Spatial Computing for Networked Collaboration

Jacob Beal
May, 2010
When you just don't have the infrastructure...

- Emergency response & disaster rescue
- Developing nations & remote areas
- ... and many more
Agenda

- Spatial Computing
- Survey of Existing Approaches
- Proto & Amorphous Medium
Networked devices are filling our environment...
Networked devices are filling our environment...
Networked devices are filling our environment...

How do we program aggregates robustly?
Spatial Computers

Robot Swarms

Biological Computing

Cells during Morphogenesis

Reconfigurable Computing

Sensor Networks

Modular Robotics
More formally...

- A spatial computer is a collection of computational devices distributed through a physical space in which:
  - the difficulty of moving information between any two devices is strongly dependent on the distance between them, and
  - the “functional goals” of the system are generally defined in terms of the system's spatial structure
More formally...

- A spatial computer is a collection of computational devices distributed through a physical space in which:
  - the difficulty of moving information between any two devices is strongly dependent on the distance between them, and
  - the “functional goals” of the system are generally defined in terms of the system's spatial structure

Notice the ambiguities in the definition
Graphs

Crystalline (e.g. CAs)  ▪  Amorphous/Continuous

density → space complexity

jitter  ▼  grain size

(w. Dan Yamins)
Graphs

Crystalline (e.g. CAs)

Amorphous/Continuous

spatial computing

density

jitter

grain size

(w. Dan Yamins)
Space/Network Duality

How well does the network cover space?

What space is covered well by the network?
Tentative Mathematical Definition

- A spatial computer is any set of $n$ devices s.t.
  - Graph $\{V,E\}$ with edge weights $w(v_1,v_2)$
  - Manifold $M$, with distance function $d$
    - $M$ is compact, Riemannian  (*may be stronger than needed*)
  - Position function $p: V \rightarrow M$
  - $w(v_1,v_2) = O(1/d(p(v_1),p(v_2)))$

Examples: unit disc network, chemical diffusion
I need a med team!

We're on our way

Example: Disaster Relief
Example: Museum Guide

I would like to see the Mona Lisa, avoiding the queues...

I've gotten lost! How can I rejoin my friends?
Example: Mobile Streaming

I want Alice to be able to listen in on this great conversation.
How can we program these?

- Desiderata for approaches:
  - Simple, easy to understand code
  - Robust to errors, adapt to changing environment
  - Scalable to potentially vast numbers of devices
  - Take advantage of spatial nature of problems
Agenda

- Spatial Computing
- **Survey of Existing Approaches**
- Proto & Amorphous Medium
A Taxonomy of Approaches

Spatial
  Geometry

Dynamics
  Uniform
  Viral

Non-Spatial
  Non-Composable
A Taxonomy of Approaches

Spatial

Geometry

Dynamics

Uniform

Viral

Non-Spatial

Non-Composable
Approaches from Local Dynamics

Primitives describe only actions between devices and the neighbors they communicate with.

- Advantages: coherent and correct semantics
- Disadvantages: programmer must figure out how to marshal local dynamics to produce coherent large-area programs
Proto: Computing with Fields

(def gradient (src) ...)
(def distance (src dst) ...)
(def dilate (src n)
  (<= (gradient src) n))
(def channel (src dst width)
  (let* ((d (distance src dst))
      (trail (<= (+ (gradient src)
        (gradient dst))
        d)))))
(dilate trail width)))
Other Uniform Approaches

- LDP/MELD (CMU Claytronics group)
  - Distributed logic programs
  - Local resolution leads to long-distance properties
TOTA: Viral tuples

T =

C = \{ \text{value} = "2", \text{color} = "green" \}

P = \{ \text{propagate to all nodes, decrease "value" for the first 2 hops then increase it, change color at every hop} \}
Other Viral Approaches

- Smart Messages (Borcea)
  - Execution migrates to nodes of interest, found via self-routing code packets
- Paintable Computing (Butera)
  - Consistent transfer, view of neighbor data
  - Code for install, de-install, transfer-granted, transfer-denied, update
- RGLL (Sutherland)
  - Code for arrival, tick, collision, departure
  - Communication via collision
Approaches from Geometry

Primitives describe large-scale geometric regions (e.g. “all devices on the left hill”)

- Advantages: coherent, easy to specify large-scale programs
- Disadvantages: generally easy to accidentally specify programs that cannot be executed correctly
MGS

Meristem formation

Turing pattern on torus

Michel, Giavitto, Spicher
Regiment

- Streaming collection of data from regions
  - Spatial primitives:
    - K-hop neighborhood
    - K-nearest nodes
  - Composition:
    - Union/Intersection
    - Map/Filter
- Distributed execution as a compiler optimization
Other Geometric Approaches

- Borcea's Spatial Programming
- EgoSpaces
- SpatialViews
- Spidey
- Abstract Regions
- Growing Point Language
- Origami Shape Language
Non-Composable Approaches

Algorithms and techniques, generally based on geometry, but not part of a system of composable parts

- Advantages: powerful spatial ideas for that are good for inclusion in code libraries
- Disadvantages: developed as stand-alone ideas, and may have limited composability
Field-Based Coordination

Mamei & Zambonelli
Self-Healing Gradients
Local Check-Schemes

Yamins
Other Non-Composable Approaches

- hood (Whitehouse, et. al.)
  - nesC library for interacting with neighbors
- McLurkin's “Stupid Robot Tricks”
  - Swarm behaviors intended mainly for time-wise multiplexing.
- Countless one-shot systems...
Significant Non-Spatial Approaches

- “roll-your-own” (e.g. C/C++)
- TinyDB
  - Distributed database queries for sensor networks
- Kairos
  - Distributed graph algorithms
- WaveScript
  - Distributed streaming language
  - Follow-on to Regiment w/o the spatial primitives
Summary

• Many approaches exist to programming pervasive applications for spatial computers
• Only approaches based on local dynamics currently offer predictable composition, correct execution, and spatial primitives
• Challenge: obtaining long-range coherent behavior from local dynamics
Agenda

- Spatial Computing
- Survey of Existing Approaches
- Proto & Amorphous Medium
Geometric Program: Channel

Source

Destination

(cf. Butera)
Geometric Program: Channel

Source

Destination

(cf. Butera)
Geometric Program: Channel

Source ——— Destination

(cf. Butera)
Geometric Program: Channel

Source

Destination

(cf. Butera)
Geometric Program: Channel

(c.f. Butera)
Geometric Program: Channel

(cf. Butera)
Geometric Program: Channel

Source

Destination

(cf. Butera)
Why use continuous space?

- Simplicity
- Scaling & Portability
- Robustness

*(we'll come back to this in a bit...)*
Amorphous Medium

- Continuous space & time
- Infinite number of devices
- See neighbors' past state

Approximate with:
- Discrete network of devices
- Signals transmit state
Computing with fields

source

gradient

+ 

gradient

destination

distance

<=

dilate

width
Computing with fields

source ➔ gradient ➔ + ➔ <= ➔ dilate

destination ➔ gradient ➔ distance

width ➔ 10 ➔ 37
(def gradient (src) ...)
(def distance (src dst) ...)
(def dilate (src n)
  (<= (gradient src) n))
(def channel (src dst width)
  (let* ((d (distance src dst))
         (trail (<= (+ (gradient src)
                     (gradient dst))
                  d)))
     (dilate trail width)))
Proto's Families of Primitives

Pointwise

Feedback

Restriction

Neighborhood

delay

41

7

+ 48

restrict

nbr

any-hood
Modulation by Restriction

source → channel

destination → channel

channel → 10

ggradcast

coord
Why use continuous space?

- Simplicity
- Scaling & Portability
- Robustness

2000 devices

150 devices
Device Motion = Vector Fields

- brownian
- flock
- cluster-to
- contour-field
- search-and-rescue
Diving into the details

Let's build this up using the Proto simulator, one piece at a time...

(break to work w. simulator)
In simulation...
Example: Disaster Relief

I need a med team!

We're on our way
In simulation...
Weaknesses

- Functional programming scares people
- Programmers can break the abstraction
- No dynamic allocation of processes
- No formal proofs available for quality of approximation in a composed program

(active research on last two)
Summary

- Amorphous Medium abstraction simplifies programming of space-filling networks
- Proto has four families of space and time operations, compiles global descriptions into local actions that approximate the global
- Geometric metaphors allow complex spatial computing problems to be solved with very short programs.
Proto is available

http://stpg.csail.mit.edu/proto.html
(or google “MIT Proto”)

- Includes libraries, compiler, kernel, simulator, platforms
- Licensed under GPL (w. libc-type exception)

Feedback on session:
- CTS2010 Website: Click “feedback” for tutorial
- Password: cts10bluestar