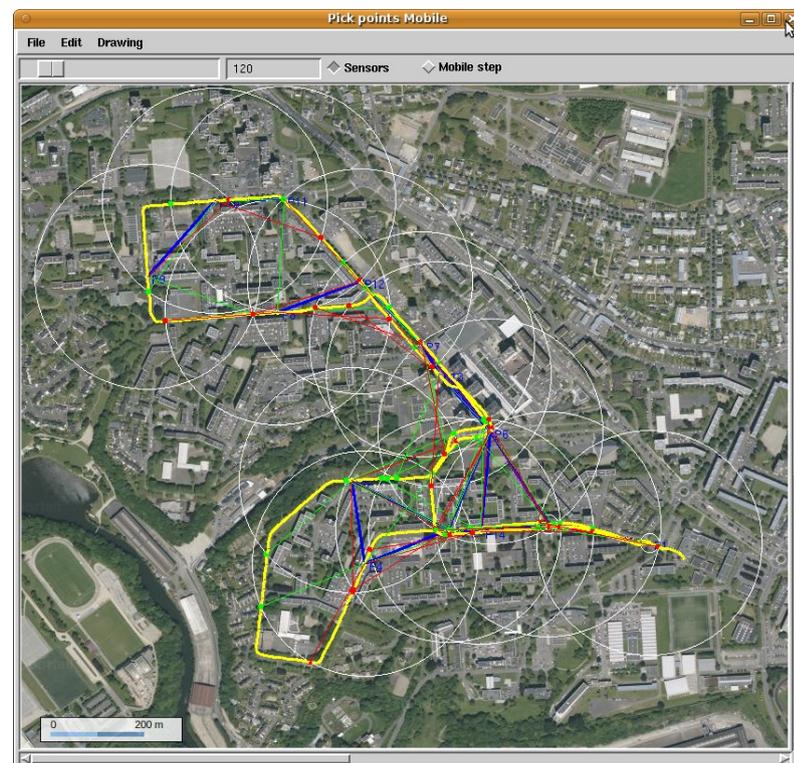
GPS

C2420 MSP

Pols

# Saisie par GPS : ré-écritures

- **Analyse des données et reformulation**
  - KML / Google map
  - Qgis (apt-get install qgis)

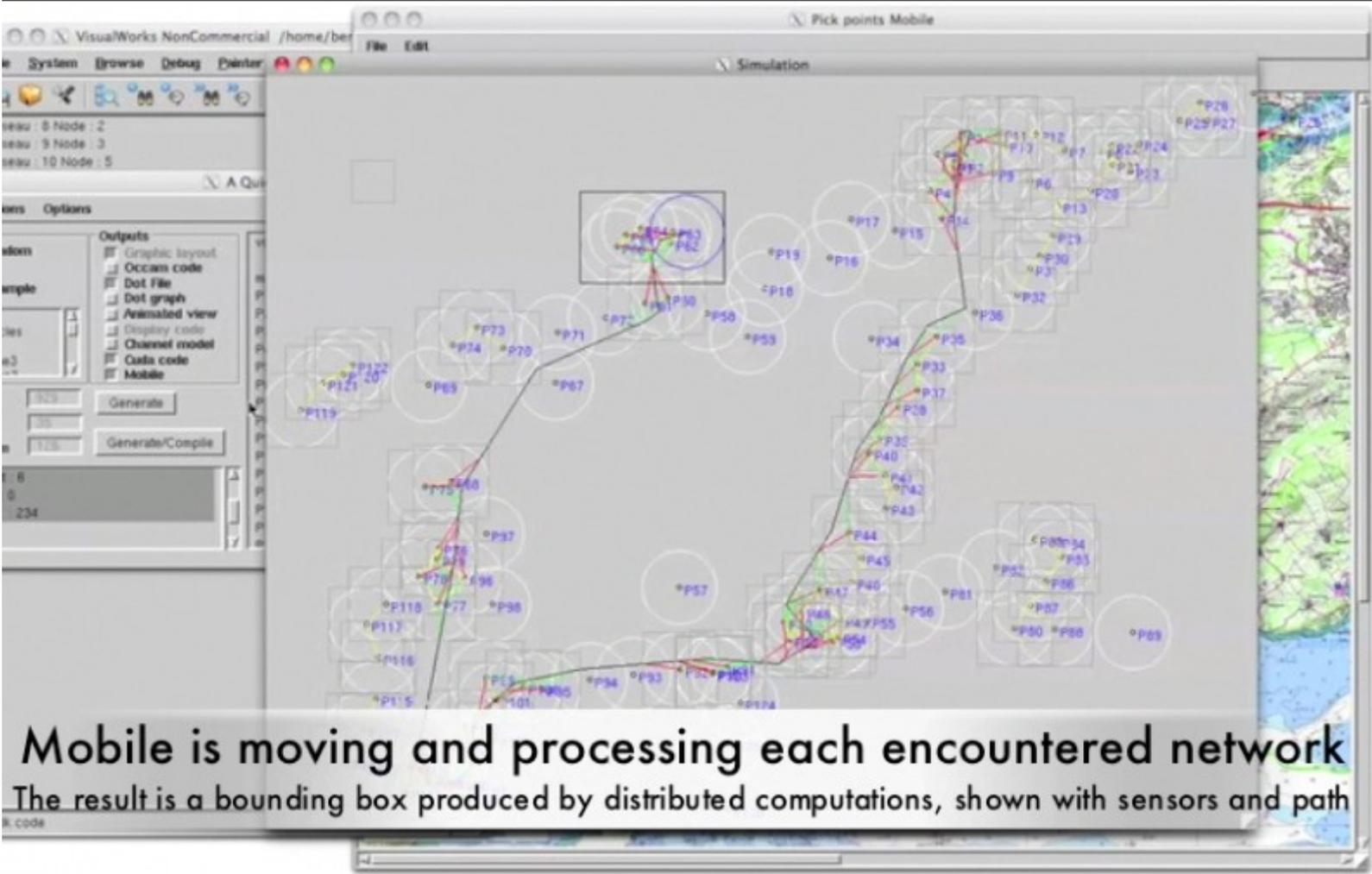


# *Proposition de flot de conception (rappel)*

**L'atelier *NetGen* propose un flot descendant inspiré de la conception matérielle:**

- **Etape 1:** conception, simulation, optimisation.
  - *Productions de code Occam, compilation, simulation*
  - *Ou Production de code CUDA, compilation et test avec feed back graphique et mobilité.*

# Exemple : réseau cotier avec 2 portées différentes et 1 mobile



The screenshot shows a simulation window titled "Pick points Mobile" with a "Simulation" tab. The main area displays a map of a coastline with numerous nodes labeled P1 through P38. Each node is represented by a small circle with a larger, semi-transparent circle around it, indicating its range. A path of nodes is highlighted in red, starting from the top right and moving along the coast. A bounding box is drawn around the nodes encountered by the mobile node. The left sidebar contains a "System" menu, a "Simulation" status bar, and a "Outputs" section with options like "Graphic layout", "Occam code", "Dot File", "Dot graph", "Animated view", "Display code", "Quannet model", "Guts code", and "Mobile".

**Mobile is moving and processing each encountered network**  
The result is a bounding box produced by distributed computations, shown with sensors and path

Mise en mémoire tampon : 100,00% 2:00 /

# Evolution

- **Passages de couvertures radio grossières à des estimations plus fines**
- **Application de ces techniques à des modélisations physiques conjointes**
- **Faisabilité de réalisations matérielles**

**Merci .**

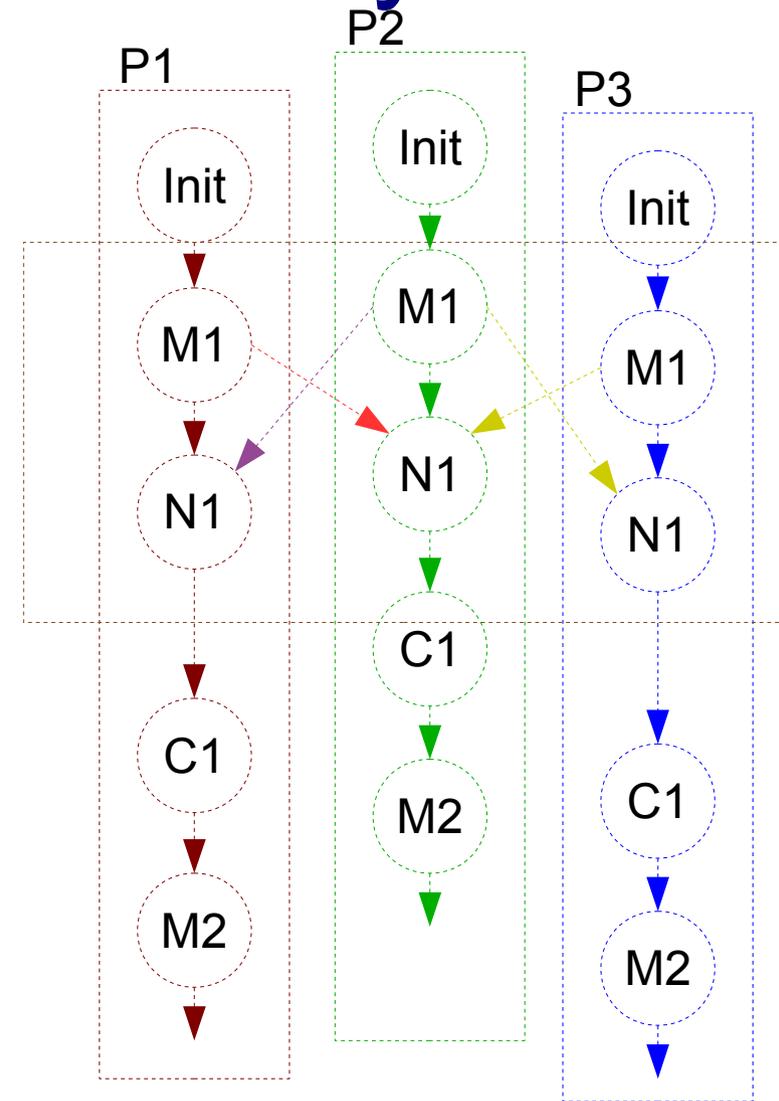
# Modèle synchrone : ref. N. Lynch

## Comportement abstrait

- $M$  : message send
- $N$  : message receive
- $C$  : state change and next message production

## Méthodes d'exécution

- Canaux bloquant Occam
- Barrières SIMD sur GPU
- Comm. TDMA par phases



# Processus communicants : CSP/Occam ... (Ref. Hoare)

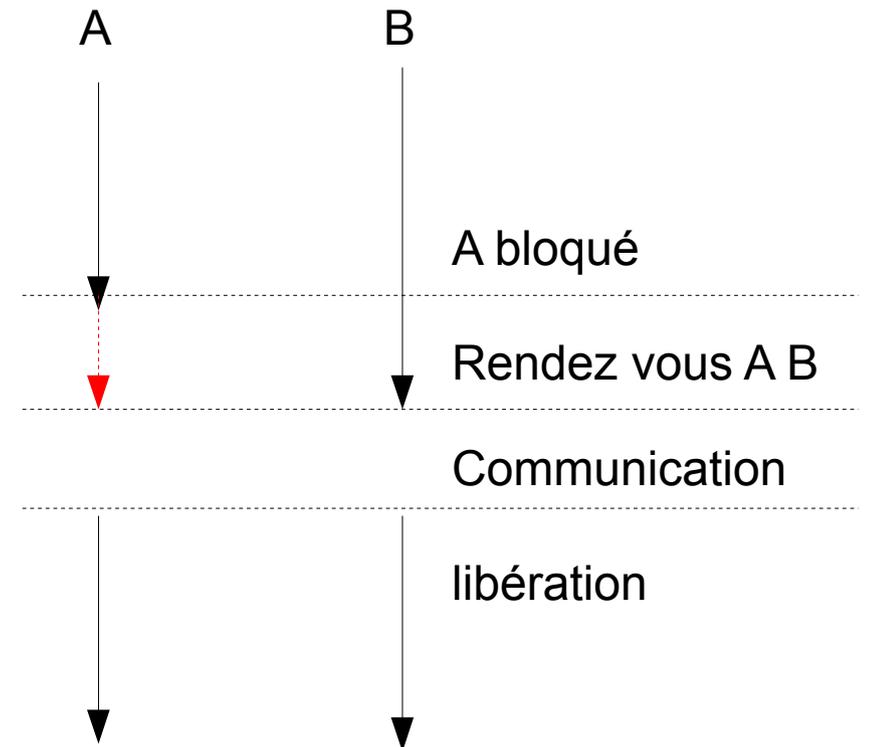
CHAN OF BYTE c :

BYTE val :

PAR

c ! 'x' – ecriture A

c ? val – lecture B



# Processus communicants : Modèle synchrone

- Si P1 et P2 sont connectés :

- P1 écrit à P2
- P1 reçoit de P2
- P2 écrit à P1
- P2 reçoit de P1

- Obligation de procéder en parallèle pour éviter un deadlock !

Echantillon de code pour les phases  $M_i$   $N_i$  du modèle synchrone

```
PAR
  PAR i=0 FOR SIZE in
    in[i] ? CASE
      table ; inMessages[i]
      SEQ
        tags[i]:=TRUE
        externalChange :=TRUE
      null ; nullByte
        tags[i]:=FALSE
  PAR j=0 FOR SIZE out
    out[j] ! table; outMessages[j]
```

# Processus communicants : Simulation

- 100 % de la topologie générée
  - Construction parallèle pour le réseau
  - Présence d'un traceur interne
- 100 % du comportement à écrire
  - Largement réutilisable
  - Code de haut niveau, fortement concurrent (PAR)
- Compilation multi\_threadée adaptée aux processeurs multi coeurs
  - Scheduler embarqué dans l'exécutable !



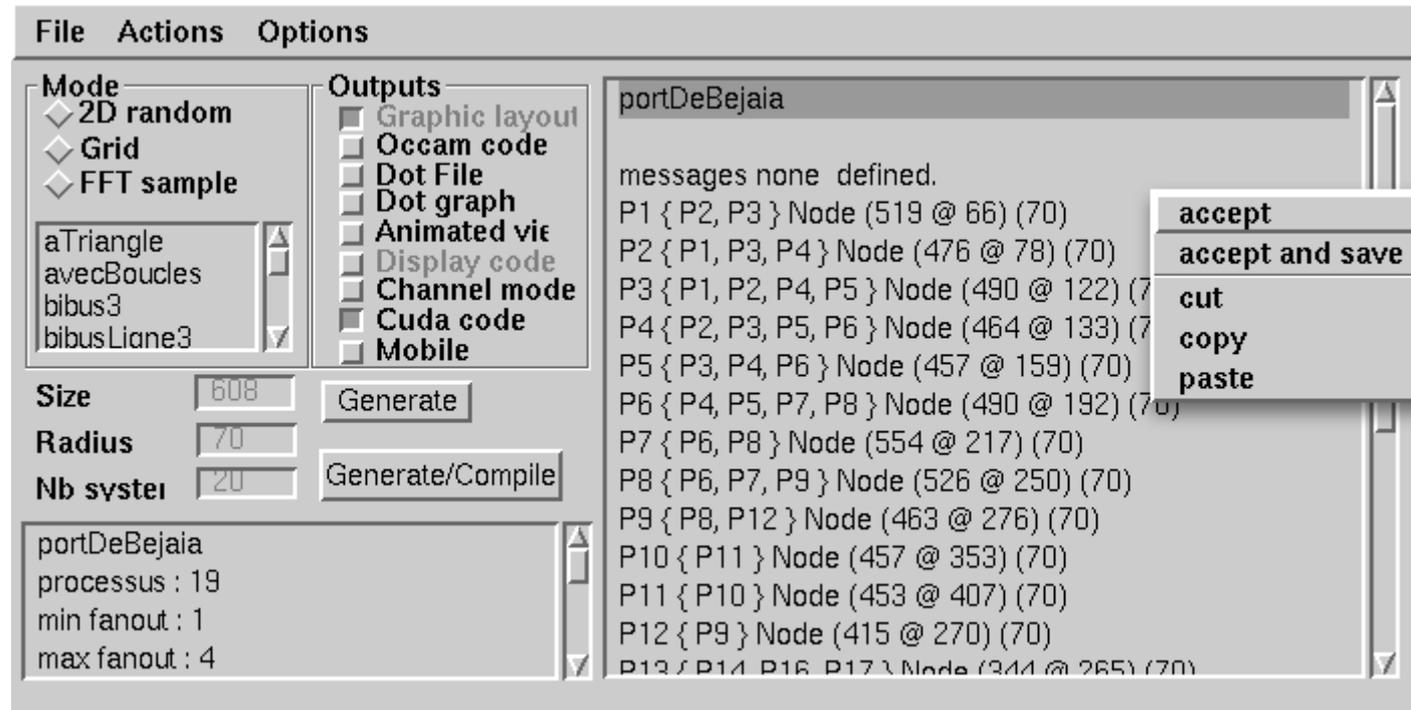
# Simulation sur GPU

- 1 *thread* par processus
  - 1 *kernel* par comportement différent
  - Parallélisme *implicite*, tout le monde fait la même chose
- Les canaux sont implantés dans des tableaux, les transferts sont 'synchrones'
  - Contenant les valeurs émises
  - Qui sont lus par les voisins

# Simulation sur GPU

```
typedef struct s_mapped
{
    int node;
    int canal;
}mapped;
```

```
typedef struct s_canaux
{
    int nbOut;
    int nbln;
    int nbDyn;
    mapped write[MAX_FANOUT];
    mapped read[MAX_FANOUT];
    mapped
writeDyn[DYNAMIC_CHAN];
    mapped readDyn[DYNAMIC_CHAN];
}canaux;
```



```
canaux channels_h[] =
{ // start array
    {2,2,5,{{0,0},{0,1}},{{1,0},{2,0}},{{0,0},{0,1},{0,2},{0,3},{0,4}},{{-1,-1},
{-1,-1},{-1,-1},{-1,-1},{-1,-1}},
    {3,3,5,{{1,0},{1,1},{1,2}},{{0,0},{2,1},{3,0}},{{1,0},{1,1},{1,2},{1,3},
{1,4}},{{-1,-1},{-1,-1},{-1,-1},{-1,-1},{-1,-1}},
    {4,4,5,{{2,0},{2,1},{2,2},{2,3}},{{0,1},{1,1},{3,1},{4,0}},{{2,0},{2,1},
{2,2},{2,3},{2,4}},{{-1,-1},{-1,-1},{-1,-1},{-1,-1},{-1,-1}}},
```

# Simulation sur GPU

- Production d'une ossature décrivant les communications
- Compilation nvcc
- Exécutable ou librairie dynamique
- Intégration au contrôleur Smalltalk pour la visualisation

Merci ..