CONSONA

Constraint Networks for the Synthesis

of Networked Applications

Lambert Meertens
Asuman Sünbül
Yunmei Wei
Stephen Fitzpatrick
http://consona.kestrel.edu/

NEST PI Meeting
Santa Fe, NM, 15–17 December 2003

Lambert Meertens
Cordell Green
Kestrel Institute

Administrative

- Project Title: CONSONA - Constraint Networks for the Synthesis of Networked Applications
- PM: Vijay Raghavan
- PI: Lambert Meertens & Cordell Green
- PI Phone #: 650-493-6871
- PI Email: meertens@kestrel.edu & green@kestrel.edu
- Institution: Kestrel Institute
- Contract #: F30602-01-C-0123
- AO number: L545
- Award start date: 05 Jun 2001
- Award end date: 04 Jun 2004
- Agent name & organization: Roger Dziegiel, Jr, AFRL/Rome

Subcontractors and Collaborators

- Subcontractors
  - none
- Collaborators
  - none

Lambert Meertens / Cordell Green
Kestrel Institute

Consona

Constraint Networks for the Synthesis of Networked Applications

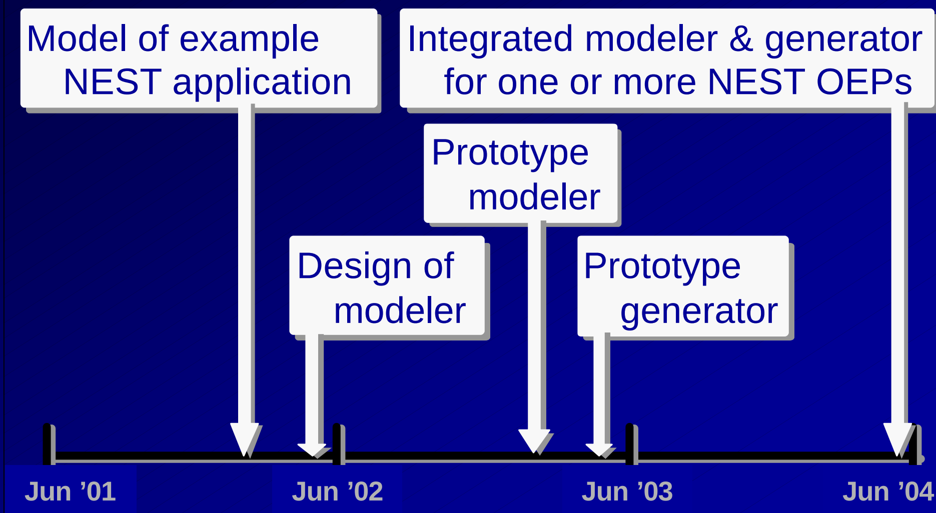
New Ideas

- Model NEST services and applications *uniformly* with *constraint networks*
- Design applications out of components *irectly at the model level*
- Use constraint-propagation t nerate *highly optimized cross-c*

Impact

- Ultra-high *scalability* and unprecedented *lev granularity*
- The technology enables *flexible*, *manageab adaptable* application design at a *mission-orie*
- Generated systems are *robust* (fault tolerant, se stabilizing) with *graceful degradation* on task over

Schedule



Problem Description/Objective

- The Consona project aims at developing *truly scalable* methods for fine-grained fusion of physical and information processes in large ensembles of networked nodes
- Focus on developing local (constraint) optimization algorithms that lead to approximate global optimality
  - –scalable, robust, adaptive, anytime
  - –also developing hierarchical algorithms, but see caution on next slide
- May need to redefine the problem to admit scalable solutions
  - –for realistic physical assumptions

idealized, bird's-eye view

hard, global metric

e.g., global consensus

soft, global metric

e.g., minimize the maximum discrepancy between any two nodes

scalable, worm's-eye view

soft, local metric

e.g., minimize the discrepancies between a node and its neighbors

Problem Description/Objective (cont.)

- Ideally, algorithms should have **bounded** per-node performance and cost metrics as the network size increases
  - –may settle for logarithmic performance and costs
- Need to watch out for hidden (communication) assumptions that cannot hold for **large** networks
  - –e.g., as the number of nodes increases, the geographical size of the network must become large compared with the physical size of a node, so single-hop, node-to-node communication would require unrealistically high power
  - –e.g., for continental or inter-continental (or bigger!) networks, the travel time of a radio signal becomes significant
- Failure of hidden assumptions may raise costs qualitatively
  - –e.g., a bounded communication range means that a message from one end of a network of N nodes to the other end requires a number of hops proportional to N, $\sqrt{N}$ or $\sqrt[3]{N}$ for 1D, 2D or 3D

Project Status

- Localization service: common coordinate system
  - –implemented in nesC
  - –tested on simulator for 100 motes
- Simulator extensions
  - –microphone and sounder
  - –ranging component
  - –magnetometer
  - –visualization for localization

The Localization Problem

- Bird's-eye view: construct a global coordinate system for all motes
  - –every mote agrees with every other mote about the coordinates of every mote
- Worm's-eye view: construct local coordinate systems that vary slowly over space
  - –each mote is to have a map containing itself and some nearby motes
  - –for two nearby motes, the two maps should have several motes in common
  - –for two nearby motes, the coordinates of any mote that is in both maps should be approximately the same in both maps
- Caution: there may not be a scalable solution to this problem
  - –there are algorithms with constant per-node, per-second costs but they may not deliver constant performance as the number of nodes increases
  - –an alternative formulation, involving coordinate transformations between interacting motes, may have scalable solutions

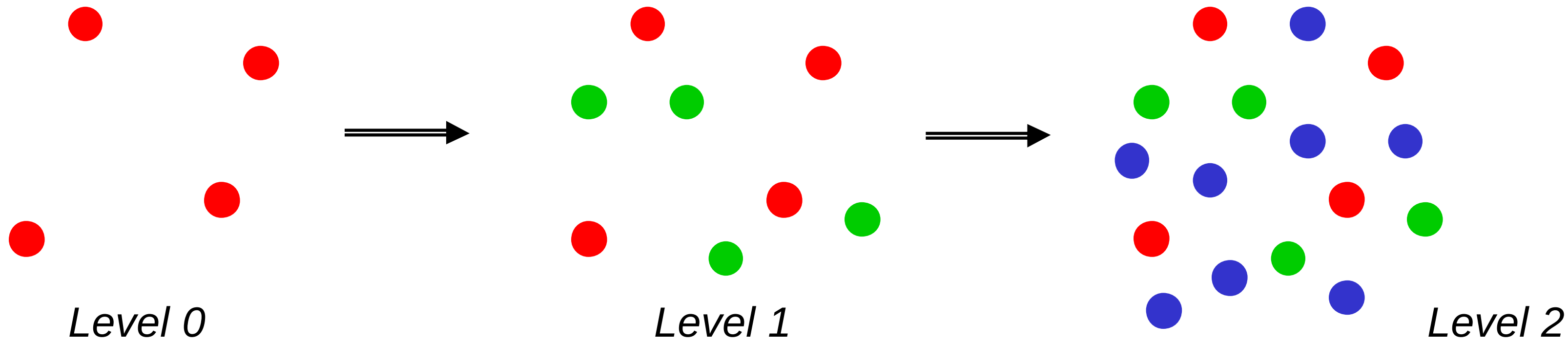
The Single-Level Localization Algorithm

1. Use VU ranging service to determine distance from each mote to some neighbors (i.e., nearby motes)
2. Exchange distance information with neighbors so that each mote also knows neighbor-to-neighbor distances
3. Use neighbor-to-neighbor distances to compute a local map for each mote independently
4. Repeatedly exchange maps with neighbors to allow motes to reconcile their own maps with their neighbors' maps
   - using translations, rotations, reflections and averaging
   - use a priority-ripple technique to speed up convergence (doesn't truly scale)

- Example of sense-fuse-disseminate idiom
  - minimizes discrepancy between mote's "world view" and local sensory information
  - minimizes discrepancies between neighbors' world views

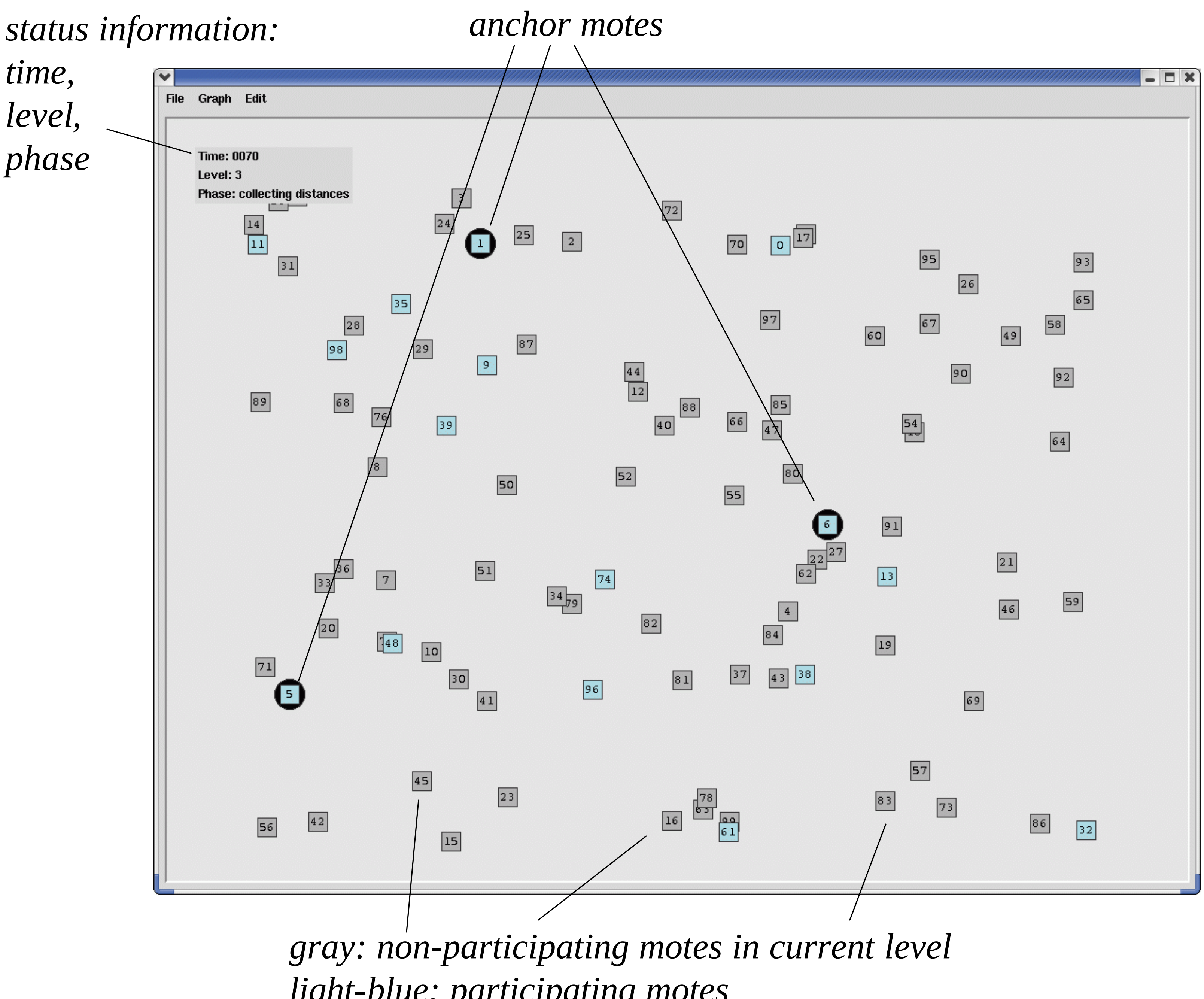
The Multi-Level Localization Algorithm

- Assign motes to levels 0, 1, …, k
  - –based on randomized mote ids to (probabilistically) ensure that levels are spread evenly throughout network
  - –if mote is in level n, it is also in level (n+1)
  - –double the number of motes for each successive level
- Each level takes a turn at single-level localization
  - –level k generates coarse grain maps
  - –successive levels add finer and finer details
- This algorithm should scale (logarithmically)
  - –if communication between any pair of motes has bounded costs regardless of the number of motes

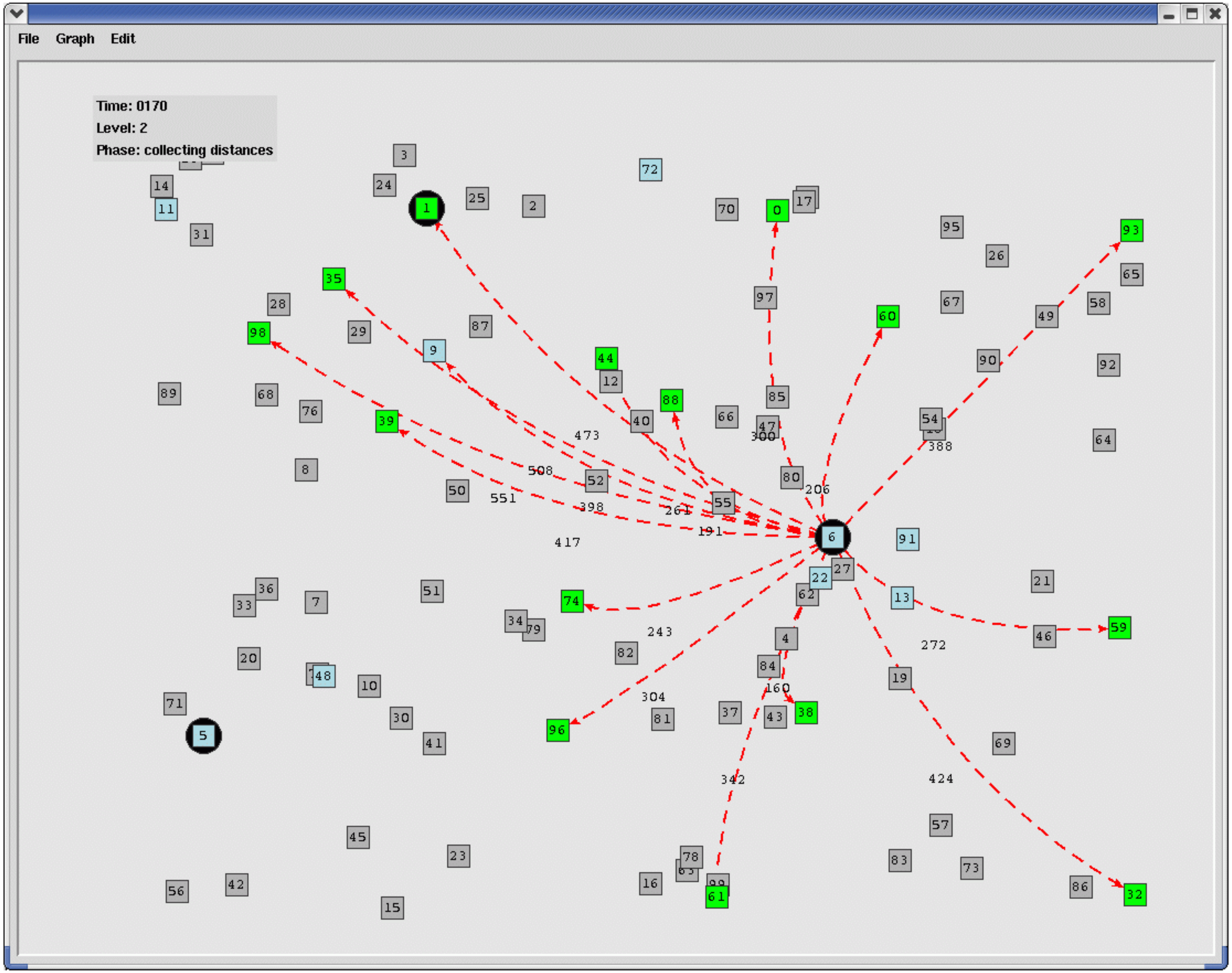


Status of Implementation

- Previously showed a centralized Java simulation of the algorithm
- Algorithm has now been implemented in nesC and tested in the TinyOS simulator
- Customized light-weight Tcl/Tk based GUI
  - –uses color codes to visualize application specific states of motes and message types
  - –interactively visualizes estimation result after termination of the algorithm

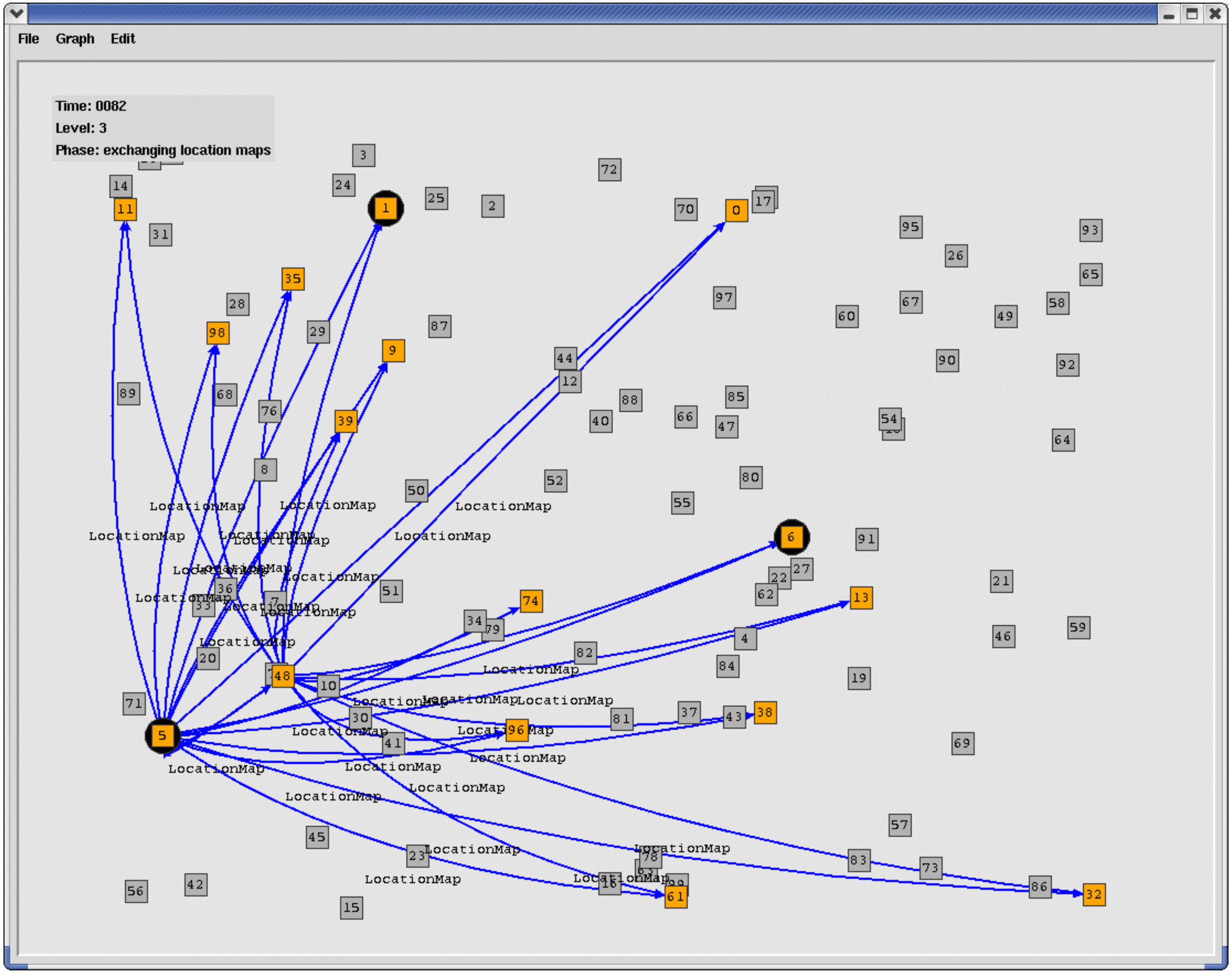

status information:
time,
level,
phase
anchor motes
File
Graph
Edit
Time: 0070
Level: 3
Phase: collecting distances
gray: non-participating motes in current level
light-blue: participating motes

visualization of radio messages, different colors for different kind of data



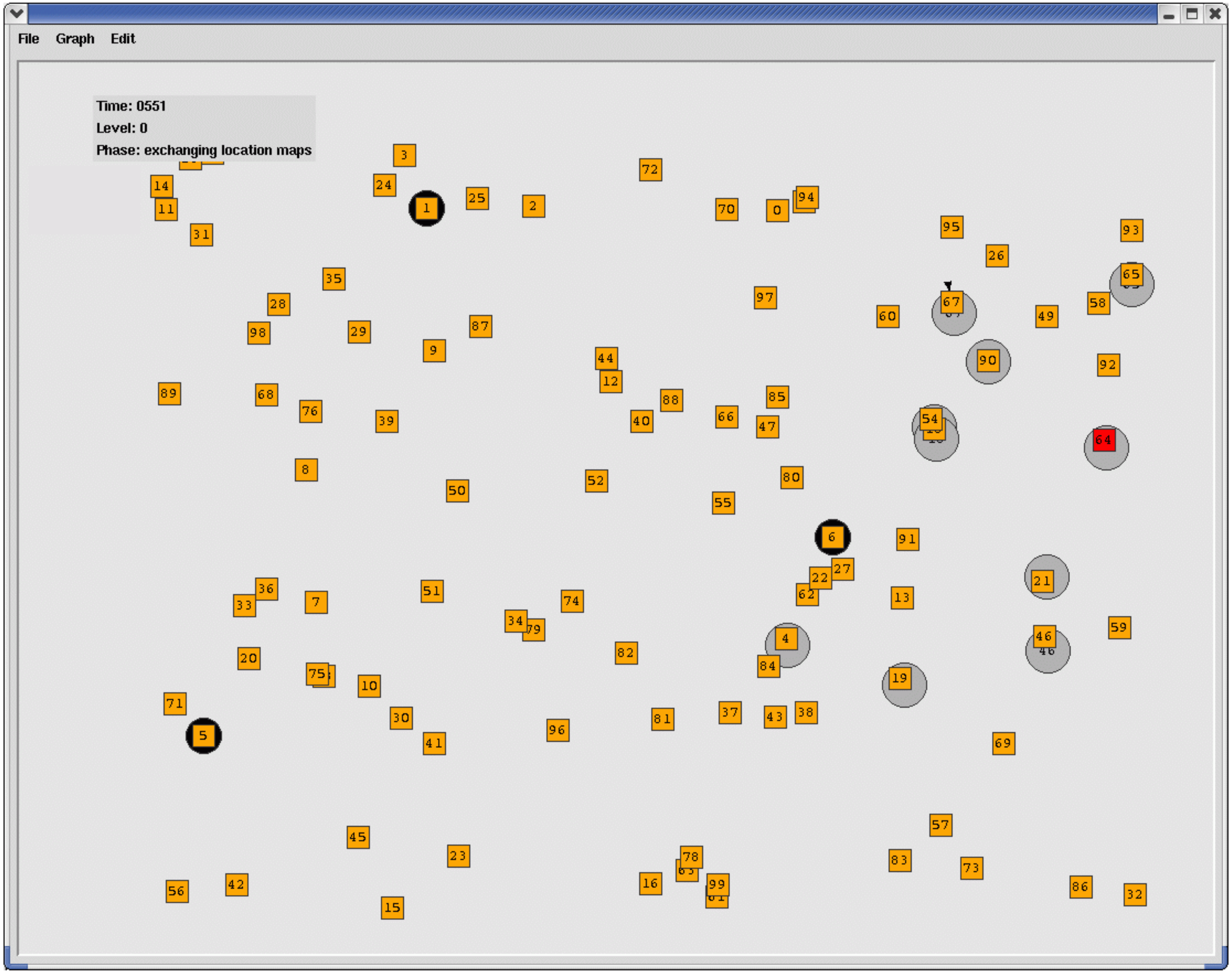
red arrows: distances

green: motes receiving beacon messages *visualization of radio messages, different colors for different kind of data*

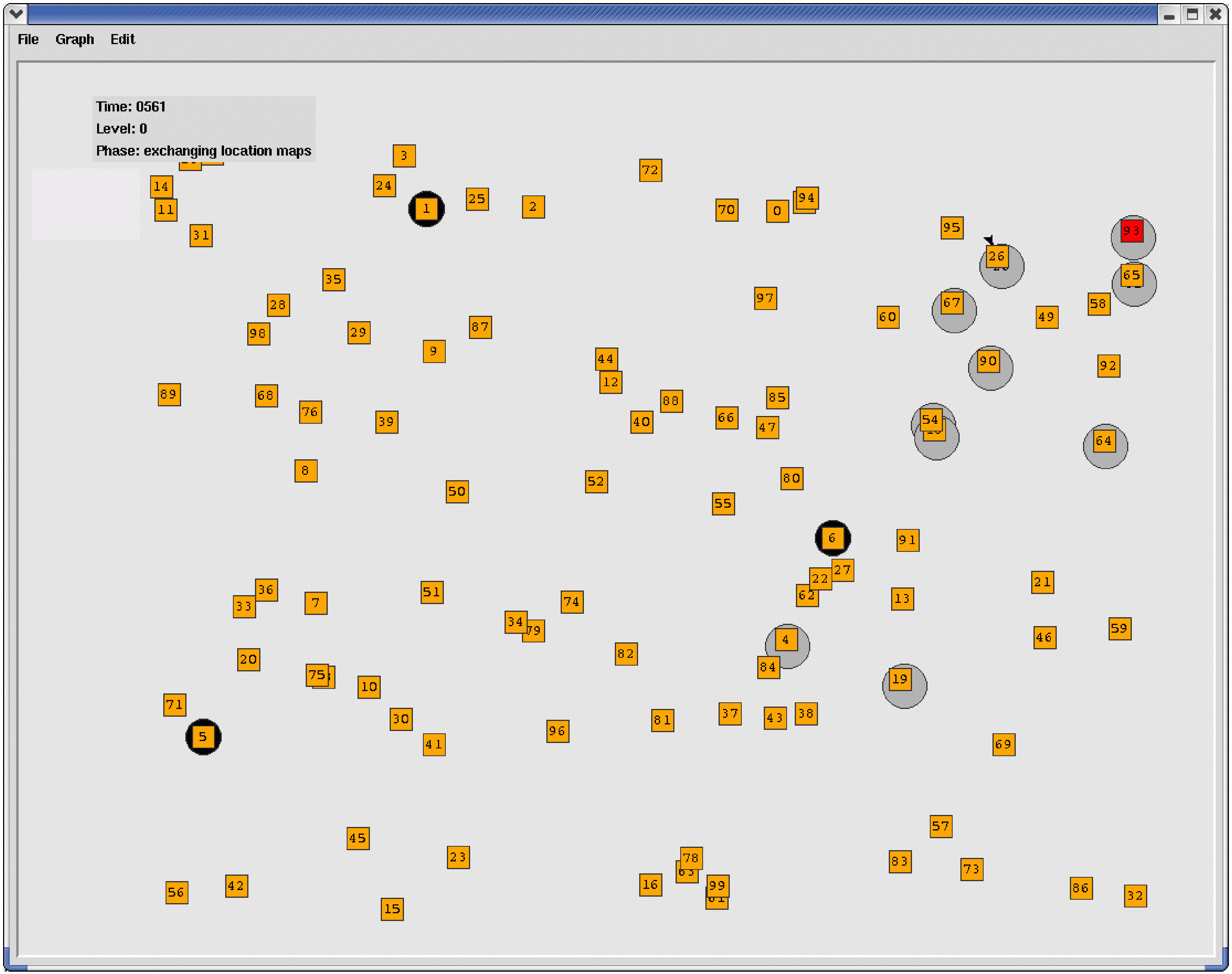


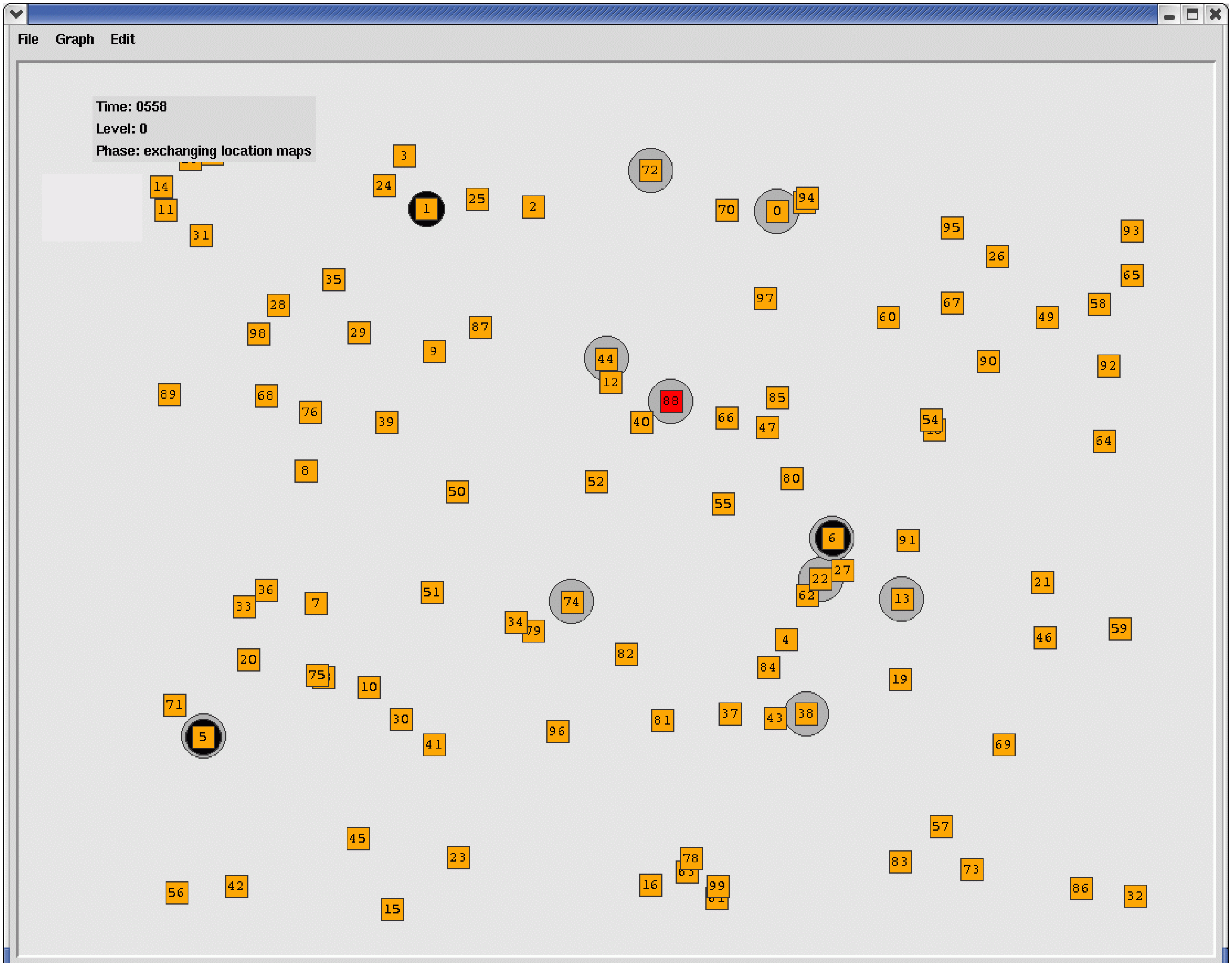
blue arrows: location maps

orange: motes that have finished the calculations for the current level



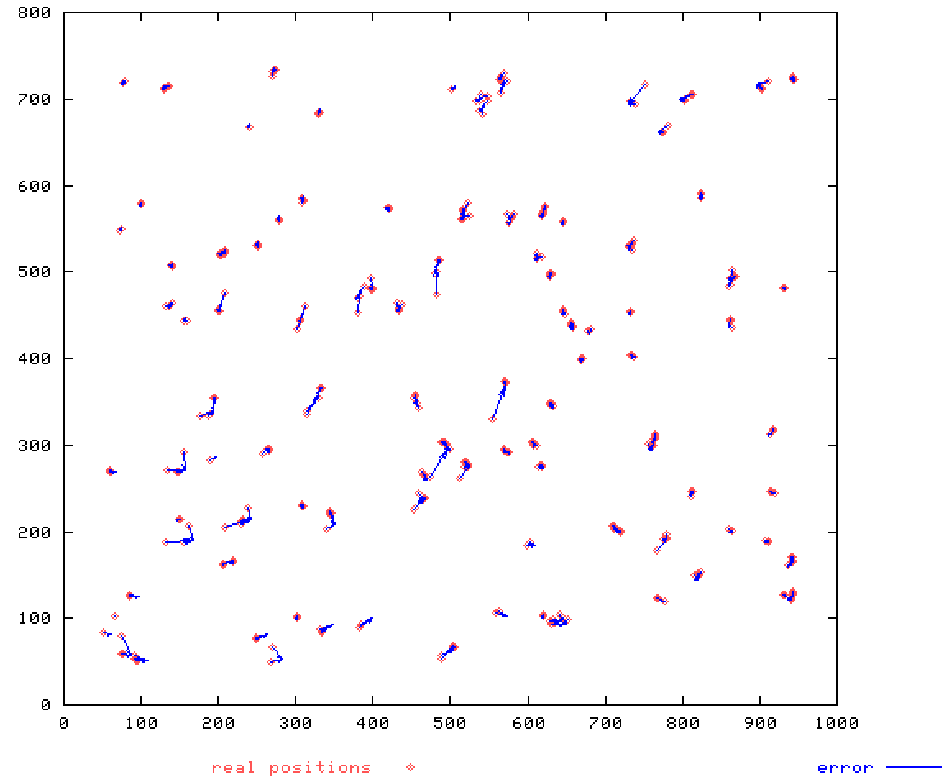
- *The estimated positions are visualized interactively after termination of the localization by moving the mouse point over a mote (here: 64)*
- *The gray circles show the resulting calculated position for the given neighbor mote.*


File
Graph
Edit
Time: 0561
Level: 0
Phase: exchanging location maps


File
Graph
Edit
Time: 0558
Level: 0
Phase: exchanging location maps

Preliminary Results

- 100 motes
  - –average separation 44cm
- simulated ranging
  - –no noise
- average error 5.4cm
- median error 4.1cm
- Still to do comprehensive tests



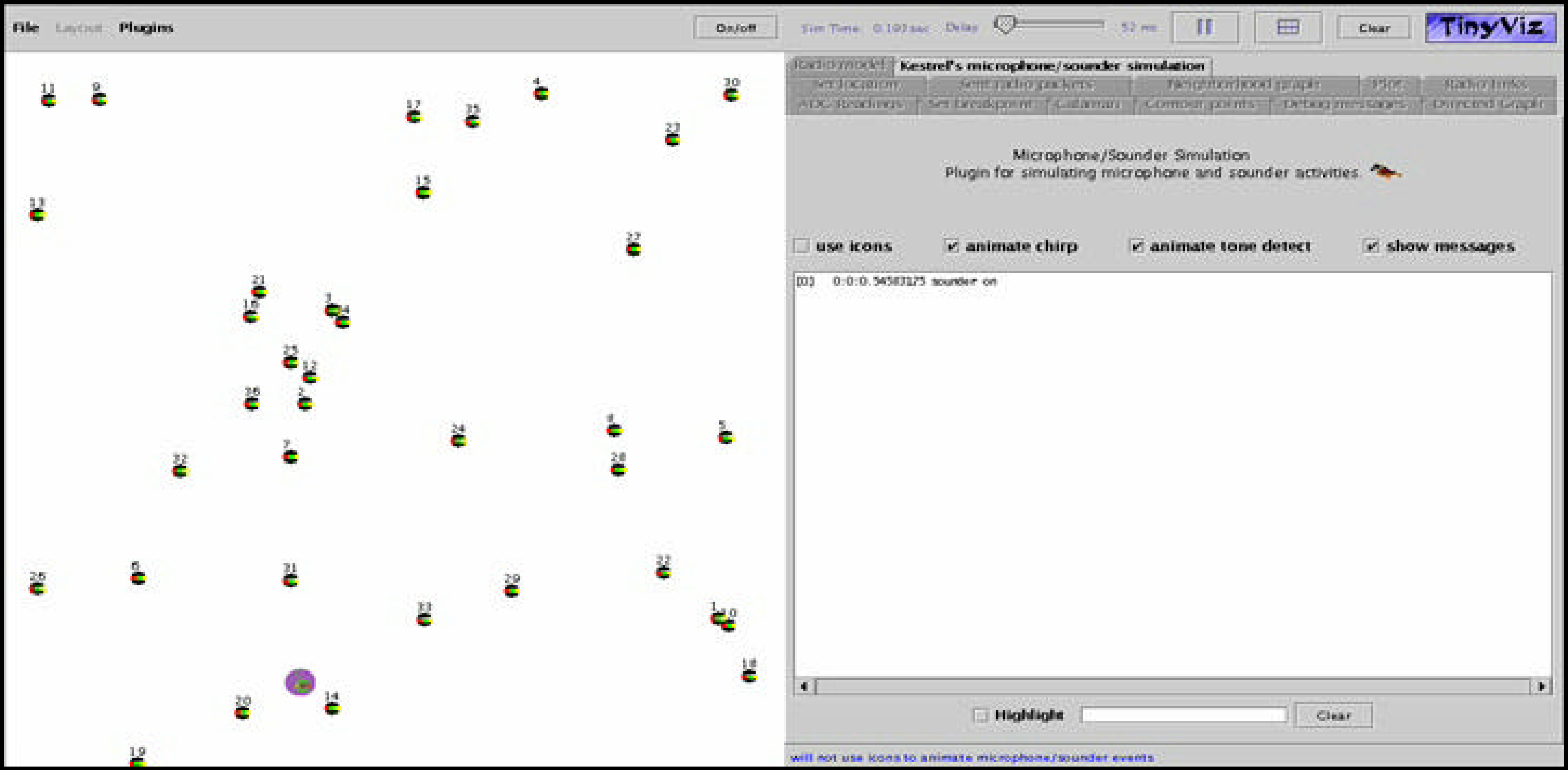
- Major impediment: it seems that in the simulator, transmissions by any two motes interfere, regardless of separation
  - –motes wait forever for clear-to-send
  - –makes simulation for large numbers of motes virtually impossible

Kestrel’s Simulator Extensions

- Based on Berkeley’s Nido
- Simulation of microphone and sounder
  - not used in experiments
  - physical model for sensitivity, delays etc. not validated
- Simulation of the VU Acoustic Ranging component
  - used for Kestrel’s work on localization
  - validated for MICA2 by similar results for TestAcousticRanging as on hardware motes
- Simulation of magnetometer
  - used for simulation of ‘MagTrackingDemo’ application

Microphone/Sounder Simulation

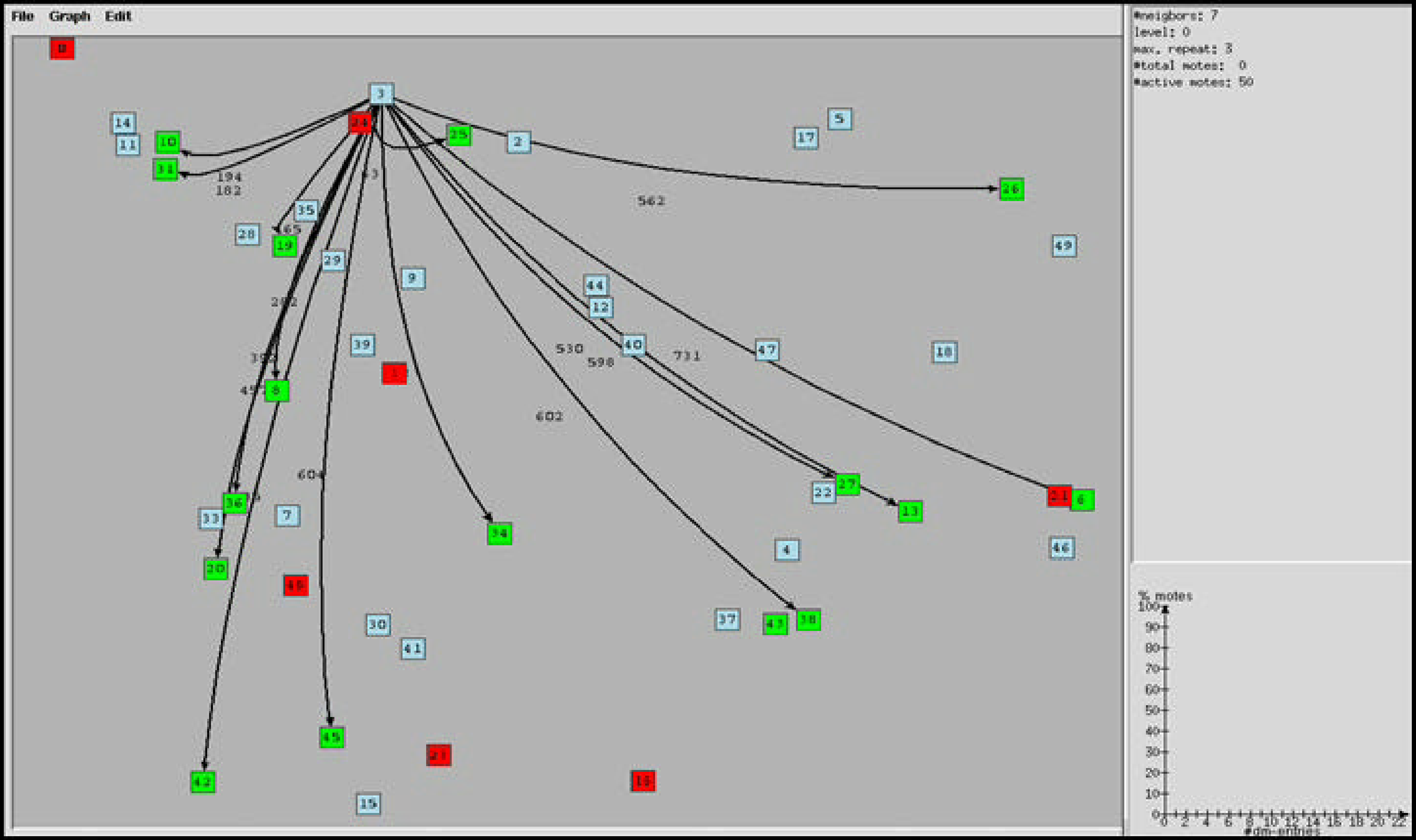
- Kestrel has extended the graphical interface to the Nido simulator (TinyViz) to visualize and animate microphone and sounder events
  - communicates with the SMSIM extension to the TinyOS simulator using the plug-in mechanism provided by the TinyViz framework
- The microphone/sounder plug-in provides the following functionality:
  - assignment of mote locations
  - animation of microphone and sounder events
- Quick start documentation/installation guide available at consona.kestrel.edu
  - implementation is explained by a walk-through scenario for a given mote emitting a sound at a given time
  - documentation describes how tone detection events are created in the neighboring motes



File
Layout
Plugins
On/off
Sim Time
Delay
Clear
TinyViz
Kestrel's microphone/sounder simulation
Microphone/Sounder Simulation
Plugin for simulating microphone and sounder activities
use icons
animate chirp
animate tone detect
show messages
sounder on
Highlight
Clear
will not use icons to animate microphone/sounder events

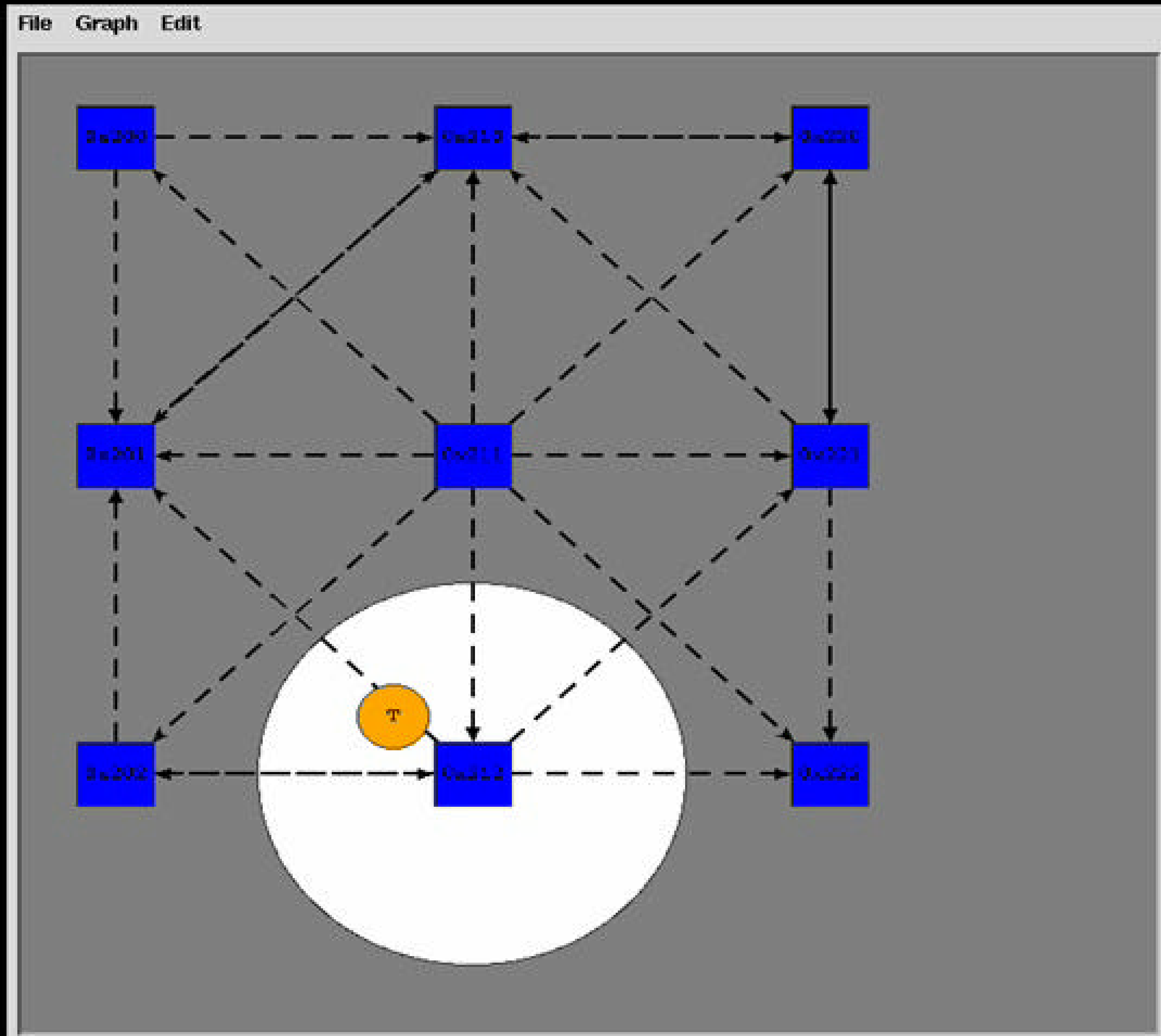
Simulation of VU Acoustic Ranging

- The VU Ranging component is used to determine distances between motes by using the difference between time-of-flight for a radio signal and a sound signal
- The motes emit once every minute a radio signal and a chirping sound at the same time, so that receiving motes can estimate the distance to the sender from the time elapsed between the times of arrival of the two signals
- We do not simulate the actual algorithm; instead, the simulated component provides the same interface and similar behavior to its ‘users’


File
Graph
Edit
#neigbors: 7
level: 0
max. repeat: 3
#total motes: 0
#active motes: 50
% motes
#dm-entries

Simulation of MagTrackingDemo

- Extension of Nido for simulating the MagTrackingDemo application
- Magnetometer sensor simulation
  - analyzed real sensor values
  - developed a model of a magnetometer
  - implemented a replacement component for the magnetometer in **`platforms/pc`**
- Documentation and implementation guide at consona.kestrel.edu


File
Graph
Edit
T

Goals and Success Criteria

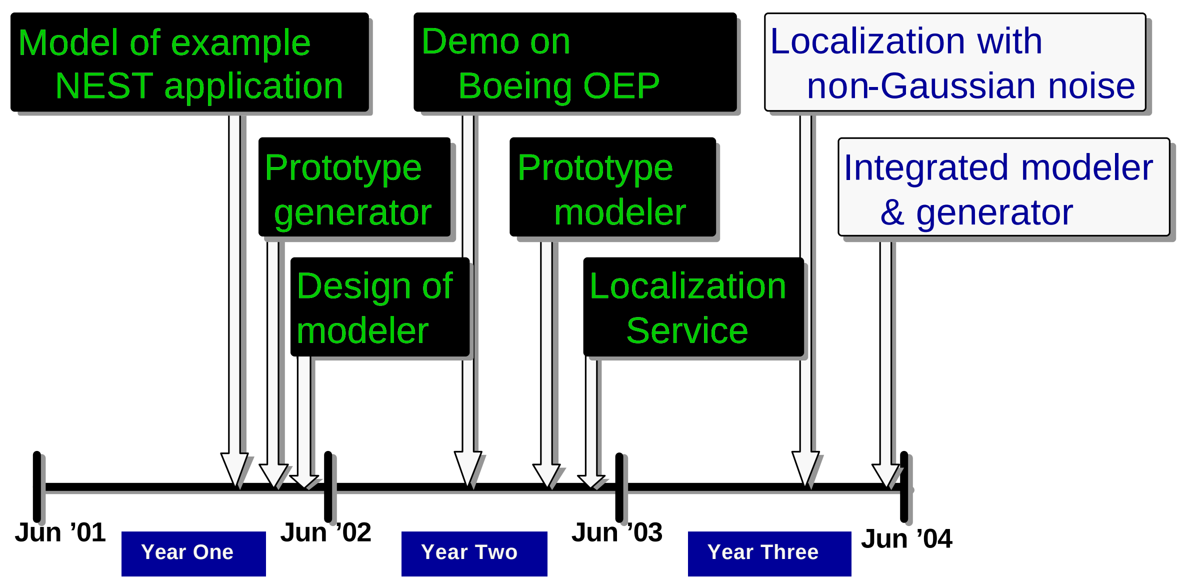
- Localization service
  - scalability: rate of increase in computational, storage and communication costs with number of motes
  - accuracy: error in position estimates as a function of error in distance estimates

Project Plans

- Investigate how localization service can be adapted to non-Gaussian noise in distance estimates
  - e.g., extend outlier rejection methods
  - e.g., use multi-hypothesis techniques for initial map construction by individual motes (before inter-mote reconciliation)
- Test localization service on hardware

Project Schedule and Milestones

- Modeling using constraints: achieved
- Toolset: preliminary design – done, informal
- Prototype modeling toolset: done
- Sensor extensions to mote simulator: acoustic, magnetic
- Localization service: designed and implemented
- Spring 2004: Handling of non-Gaussian noise in localization service
- June 2004: Integrated modeler & generator



Technology Transition/Transfer

- Technology transition activities identified: currently none

Program Issues

- It is important that the simulator support **large** networks
  - –would like to try algorithms with hundreds or thousands of nodes