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 : S*ystèmes 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

Systèmes pervasifs, LabSTICC

- *Thème* de l'équipe Méthodes et Outils pour la Conception des Systèmes
  - 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 Diguet)
  - 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.

1

2

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

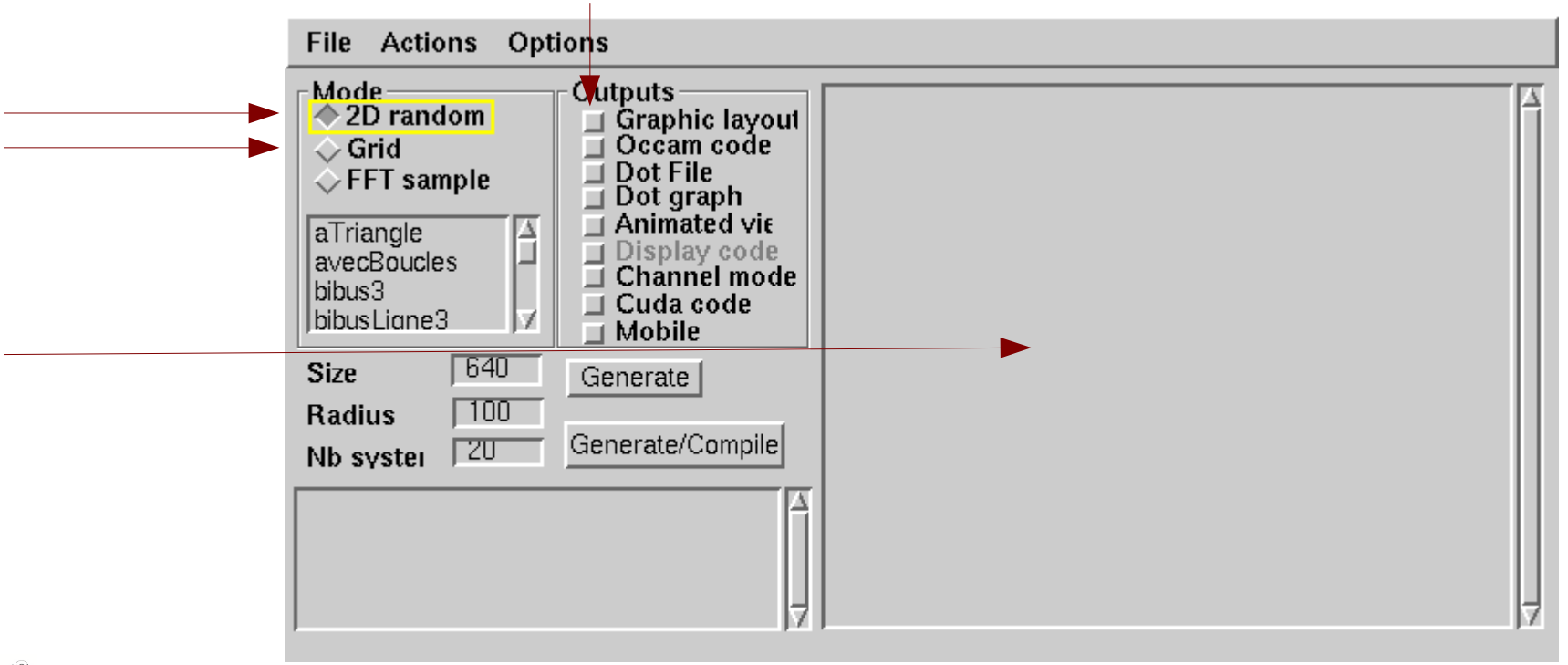
  Bouclages sur chacune des étapes et sur le flot complet.

Structuration des outils

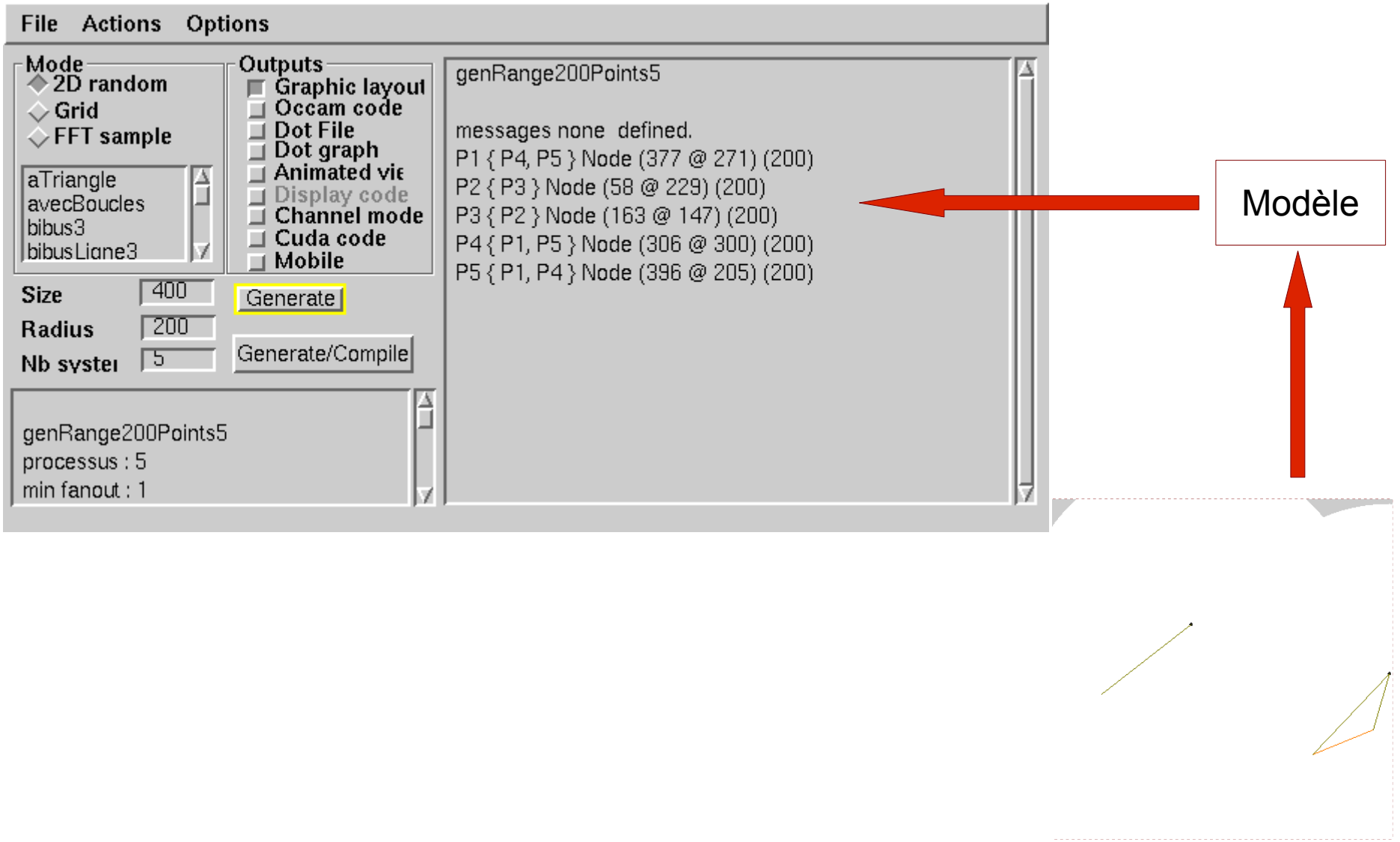
- Outil de développement principal : Smalltalk
  - VisualWorks : prototypage rapide,constructions de modèles. Licence universitaire gratuite
- Pour l'expression et la simulation systèmes
  - Occam, compilateur de U. de Kent, libre
  - CUDA, compilateur et kit nvcc de Nvidia, licence
  - Graphviz, dot, 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 : *Pi { Pj Pk Pn } Programme*

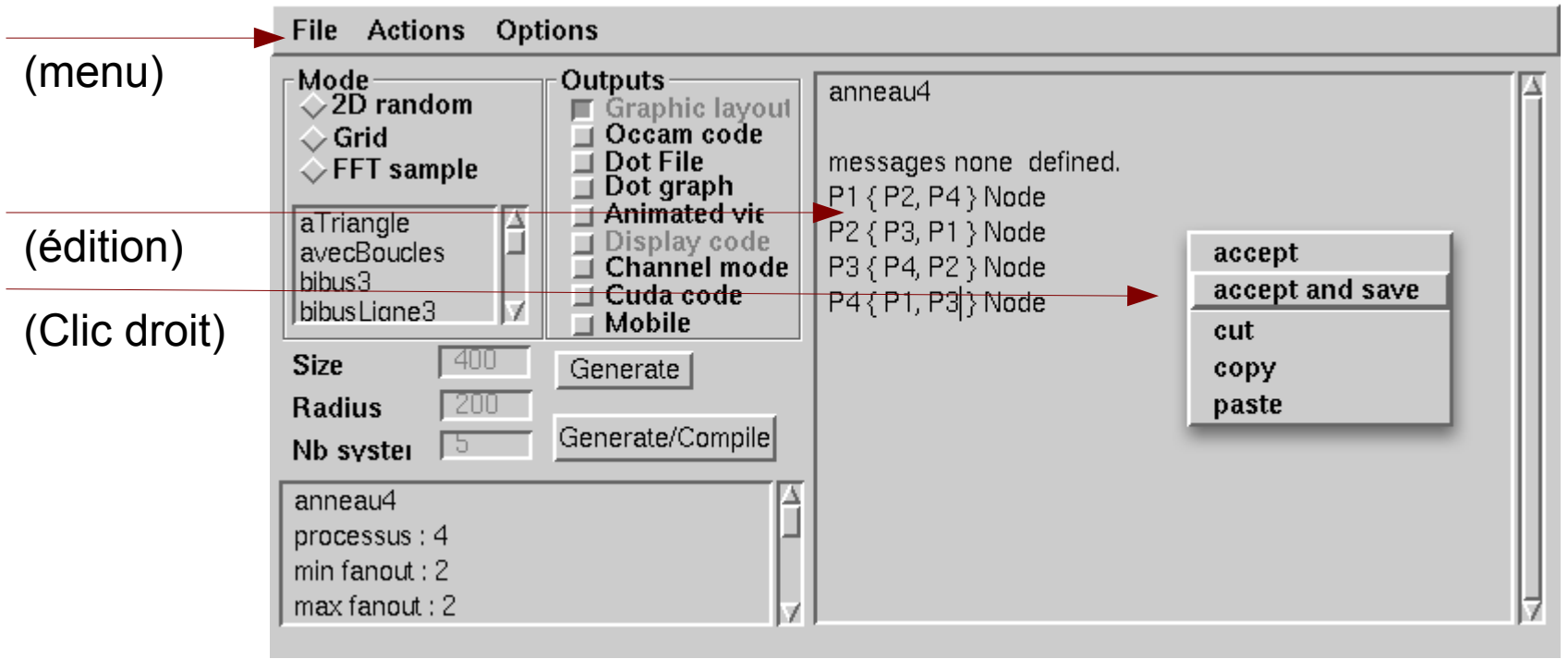


Spécification géométrique (2)



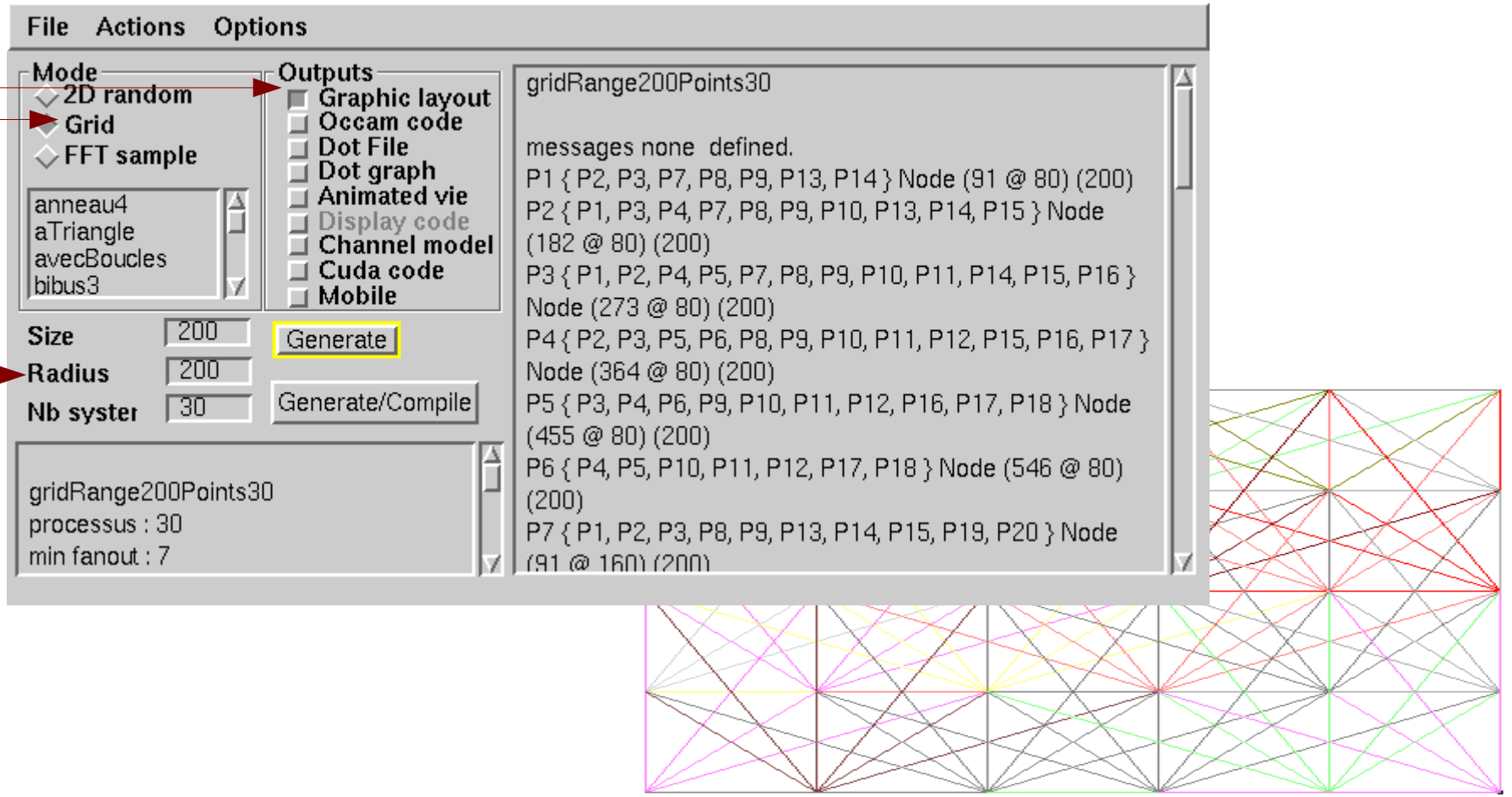
Spécification textuelle / fichiers

- Editeur de spécification
  - Syntaxe spécification : titre messages (*Pi { Pj Pk Pn } Programme) ..*
  - *Load et save :* possibilité d'écrire vos propres générateurs.



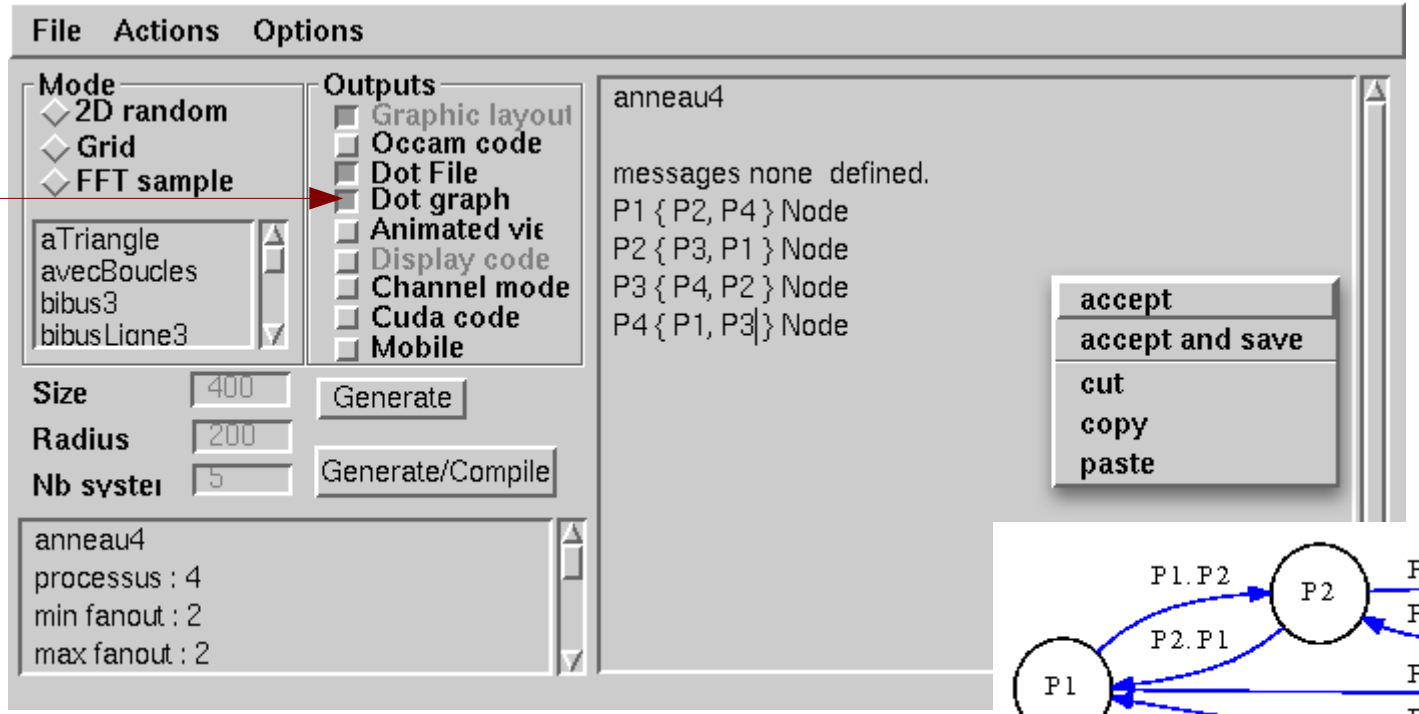
Spécifications régulières

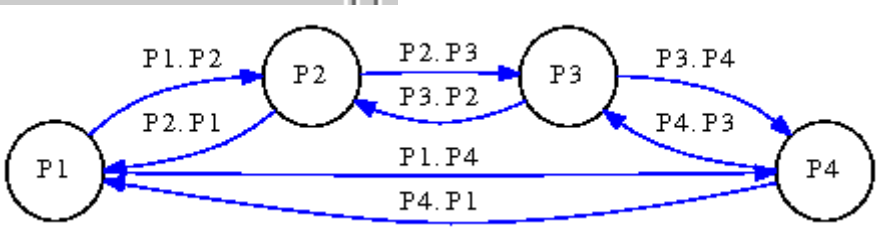
- Grilles de connectivité variable



Présentation graphe logique

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



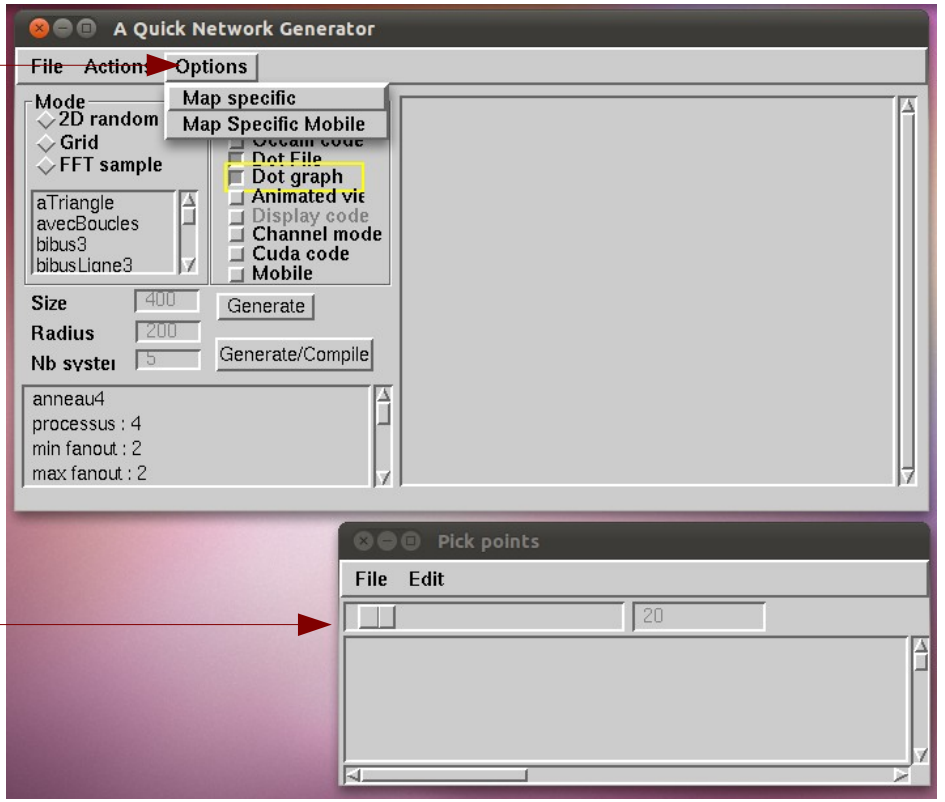


Saisie sur carte (1)

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

Menu option : map

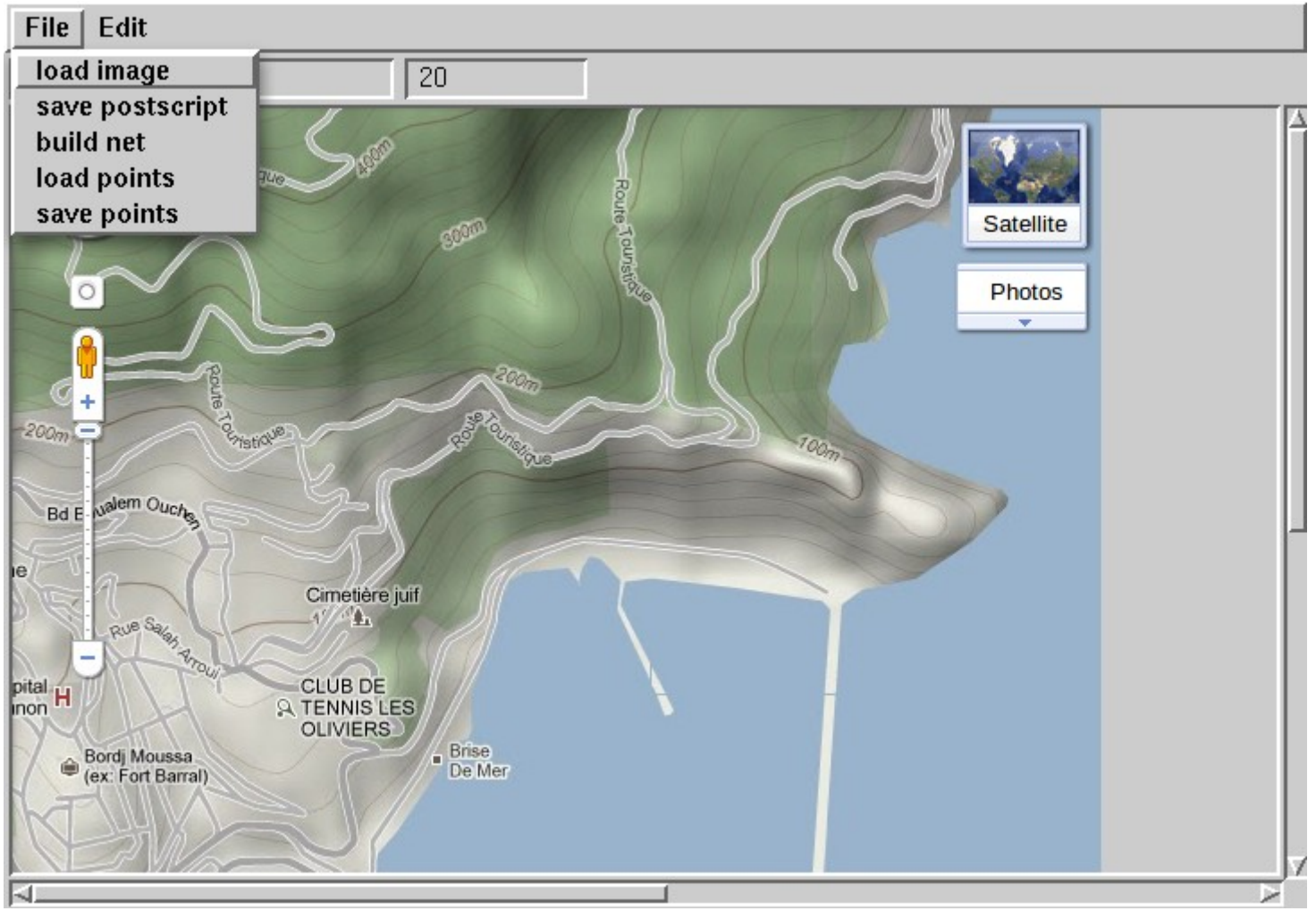
Fenetre graphique



Saisie sur carte (2)

Chargement d'une image (.png

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



Saisie sur carte (3)

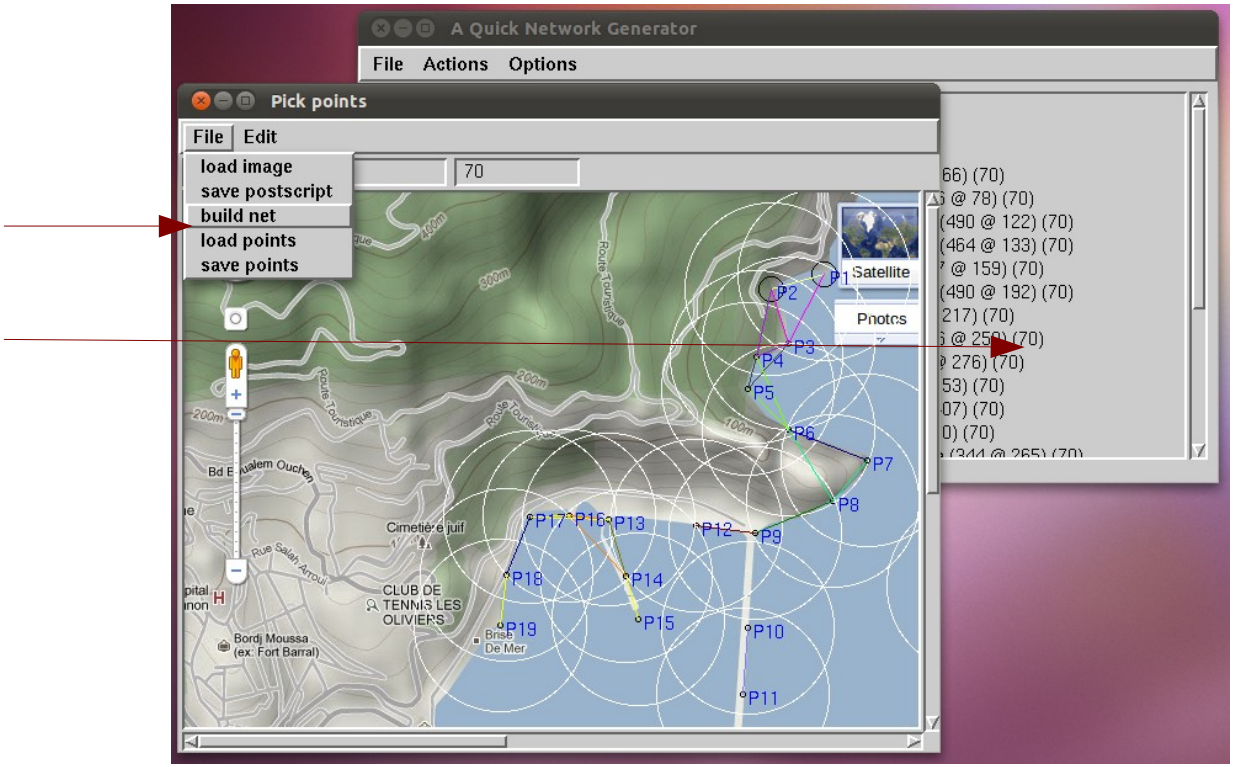
Positions des capteurs

- 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



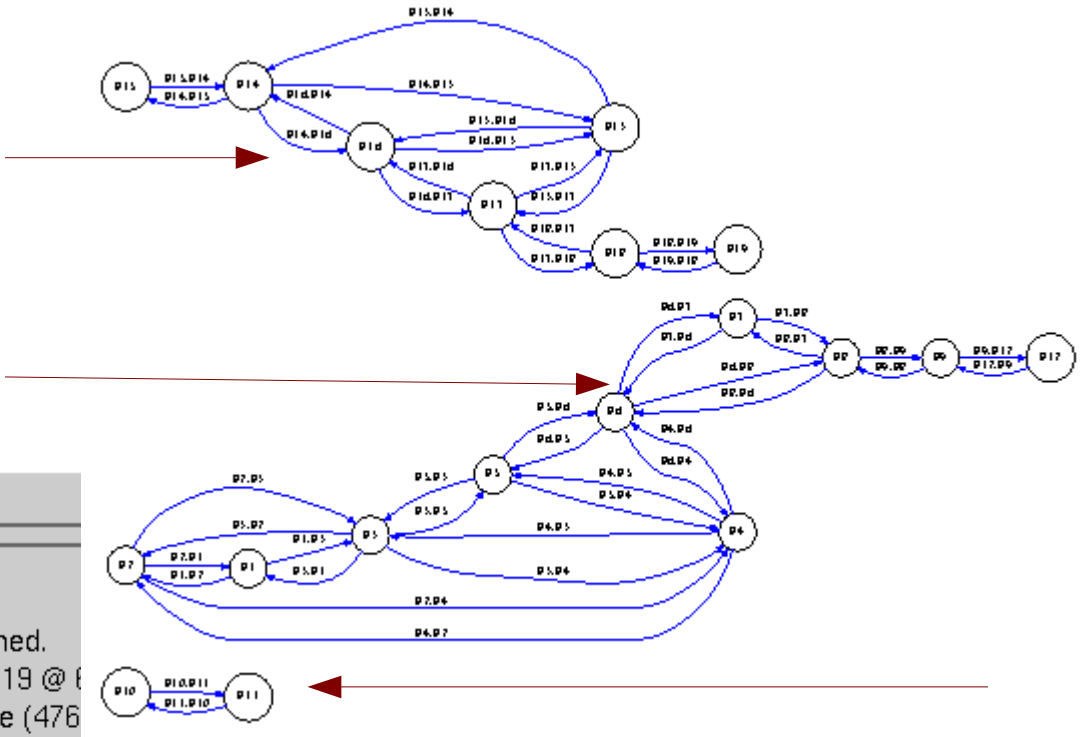
Chemins de mobiles

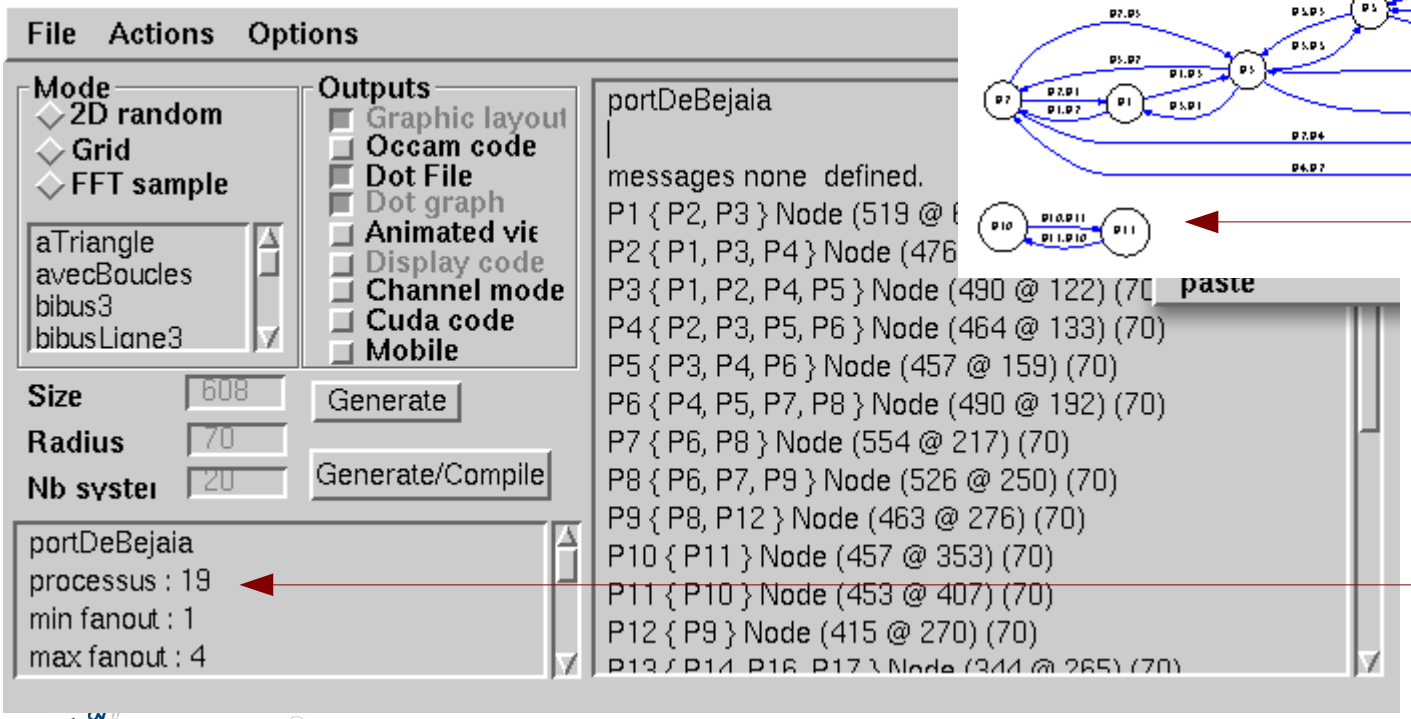
- similaire

Saisie sur carte (5)

Analyse du modèle

- Application des outils sur le modèle produit



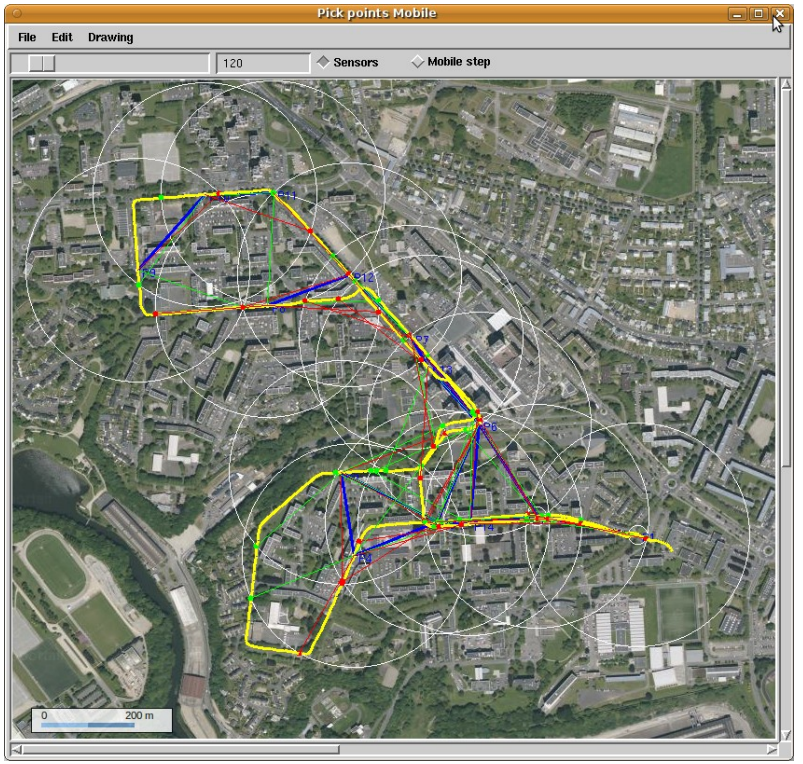


Alternative : saisie par GPS

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

Saisie par GPS : ré-écritures

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

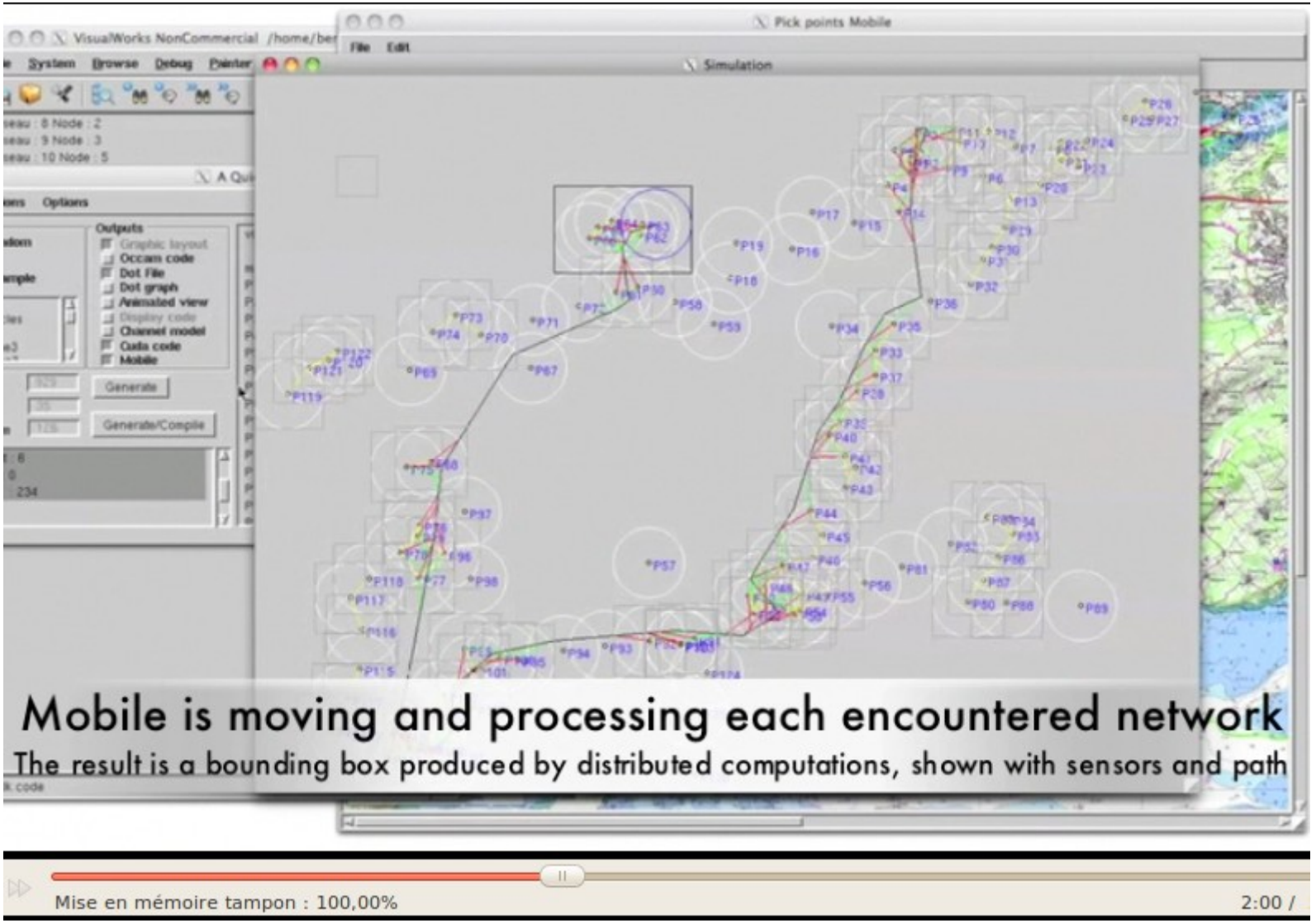


Proposition de flot de conception (rappel)

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



RESSACS 2011, B.Pottier et al. http://wsn.univ-brest.fr/VoyageEnRade

Evolutions

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

Merci .

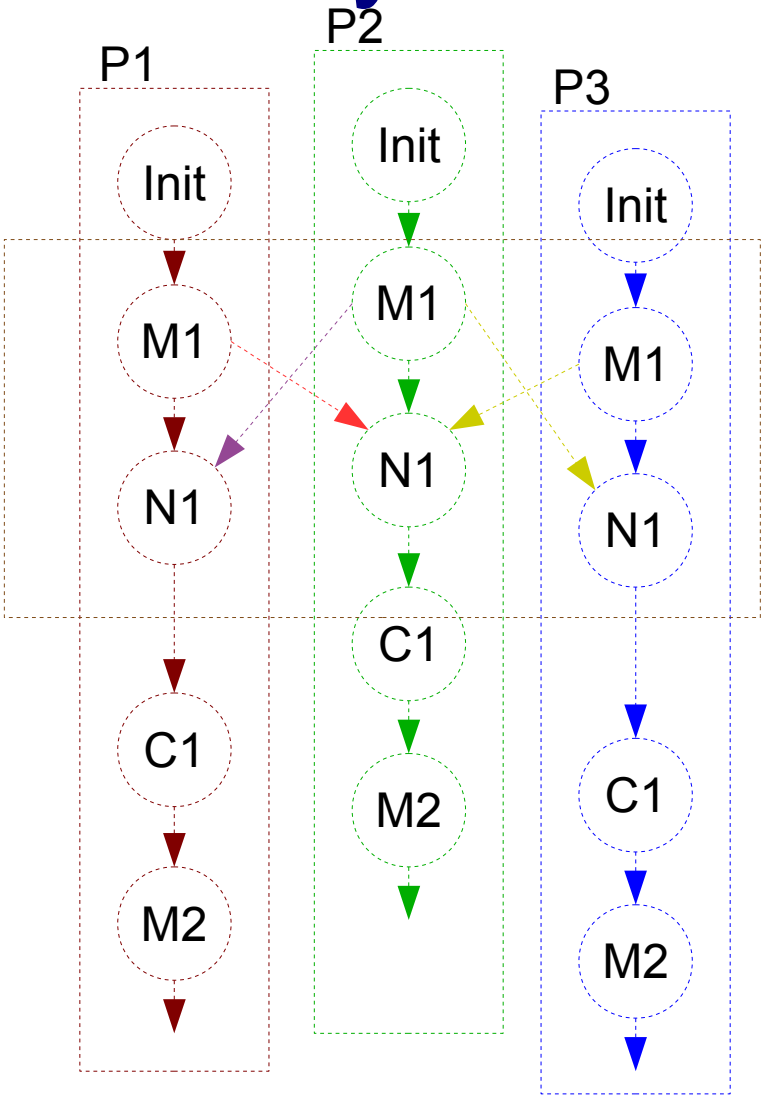
Modèle synchrone : ref. N. Lynch

Comportement abstrait

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

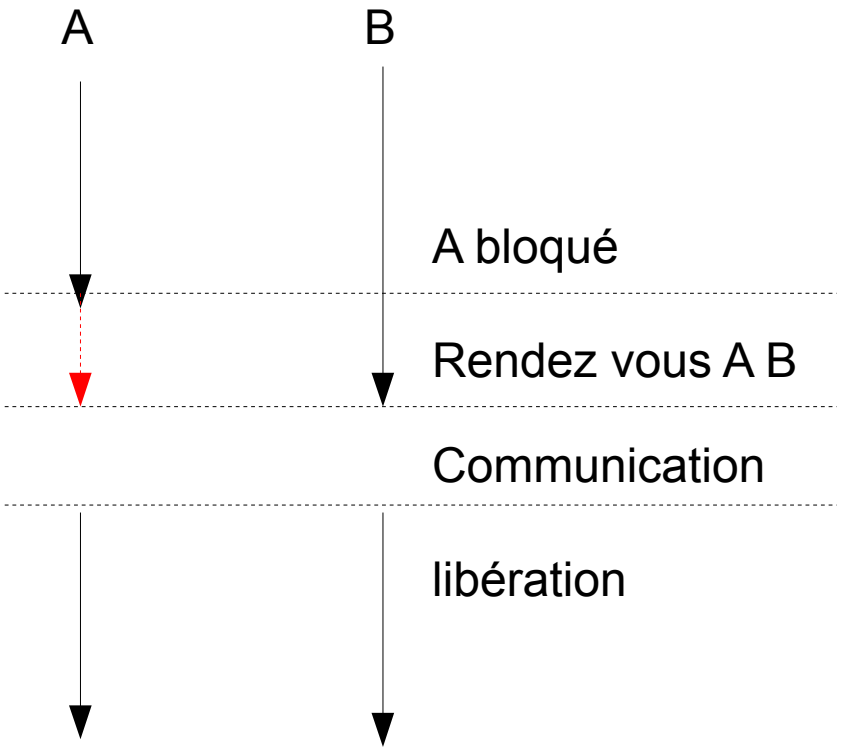
Méthodes d'exécution

- *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 ecrit 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 Mi Ni du modele synchrone

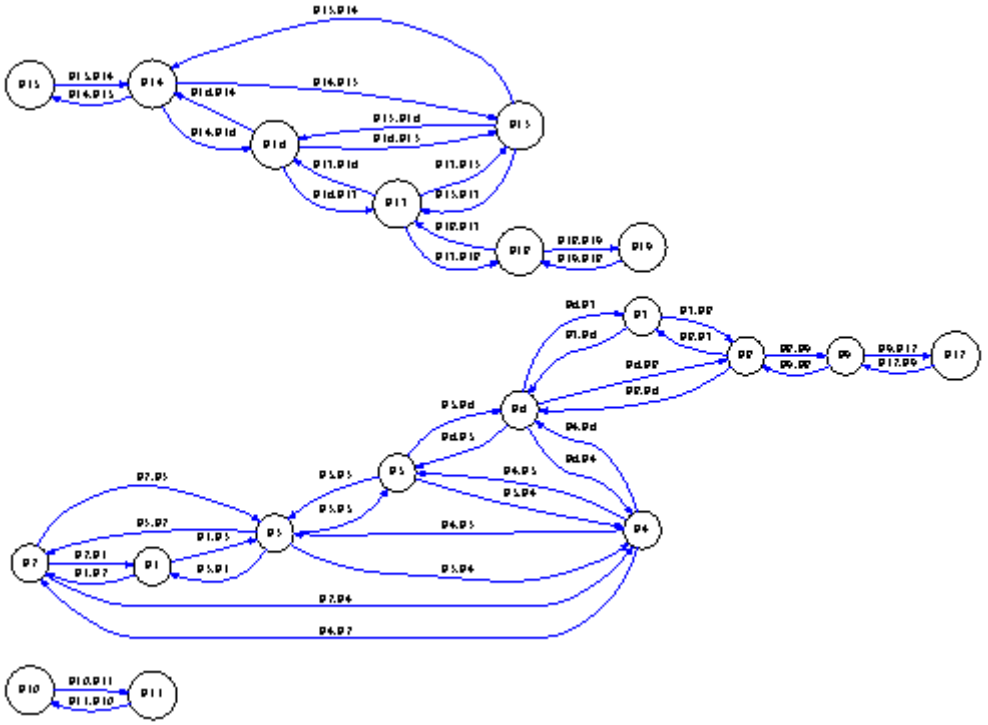
```
PAR
  PAR i=0 FOR SIZE in
    in[i] ? CASE
      table ; inMessages[i]
        SEQ
          tags[i]:=TRUE
          externalChange :=TRUE
      null ; nullByte
        tags[i]:=FALSE

  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 !

Processus communicants : Trace

1ere colonne : Pi
2nde colonne : diametre
3ieme colonne : leader

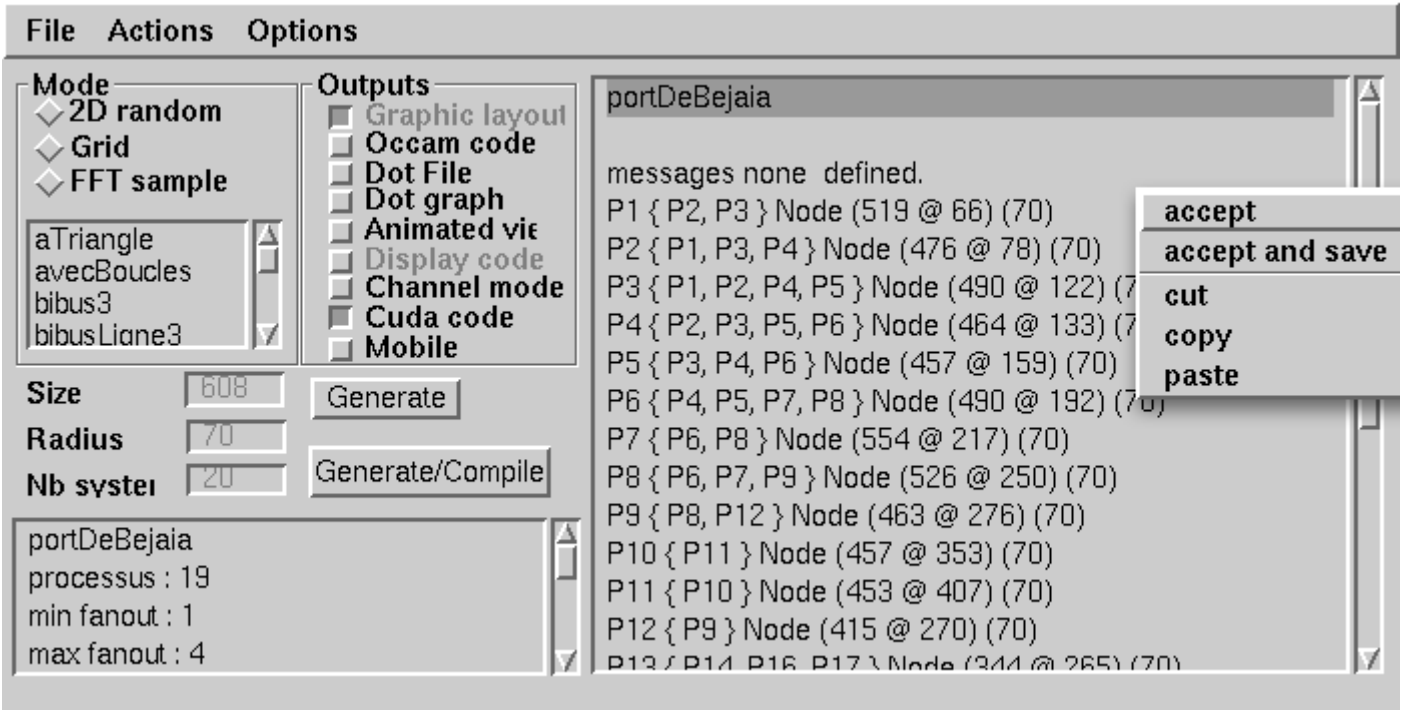


10	1	10
9	1	10
18	5	18
0	6	11
1	6	11
2	6	11
3	6	11
4	6	11
5	6	11
6	6	11
7	6	11
8	6	11
11	6	11
12	5	18
13	5	18
14	5	18
15	5	18
16	5	18
17	5	18

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 nbIn;
	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 controleur Smalltalk pour la visualisation

Merci ..