CONSONA

Constraint Networks for the Synthesis of Networked Applications

Lambert Meertens
Cordell Green
Asuman Sünbül
Stephen Fitzpatrick

Kestrel Institute

Palo Alto, California

http://consona.kestrel.edu/

NEST PI Meeting, Napa, CA, February 6–8, 2002

Administrative

- Project Title: CONSONA — Constraint Networks for the Synthesis of Networked Applications
- Program Manager: Vijay Raghavan, DARPA/ITO
- PI: Lambert Meertens 650-493-6871 lambert@kestrel.edu
- Co-PI: Cordell Green 650-493-6871 green@kestrel.edu
- Company/Institution: Kestrel Institute
- Contract Number: F30602-01-2-0123
- AO Number: L545
- Award Start Date: 05 Jun 2001
- Award End Date: 04 Jun 2003
- Agent Name and Organization: Juan Carbonell, AFRL/IFSC

Past Collaboration

NEST Wireless OEP Exercise

- Partners:
  - University of California, Berkeley
  - University of California, Los Angeles
  - University of California, Irvine
- Goal: to see (in a dry run) how
  - Application-Independent Coordination Services
  - Time-Bounded Synthesis and
  - Service Composition and Adaptation

  come together in a non-trivial example application

Future Collaboration

- Another OEP minitask?
- With projects with commonalities
  - Parc?
  - MIT?
  - University of Virginia?

  (to “pool” efforts and results)
- With “complementary” projects
  - Austin/Iowa?
  - . . . ?

  (to use / try out each other’s results)

CONSONA: Constraint Networks for the Synthesis of Networked Applications

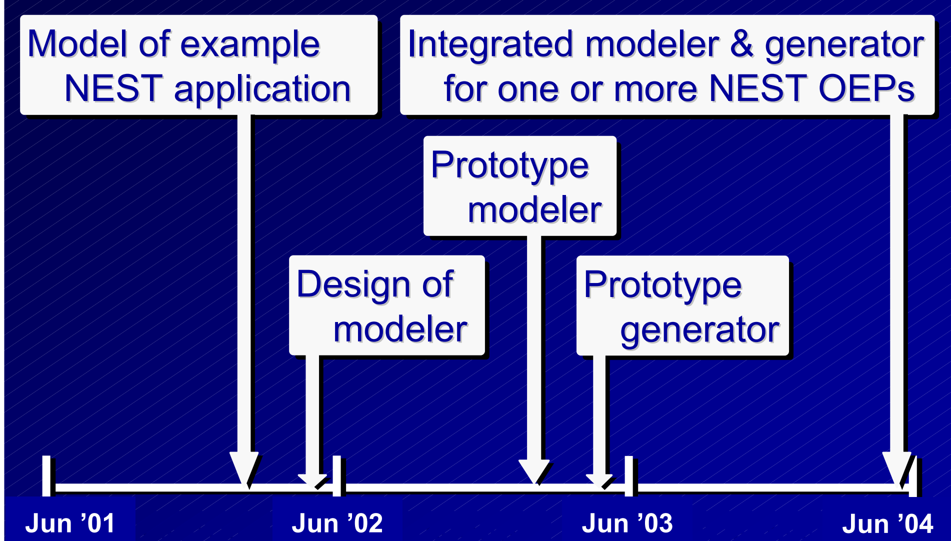
New Ideas

- Model NEST services and applications *uniformly* with *constraint networks*
- Design applications out of components *directly at the model level*
- Use constraint-propagation technology to generate *highly optimized cross-cutting code*

Impact

- Ultra-high *scalability* and unprecedented *level of granularity*
- The technology enables *flexible*, *manageable* and *adaptable* application design at a *mission-oriented level*
- Generated systems are *robust* (fault tolerant, self-stabilizing) with *graceful degradation* on task overload

Schedule



Kestrel Institute: Lambert Meertens, Cordell Green

Aim of the CONSONA project

- Develop model-based methods and tools that
  - integrate design and code generation
    - ⇒ *design-time performance trade-offs*
  - in a goal-oriented way
    - ⇒ *goal-oriented run-time performance trade-offs*
  - of, simultaneously, NEST applications and services
    - ⇒ *low composition overhead*
- Measures of success:
  - Flexibility of combining components
  - Dynamic adaptivity
  - Run-time efficiency
  - Correctness & maintainability of generated applications

Technical Approach

- Both services and applications are modeled as sets of *soft constraints*, to be maintained at run-time
- High-level code is produced by repeated instantiation of *constraint-maintenance schemas*

  Constraint-maintenance schemas are represented as triples (C, M, S), meaning that

  - constraint C can be maintained by
  - running code M,
  - provided that ancillary constraints S are maintained
- High-level code is optimized to generate efficient low-level code

FAQ

- Why *soft* constraints?
  - Complete constraint satisfaction is typically not feasible under real-time constraints in NEST networks
    - “conventional” requirements are naïve: communication, failures
  - Constraint optimization is feasible
    - quality improves with time available
    - quantitatively trade-off quality against costs incurred
- Where do these constraint-maintenance schemas come from?
  - From libraries of distributed-programming paradigms and patterns extracted from middleware services
  - From (possibly ad-hoc) libraries of application-specific computations
- Is this an automated process?
  - Schema selection is “manual” (interactive or scripted)
  - Most of the rest is automatic

Progress

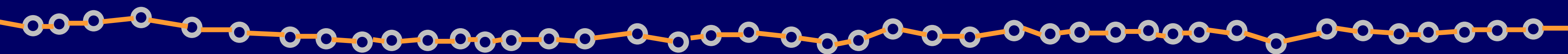
- Assessed conventional distributed algorithms with respect to scalability in large NEST networks
- Looked at diffusion as unifying distributed computing paradigm for NEST networks
- Developed prototype tool for modeling dynamics of distributed algorithms in NEST networks
- Modeled semi-realistic NEST application
  - mini-task collaboration with UCB, UCI & UCLA
- Expressed conventional distributed algorithm in terms of constraint maintenance

Scalability in NEST Networks

- We have looked at published distributed algorithms and protocols in order to model them in terms of constraint-maintenance schemas
- What we found is that surprisingly many are *not scalable* under realistic models of wireless communication.
  - Exception: Amorphous Computing group @ MIT
- Core of the problem is limited information flow
  - E.g., computing the mean of a distributed data field by each node repeatedly replacing its value with the mean of its and its neighbors’ value has extremely poor convergence for large networks
- Scalable applications have *locally expressible* optimality metrics!
  - E.g., smoothing achieves local (approximate) agreement

Scalability … (continued)

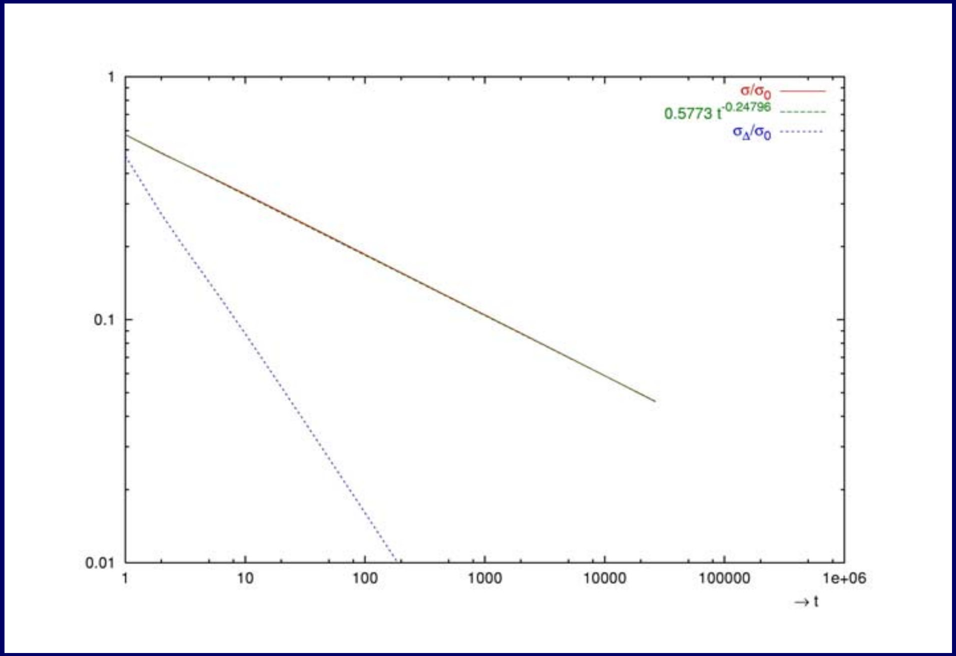
- Experiment: linear string of nodes



- Initially nodes have independent Gaussian random values
- Repeat: replace node value by its and its two neighbors' mean (systolically)

σ goes to 0, but very slowly: not yet at 1% of σ_0 after 1,000,000 steps

σ_Δ goes to 0 much more quickly



Scalability … (continued)

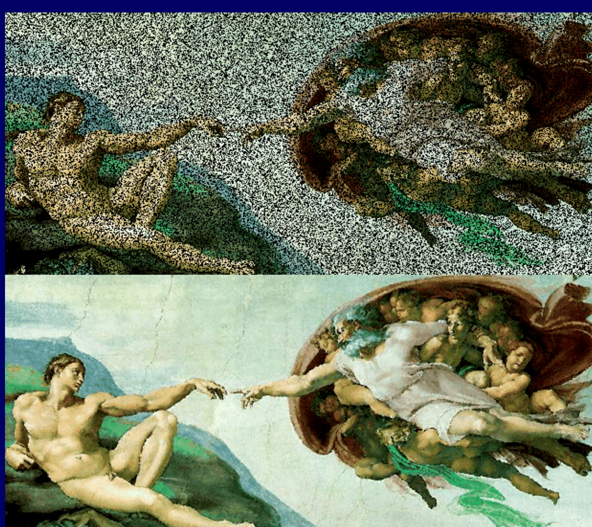
- Structuring wireless networks into layers:
  - 1 in 10 nodes has broadcast area 10 times normal
  - 1 in 100 nodes has area 100 times normal, etcetera
  - Stronger broadcasters form higher layers through which summaries of lower layers' state pass
  - May sometimes help, but effect is limited:
    - "Recurrent Ultracomputers are not log *N*-fast" (Meertens, *Ultracomputer Note #2*, NYU, 1979)
    - "Multiprocessor Architectures and Physical Law" (Vitányi, *Proc. 2nd IEEE Workshop on Physics and Computation,* 1994)
  - Most NEST problems need considerable relaxation of the requirements before they can be scalably solved

Scale-insensitive Performance Metrics

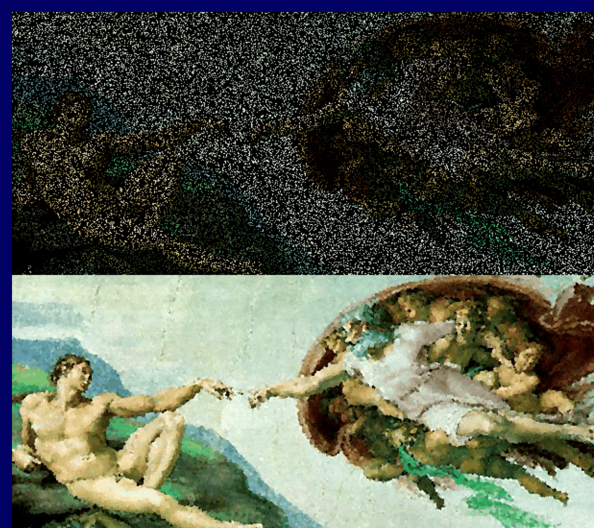
- A NEST application is scalable when its performance does not deteriorate as network size increases
- While seemingly obvious, this definition only leads to a meaningful concept of scalability if an appropriate *scale-insensitive* notion of performance is used
- An example of a scale-insensitive performance metric is: the average performance over all nodes (assuming a reasonable performance metric for individual nodes)
- Another example: average performance over all edges
- In *quasi-scalability*, performance does decrease with network size, but it does so very slowly (for example inversely proportional to the logarithm of network size).

Modeling Information Flow

- Diffusion as a general model of computation in large, wireless networks
- Example: failing-sensor bypass computation
  - Each node with a failing sensor takes for its own sensor reading the median of its neighbors’ available readings



failure rate 40%



failure rate 80%

failure rate 99%

Modeling Information Flow (cont.)

- How far does information diffuse?
  - Finite storage essentially limits diffusion – each node can retain information about only so many nodes
  - Arbitrary point-to-point communication is not tenable
- Experiment: majority computation in a random bit field
  - Each node replaces its own bit with the majority (= median!) of its neighbors’ bits and a random bit (think sensor input)
  - Run computation on two fields, initially identical except for one bit, with identical sensor processes
  - Visualize exclusive-or of fields at each step of computation
    - shows limited diffusion of initial perturbation
- Fundamental performance limits engendered by limited capacity of information flow in the field?

Modeling Algorithm Dynamics

- We have prototyped a modeling tool for rapid experimentation with algorithms on large ad-hoc wireless networks of computational nodes
  - Want to gather statistics on, e.g., rates of convergence
    - *Caveat*: the tool is *not* intended as a code validator and does not model all relevant aspects (accuracy traded in for speed)
  - High-level language to describe nodes' computations
  - Visualization of dynamic state of nodes
  - Fast enough for 1000s of nodes — statistics of scalability experiments

Example NEST Problem/Solution Formulated as Constraints

- We have collaborated with UCB, UCLA & UCI in outlining a typical NEST application (tracking) and in detailing some important components
  - system-wide constraints express the requirements
    - cameras must point towards fast-moving targets
  - refined constraints express solution method
    - motes compute local target estimates to best match what they measure and are told by nearby motes
    - measurements and/or target estimates diffuse through the network
    - cameras optimize quality of their own actions, measured with respect to their local knowledge
  - details to be presented later today

Example Simple NEST Algorithm as Constraint Maintenance

- We have formulated the “clubs” algorithm as a constraint maintenance problem/solution
  - the problem is to partition a “wireless” network into groups of reasonable size & diameter
  - the basic algorithm works as follows:
    - to start, each node remains silent for some random time
    - after that time, each node sends a *recruiting call* broadcast, unless it has already received a broadcast from a higher-ranking node
    - when a node receives a *recruiting call*, it records the highest-ranking sending node as its leader
    - a group consists of a leader with its followers

Constraint Formulation of Basic "Clubs" Algorithm

- every node is a member of a group of reasonable size & diameter
- strengthen: each group has a unique leader which every member can hear
- strengthen to break symmetry: introduce arbitrary ranking for nodes
  - unique ids or (probabilistic) local total-orders
- $\Rightarrow$ defines unique leader for every node: every node follows the highest-ranking node it can hear
- communication optimization: recruit followers in order of rank

Constraint Formulation issues

- Leaders may die or become incommunicado
- So, for the sake of robustness, this is not a one-shot algorithm, but an *ongoing activity*
- In general, NEST applications must not be formulated or thought of as working in “phases”, like for instance some “network initialization phase”
- Constraints are to be interpreted in the scope of a modal operator **Everywhere Eventually Always** — under loose (but quantifiable) interpretations of **Everywhere**, **Eventually** and **Always**
- As ongoing activity a (trivial) instance of diffusion

OEP Participation

- To date, we have focused on Berkeley OEP
- However, we aim for results that are as platform-independent as possible
- Platform-dependent aspects:
  - different versions of constraint-maintenance schema libraries
  - low-level code generation
  - (modeling for) analysis

Potential Role in OEP

- We are formulating NEST problem requirements in solution-independent fashions
  - describing what is to be achieved and
  - explicit *Optimality Metric*: how achievement can be measured/quantified relative to run-time costs (*mission-level* goals)
- We are describing NEST algorithms in an abstract framework amenable to modeling/analysis
  - emphasizing scalability, stability, dynamics
- We are formulating concrete, real-time/anytime, distributed solutions for specific NEST problems
  - we believe they epitomize a class of algorithms suited to NEST networks

Potential OEP Role (continued)

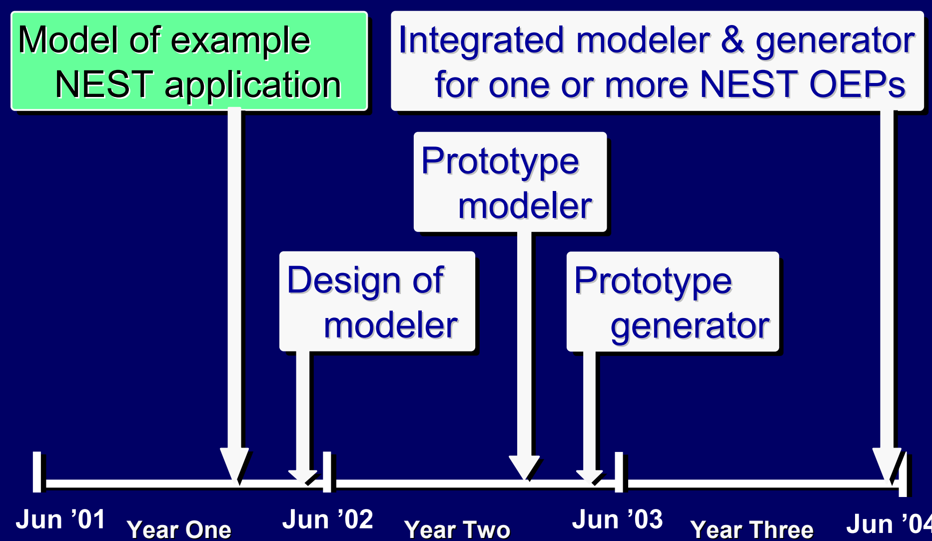
- We are developing a toolset supporting our modeling framework
- We are developing code generators that will ultimately target highly optimized code
- The modeling toolset and code generator are to be integrated into a (prototype) NEST application designer's workbench

OEP Contacts

- Kestrel: Lambert Meertens
  - lambert@kestrel.edu
- Berkeley: David Culler
  - culler@cs.berkeley.edu

Project Schedule

- Modeling using soft constraints: achieved
- Constraint technology: study on solver-driven of service integration June 2002
- Toolset: preliminary design June 2002
- Prototype modeling toolset March 2003
- Immediately: try-out Berkeley motes



Performance Goals

- Measures of success:
  - Flexibility of combining components
  - Dynamic adaptivity
  - Run-time efficiency
  - Correctness & maintainability of generated applications
- Metrics
  - adaptivity: some TBD benchmark application (e.g., tracking)
    - with range of events to which the system ought to adapt resource assignment to maintain optimality
    - measure degree to which appropriate reassignment occurs
  - efficiency: some TBD benchmark application
    - measure quality/resource usage compared with some baseline controller
  - correctness: number of bugs in generated code

Technology Transition/Transfer

- Kestrel Technology,LLC: Kestrel Institute spin-off vehicle for transitioning research
- Possible areas for commercial/scientific interest:
  - grid computing – distributed super-computers
  - more advanced peer-to-peer applications
- Status: speculative

Program Issues

- Metrics
  - Try to keep as clean a separation as possible between application-performance metrics and program-goal metrics
  - Use *target* line instead of (nonexistent) base line
- Need tight concrete complexity results for NEST
  - based on information-flow capacity considerations?
  - to serve as target line
  - to help identify bottlenecks in early stage

Program Issues (continued)

- Focus on extreme scalability
  - Note that on very small networks (< 250 nodes) less scalable solutions may outperform fully scalable solutions
  - Will it work on an infinite network? Pass to limit (continuum of infinitesimal nodes)?
- Future hardware profiles
  - What should we expect of mote-like systems in 5 years? (don't design for obsolescence on technology maturation)
- What is NEST's killer app?
  - Identify Technology Transition playing field
  - Focus program effort where it will count most