Introduction to Spatial Computing

Jacob Beal
April, 2009

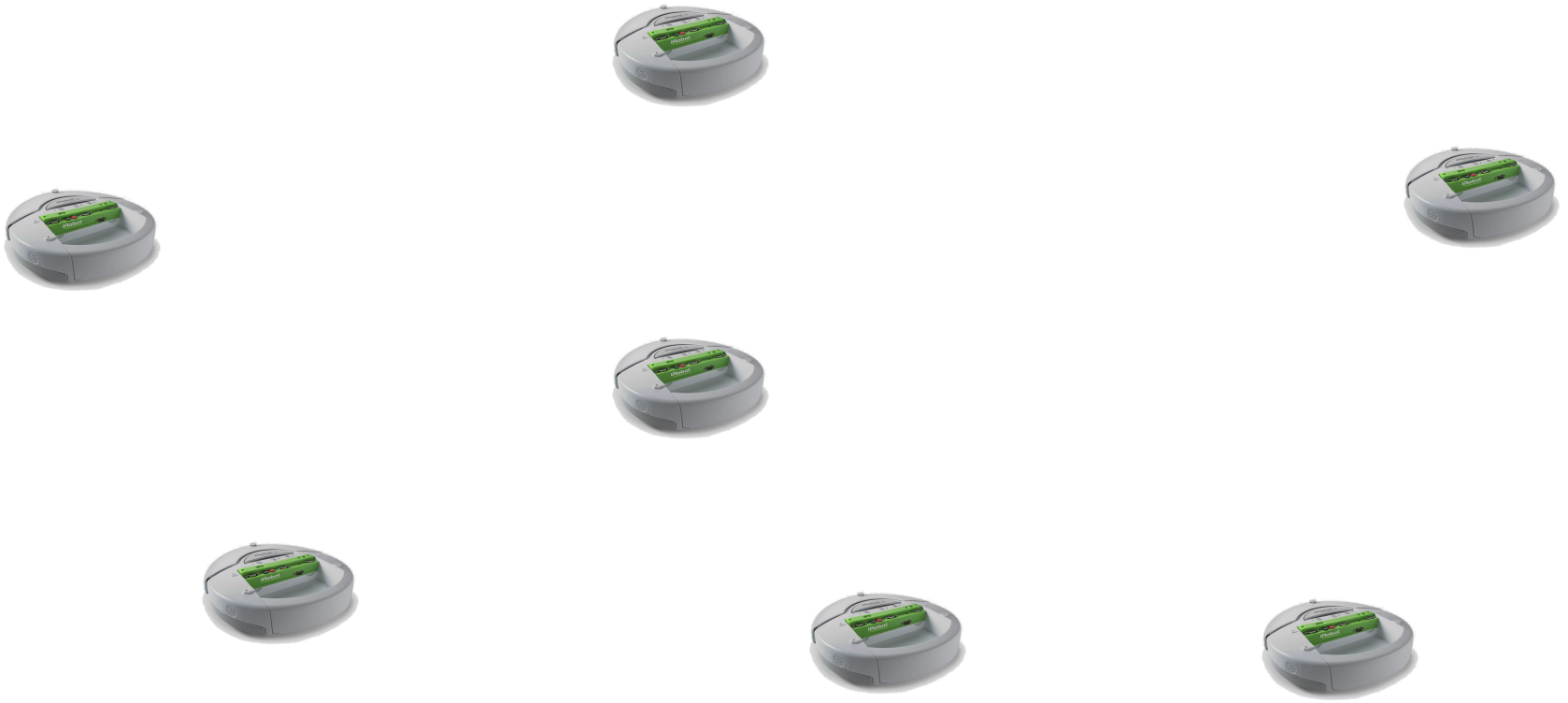
BBN TECHNOLOGIES

Agenda

- Spatial Computing
- Survey of Existing Approaches
- Proto & Amorphous Medium

From one robot, to many

From one robot, to many



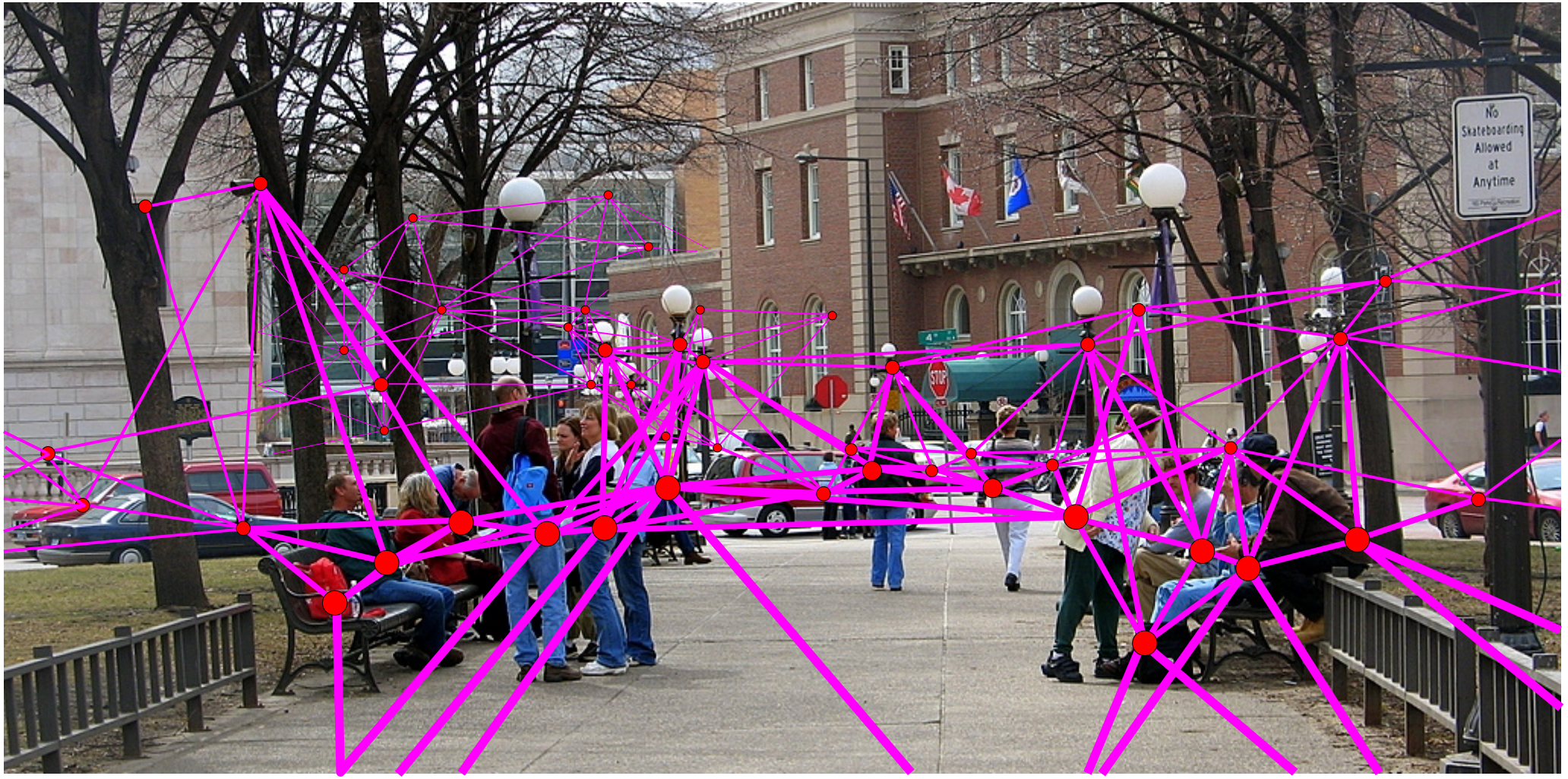
From one robot, to many

Robotic density is currently very low, but...

Networked devices are **filling** our environment...

Networked devices are **filling** our environment...

Networked devices are **filling** our environment...



How do we program aggregates robustly?

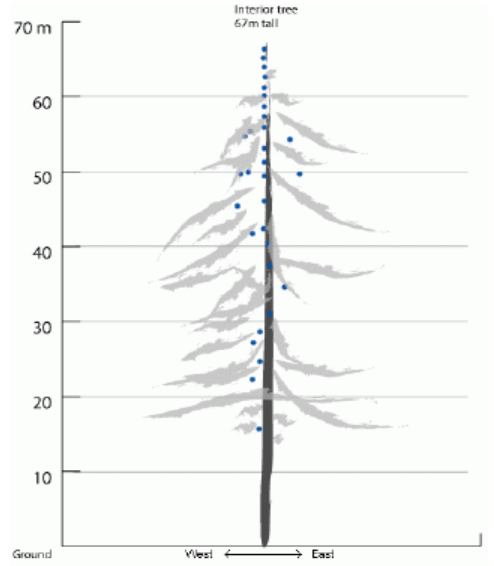
Wireless-enabled Embedded Systems

- >3.3B cell phones vs. 600M Internet-connected PC's in 2007
  - >600M cell phones with Internet capability, rising rapidly
- New cars come equipped with navigation systems and will soon have wireless interfaces (WiFi/DSRC, cellular, WiMax)
- Sensor deployment just starting, but some estimates ~5-10B units by 2015
- Military/emergency response wireless robots, unmanned vehicles, unmanned aircraft

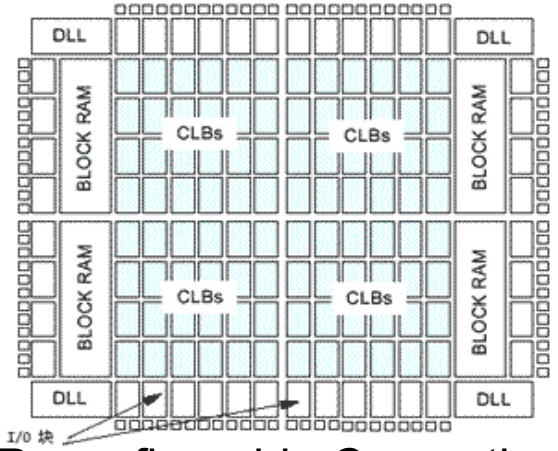
Spatial Computers

Robot Swarms

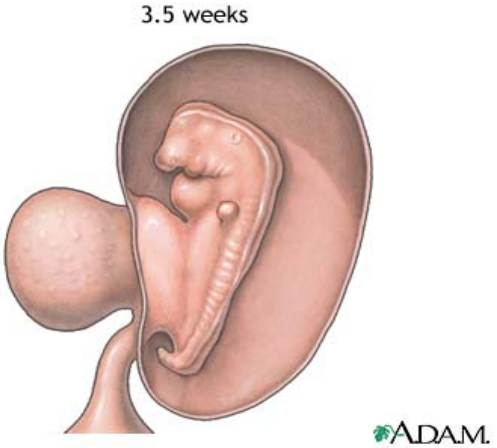
Biological Computing



Sensor Networks



Reconfigurable Computing



Cells during Morphogenesis

Modular Robotics

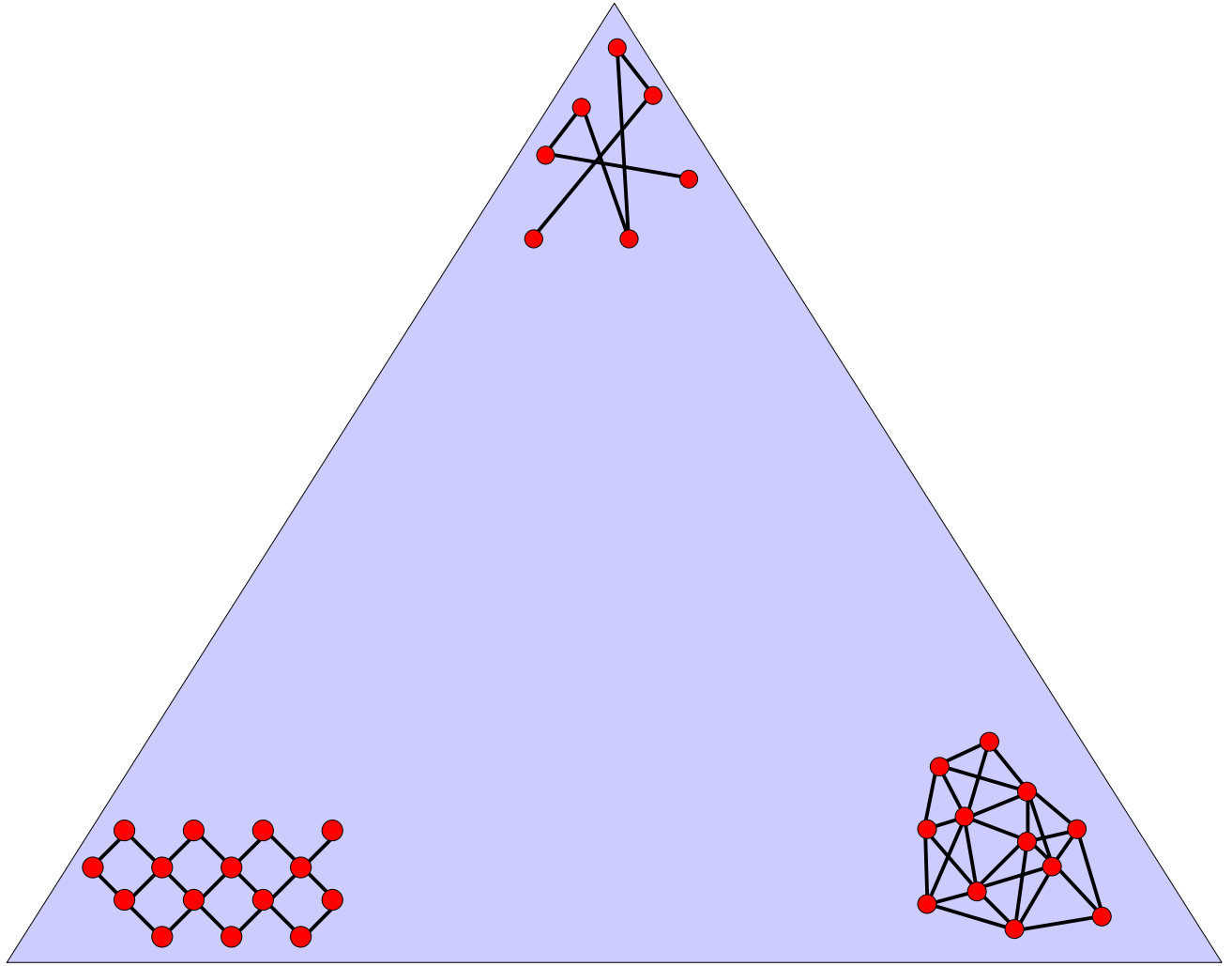
More formally...

- 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

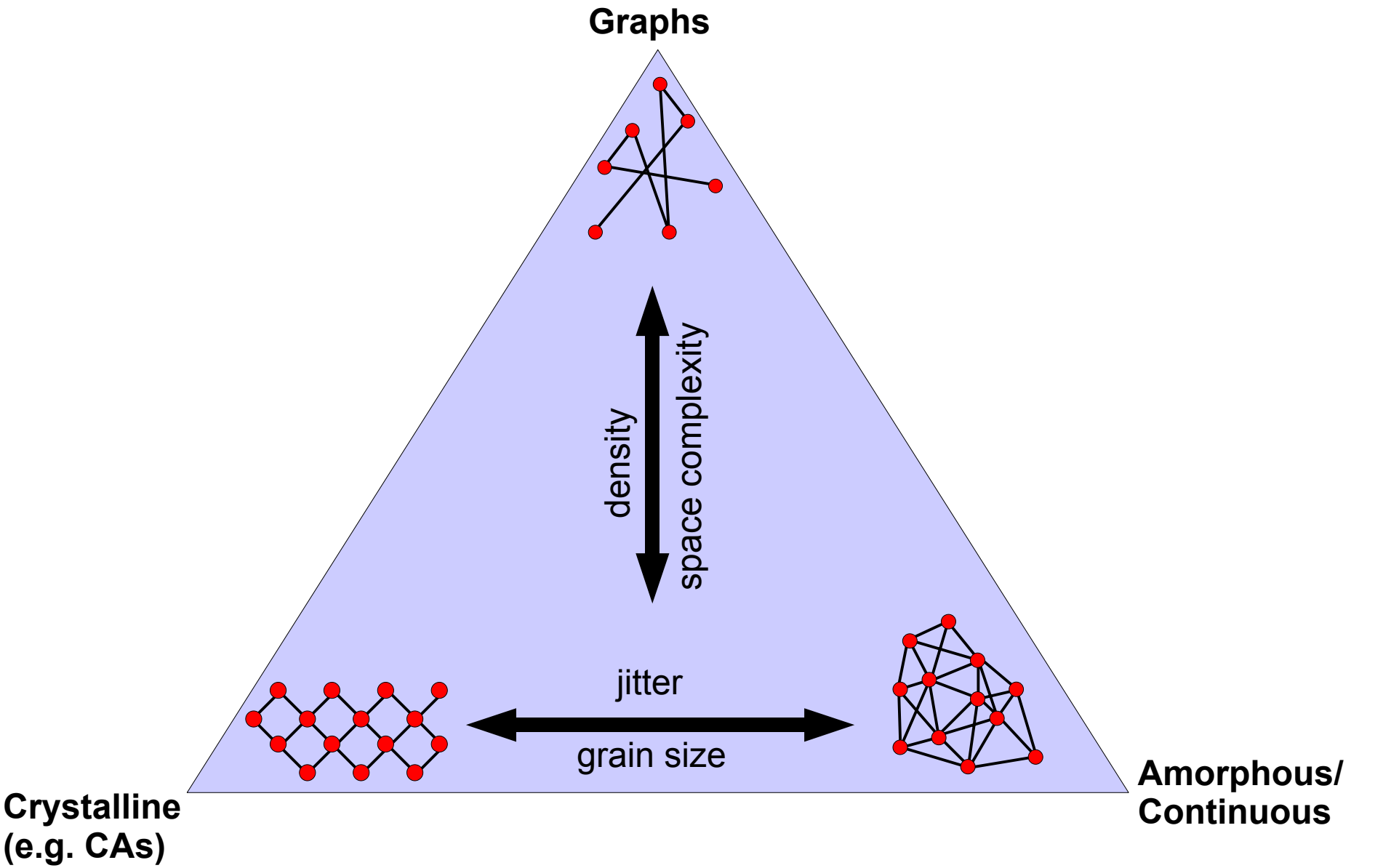
More formally...

- 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

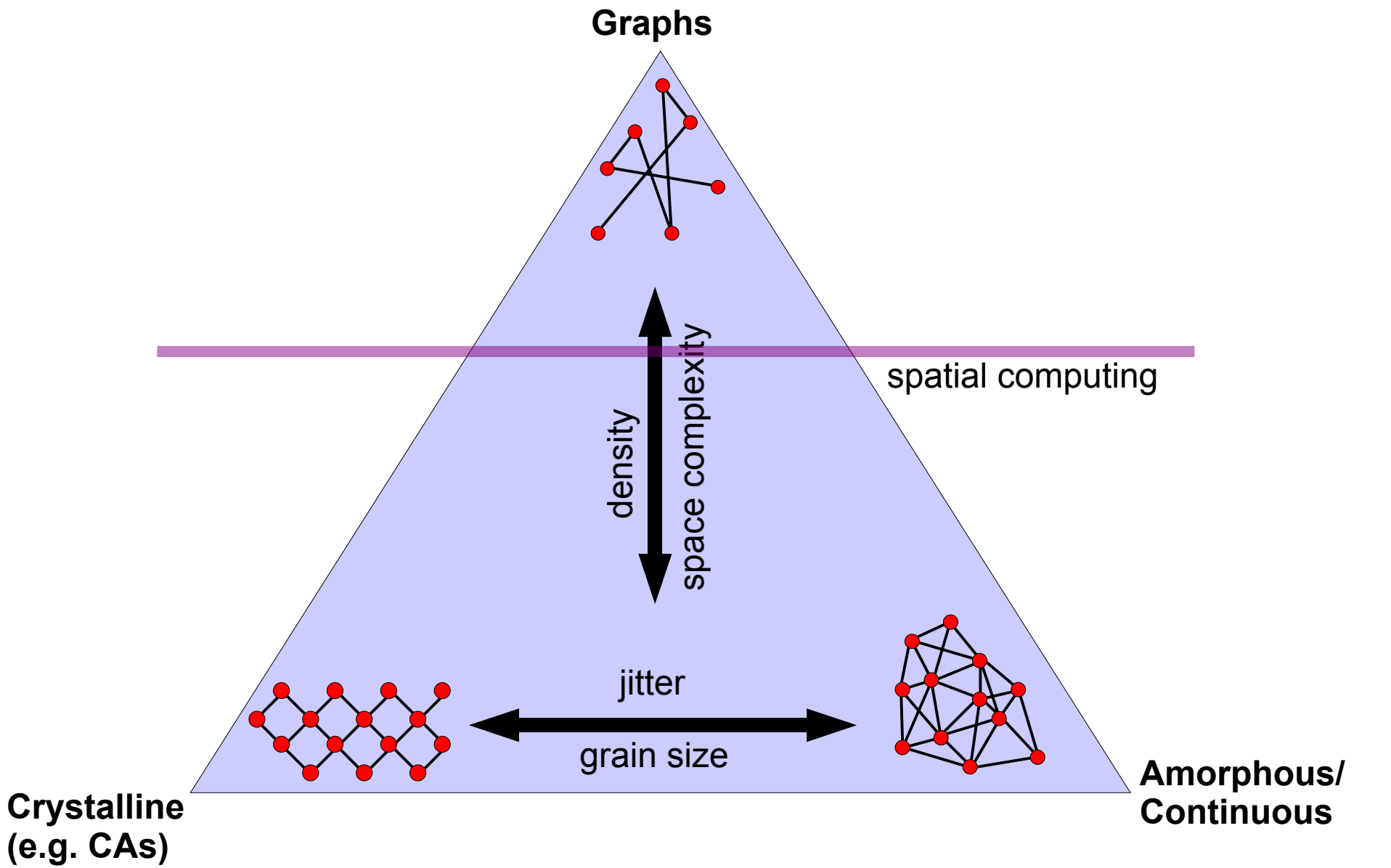
Notice the ambiguities in the definition



(w. Dan Yamins)

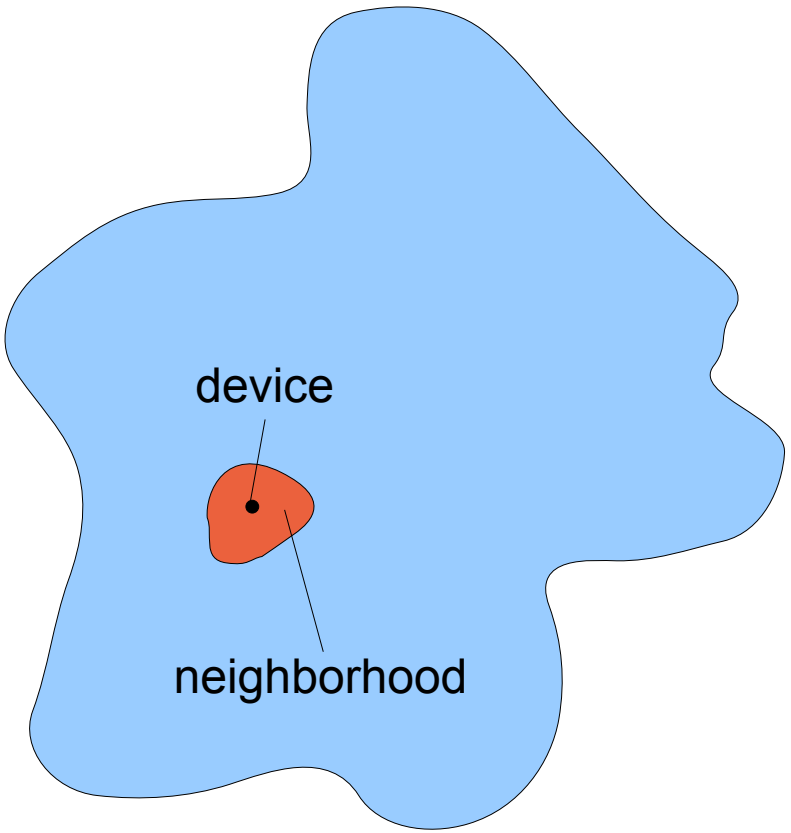

Graphs
density
space complexity
jitter
grain size
Crystalline
(e.g. CAs)
Amorphous/
Continuous

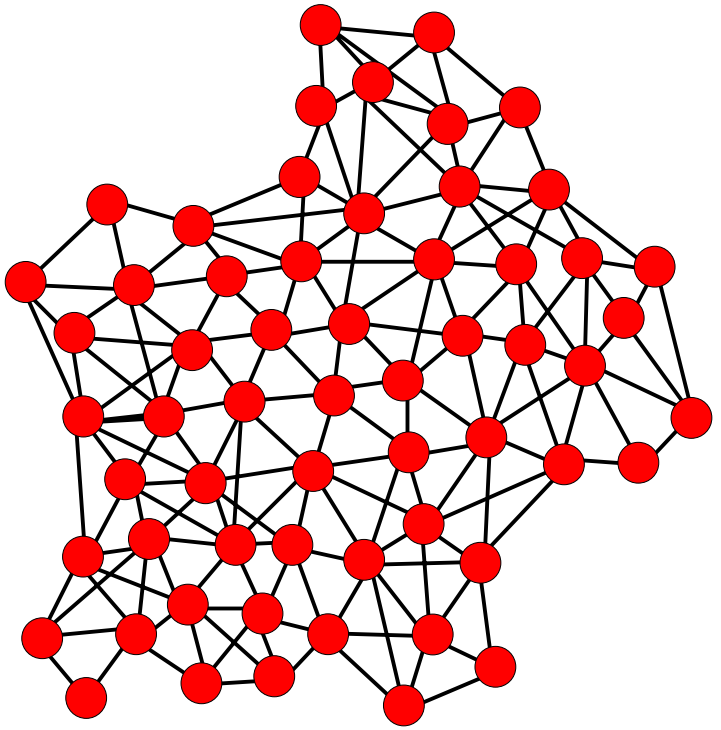
(w. Dan Yamins)


Graphs
spatial computing
density
space complexity
jitter
grain size
Crystalline
(e.g. CAs)
Amorphous/
Continuous

(w. Dan Yamins)

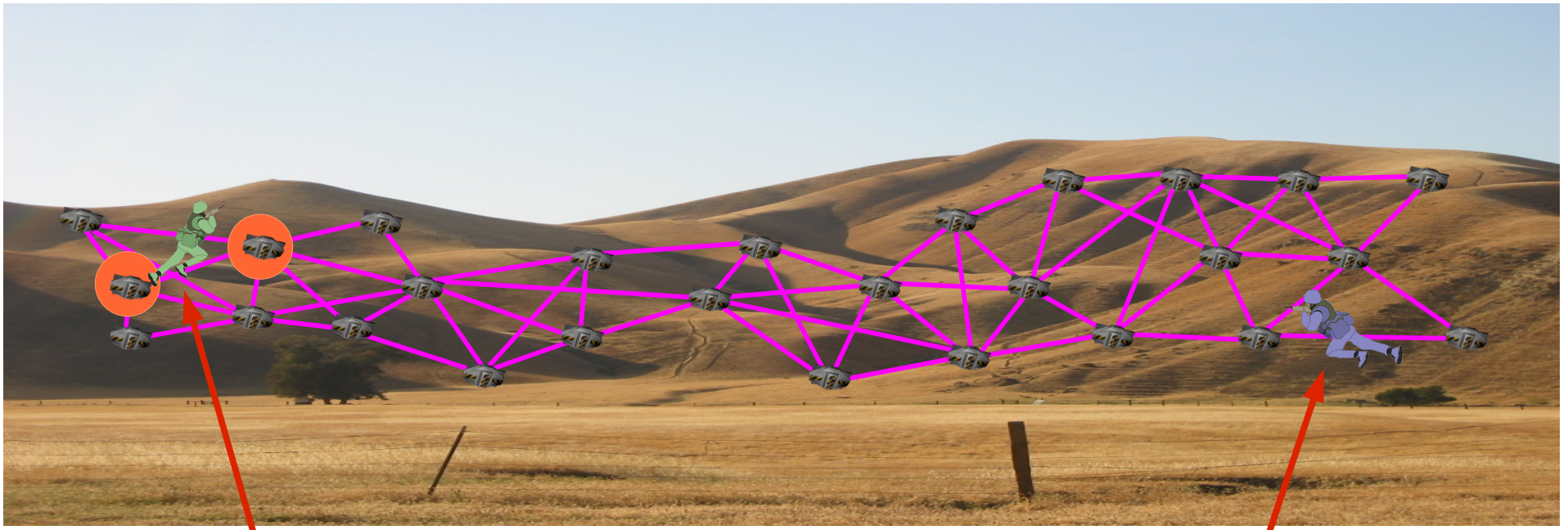
Space/Network Duality



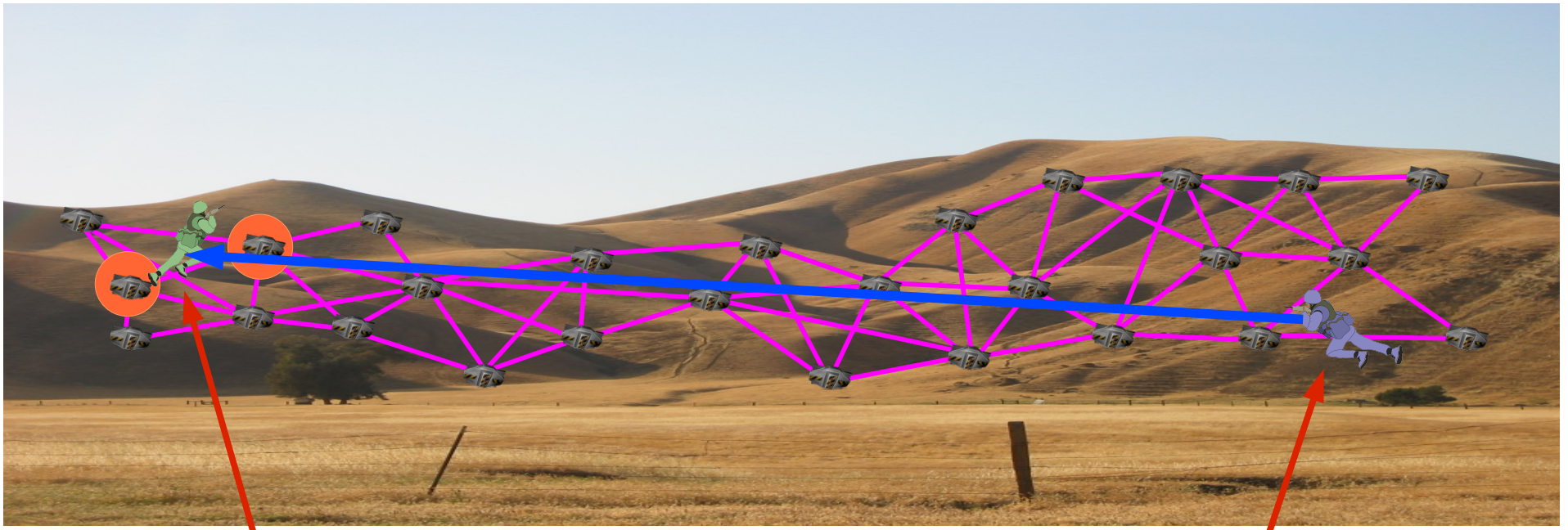


Example: Target Tracking

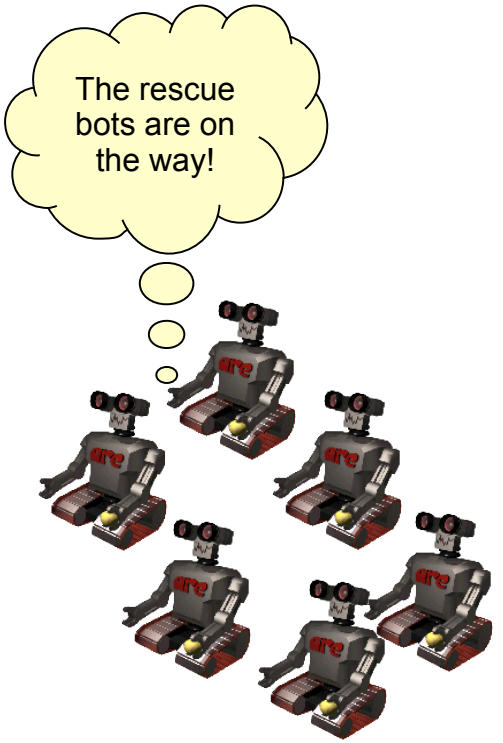
Example: Target Tracking



Example: Target Tracking



Example: Search & Rescue



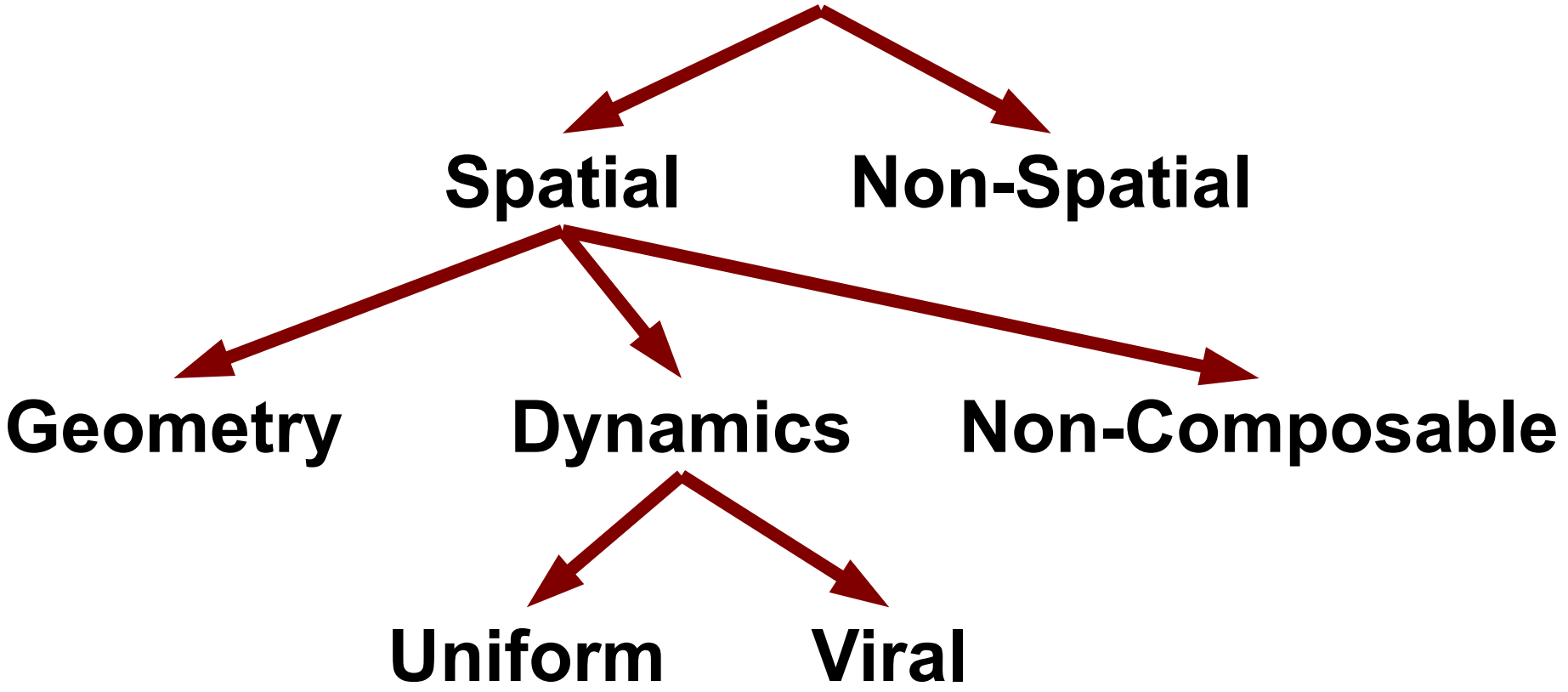
How can we program these?

- 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

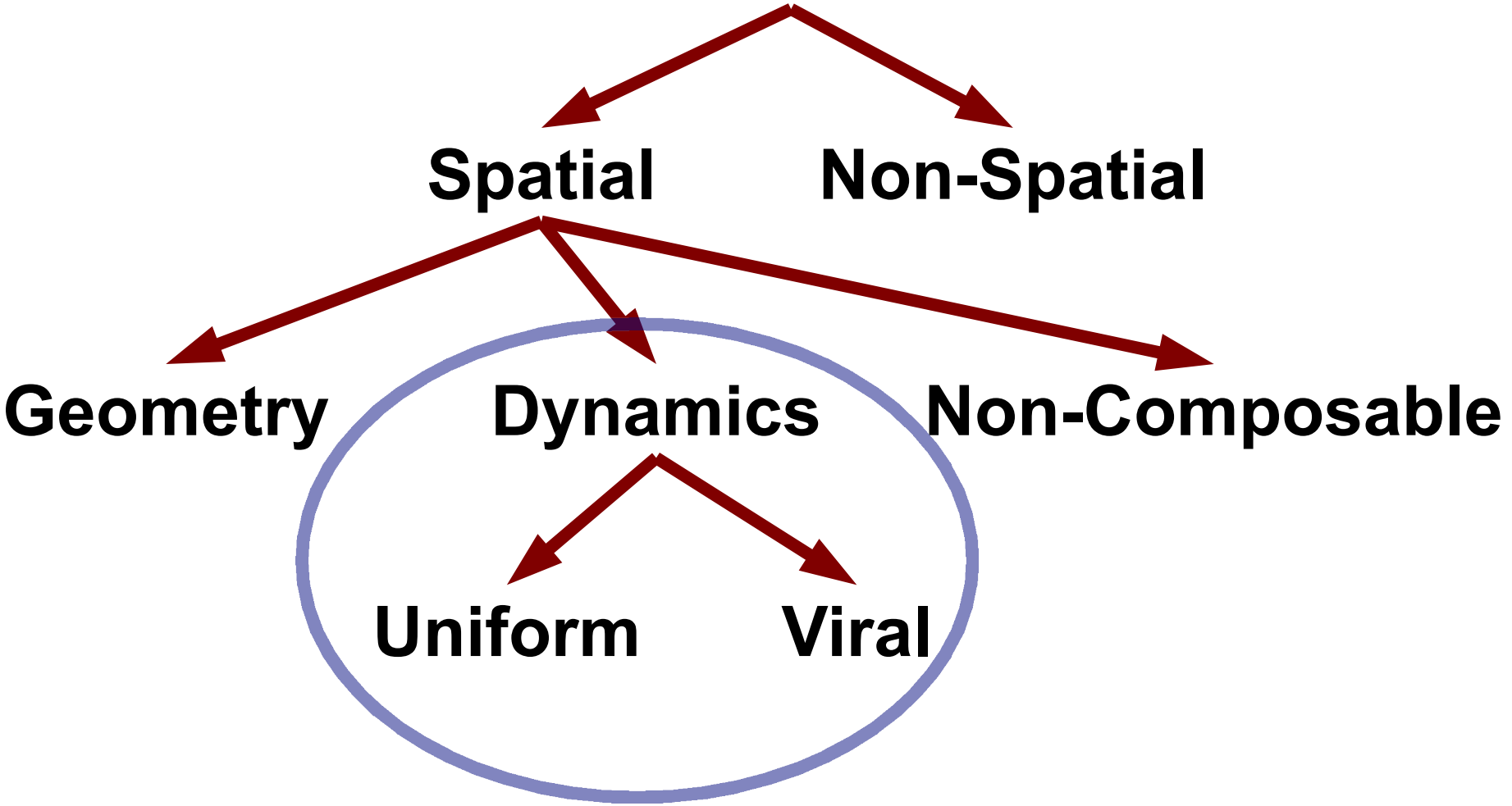
Agenda

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

A Taxonomy of Approaches



A Taxonomy of Approaches



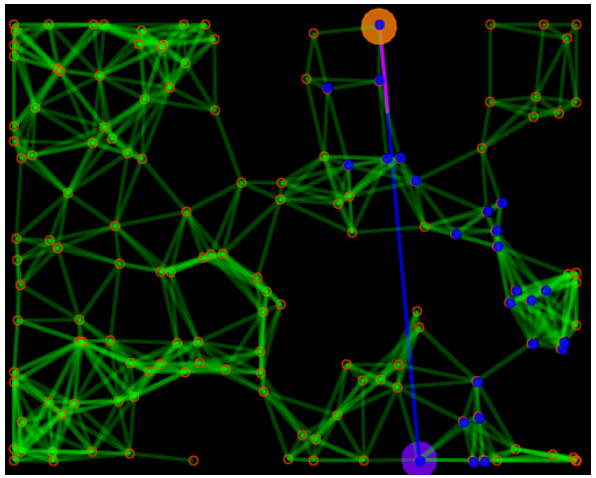
Approaches from Local Dynamics

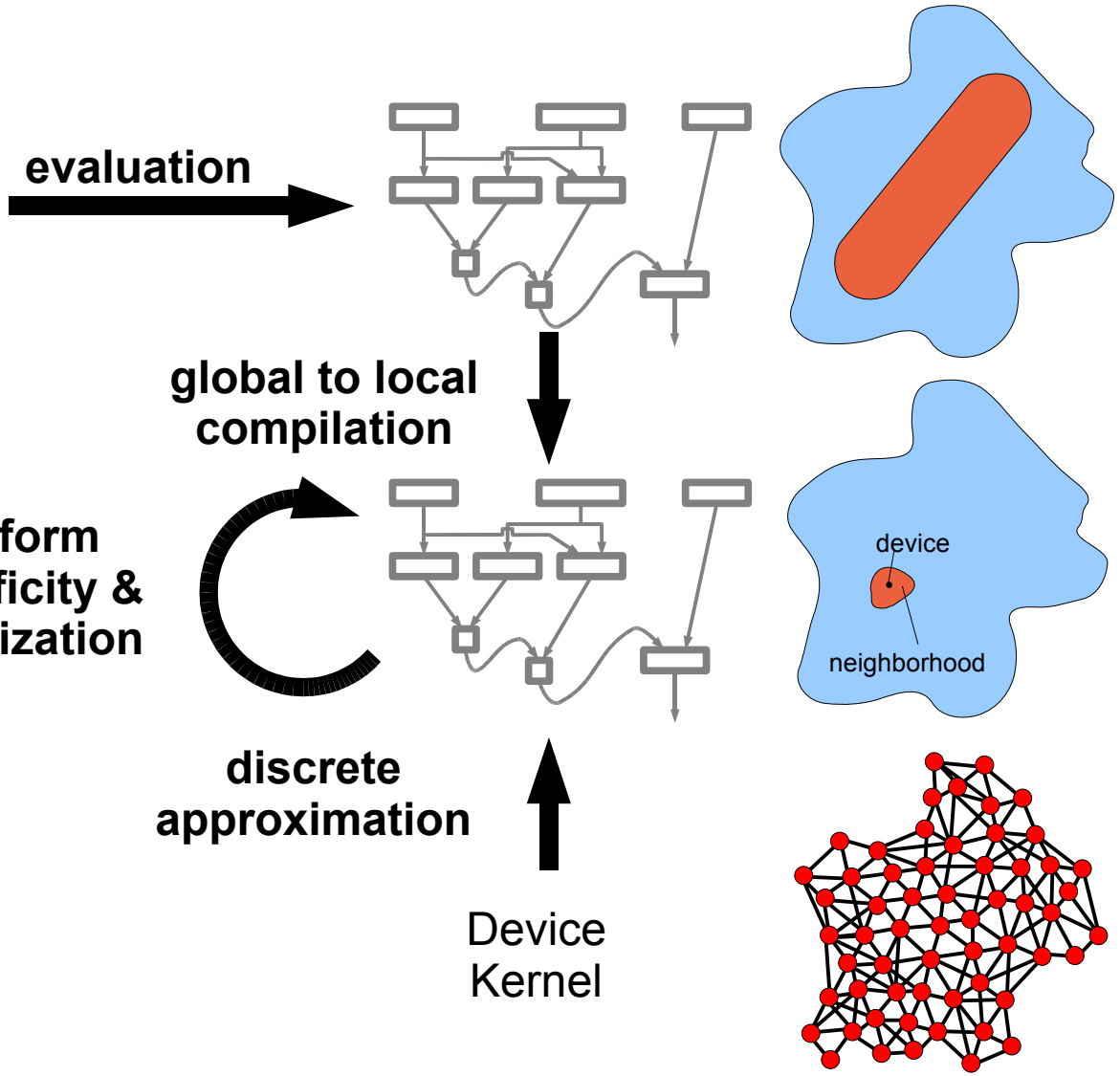
Primitives describe only actions between devices and the neighbors they communicate with.

- 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)))
```



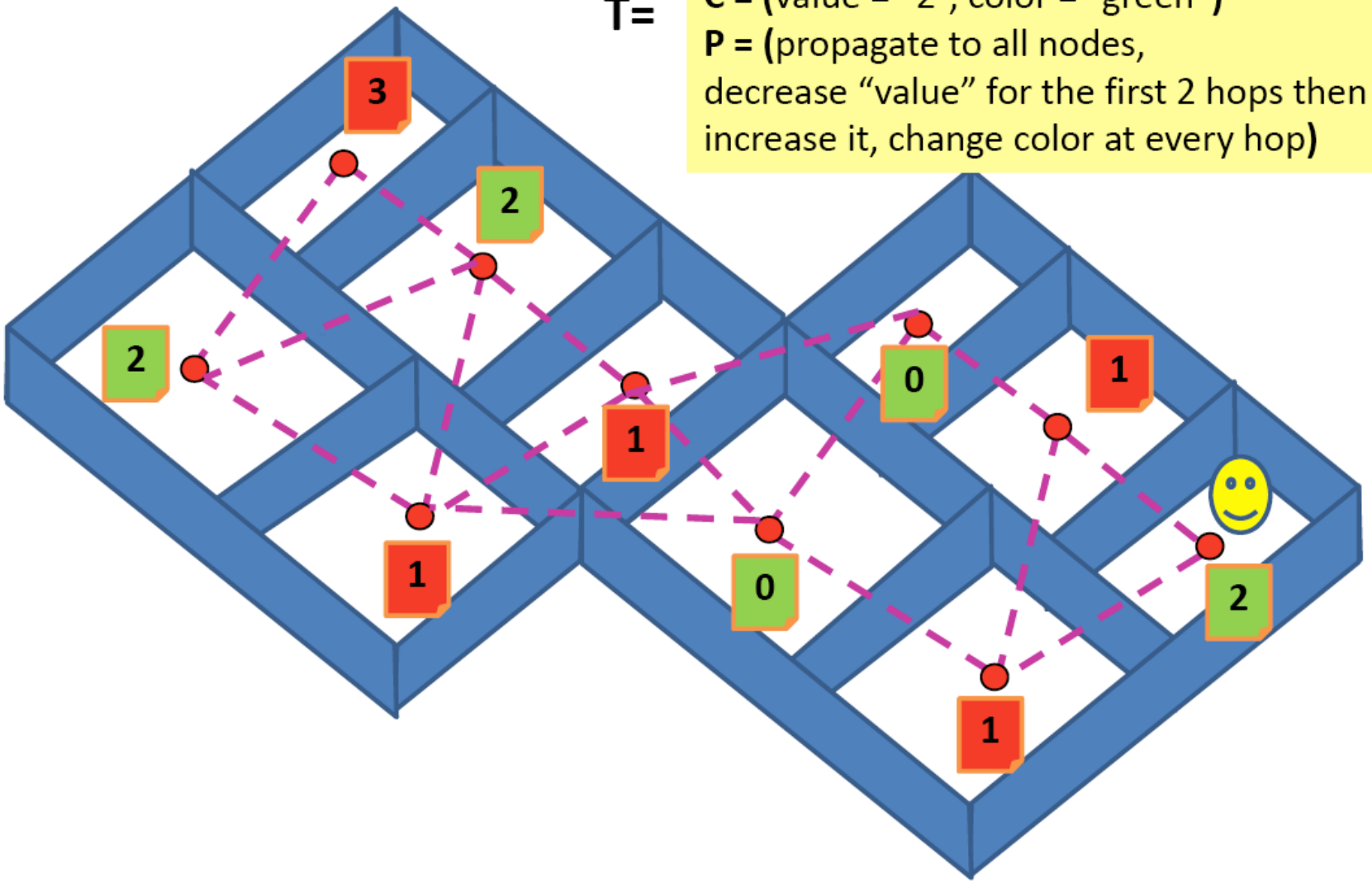


Beal & Bachrach

TOTA: Viral tuples

T=

C = (value = “2”, color = “green”**)**
P = (propagate to all nodes,
decrease “value” for the first 2 hops then
increase it, change color at every hop**)**



Other Viral Approaches

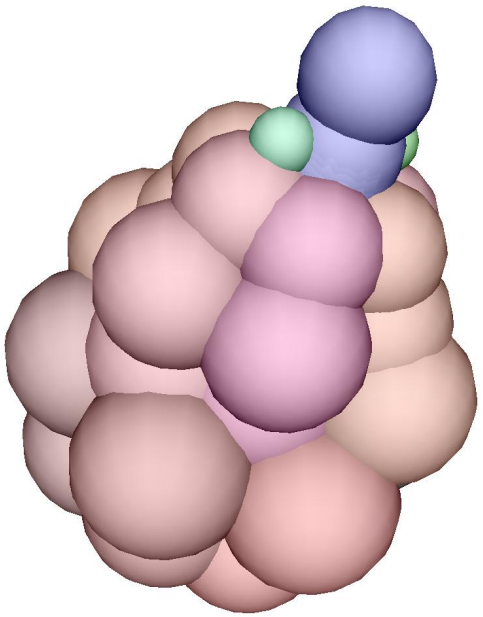
- Smart Messages (Borcea)
  - Execution migrates to nodes of interest, found via self-routing code packets
- Paintable Computing (Butera)
  - Consistent transfer, view of neighbor data
  - Code for install, de-install, transfer-granted, transfer-denied, update
- RGLL (Sutherland)
  - Code for arrival,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

Approaches from Geometry

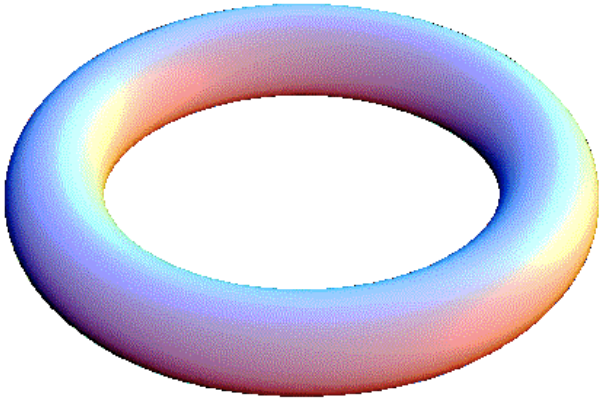
Primitives describe large-scale geometric regions (e.g. “all devices on the left hill”)

- 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

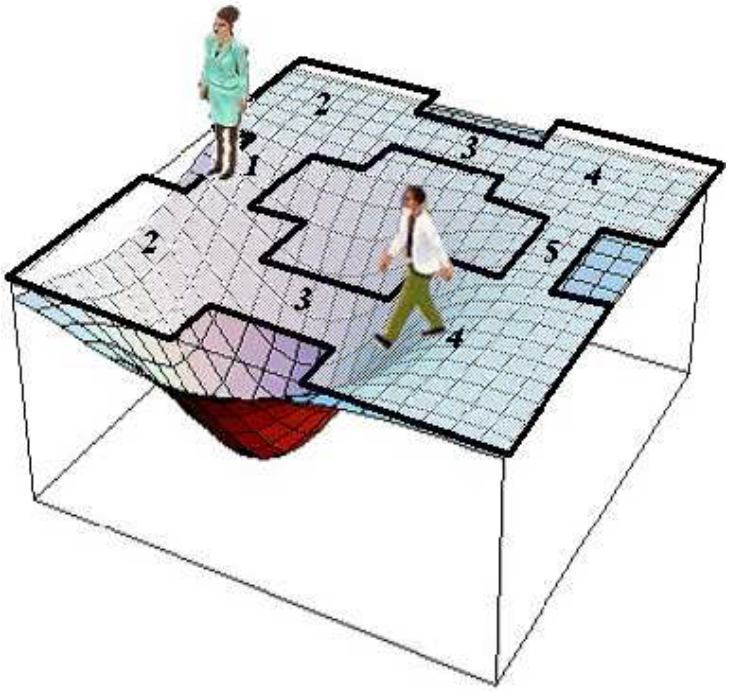
- Borcea's Spatial Programming
- EgoSpaces
- SpatialViews
- Spidey
- Abstract Regions

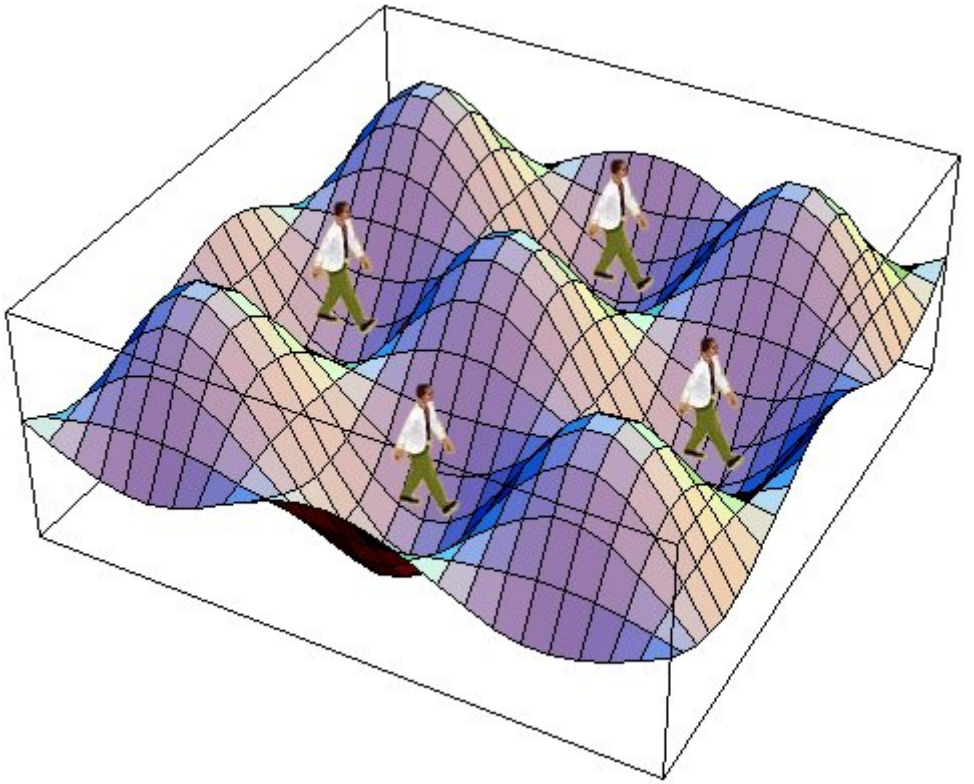
Non-Composable Approaches

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

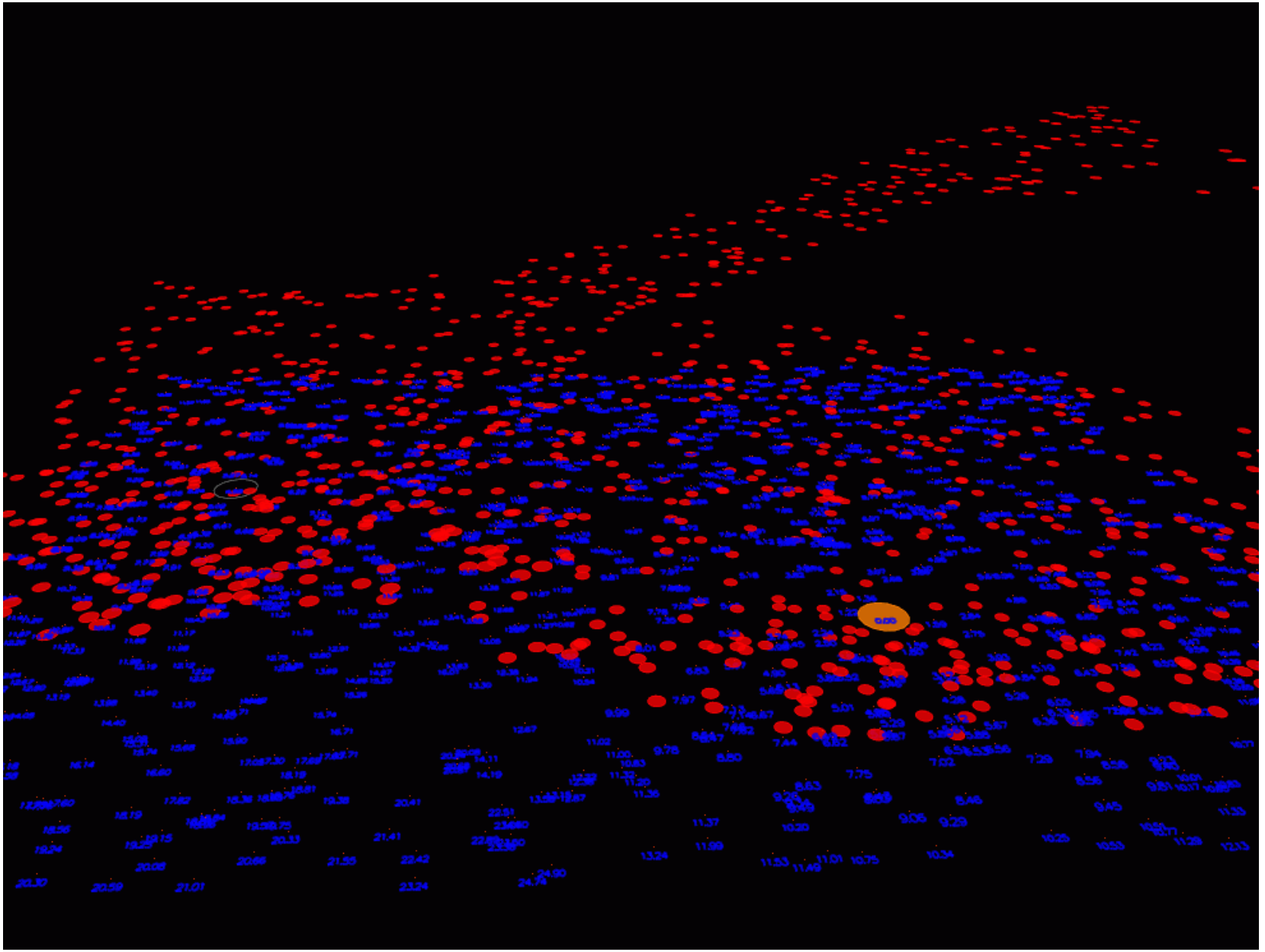
Field-Based Coordination



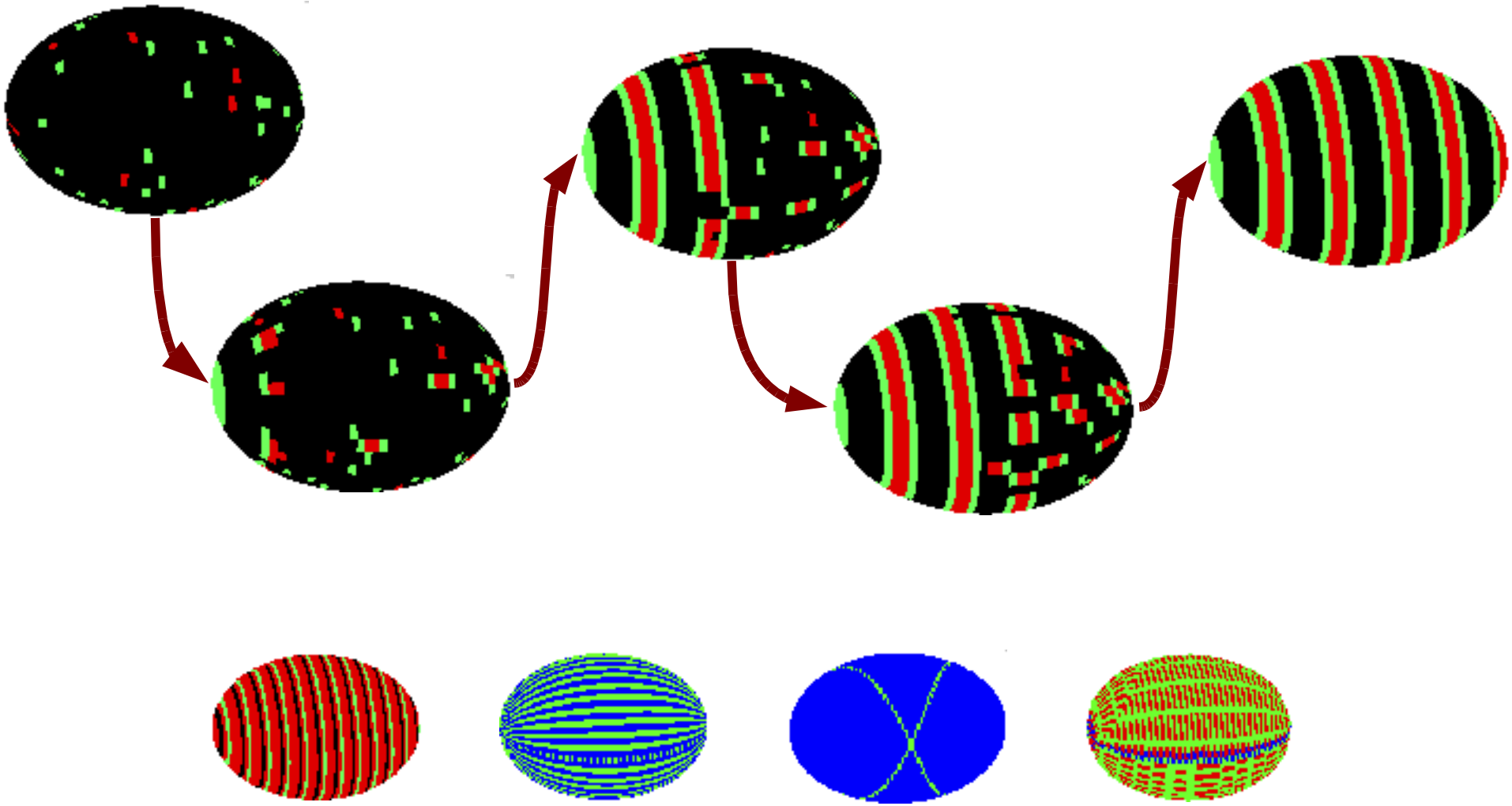


Mamei & Zambonelli

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

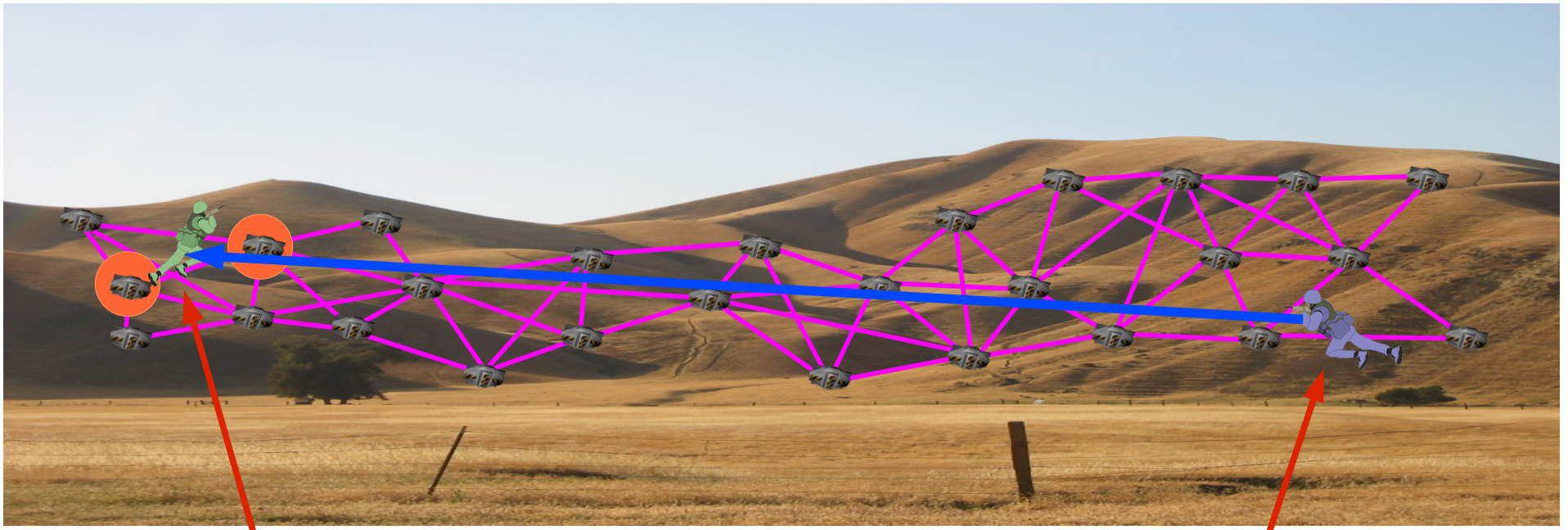
Summary

- 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**

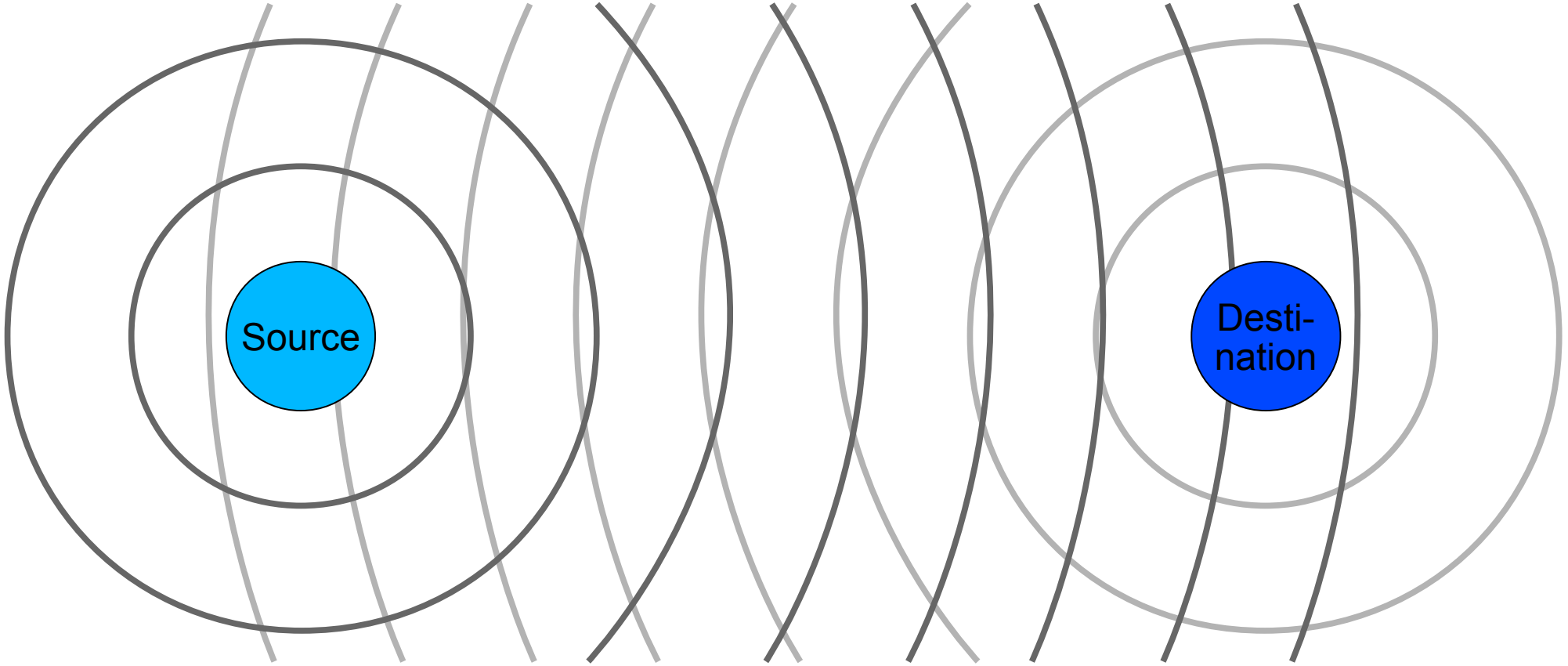
Example: Target Tracking



Geometric Program: Channel

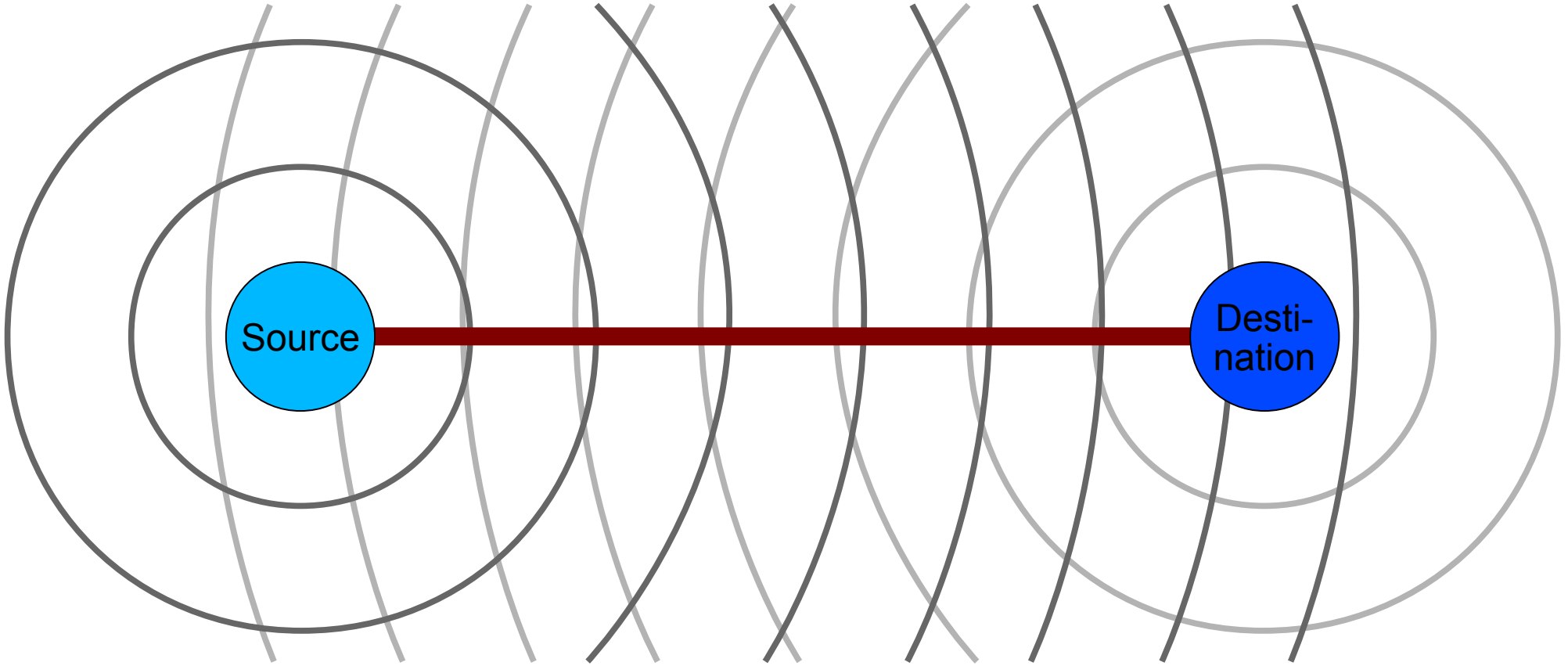
(cf. Butera)

Geometric Program: Channel



(cf. Butera)

Geometric Program: Channel

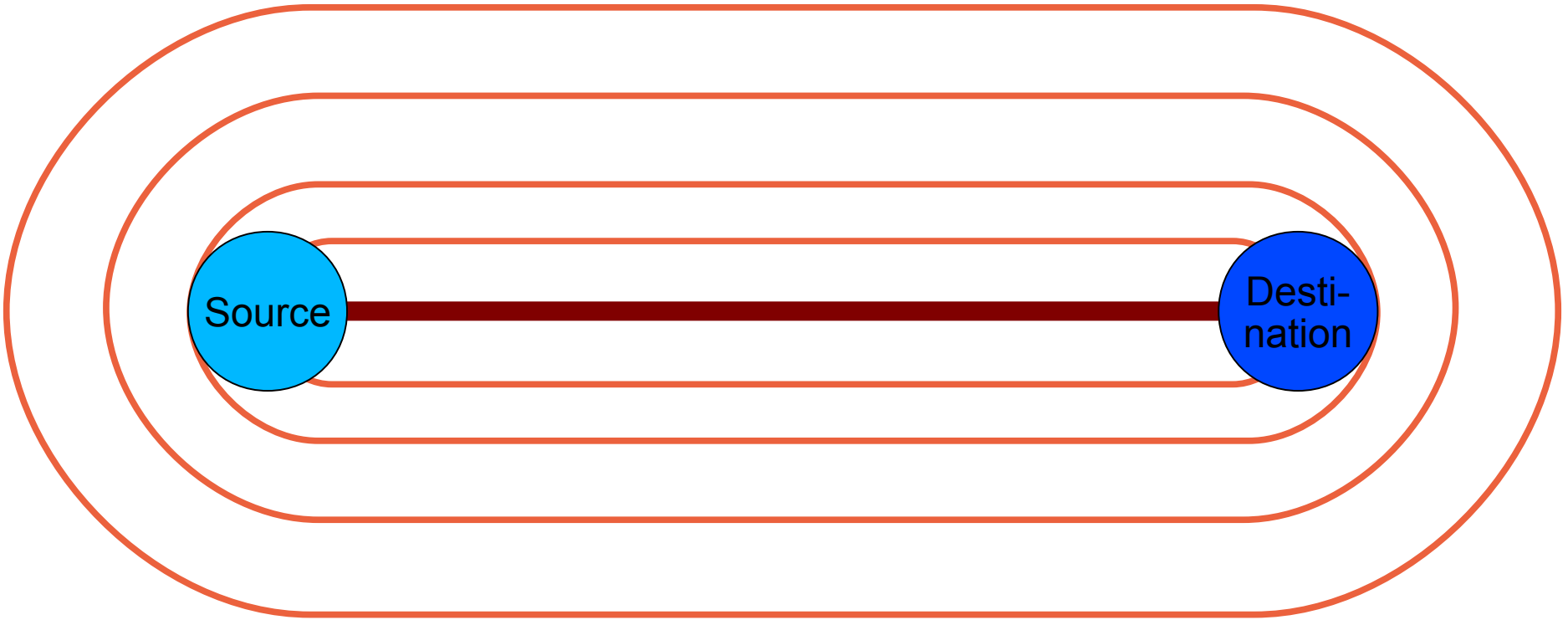


(cf. Butera)

Geometric Program: Channel

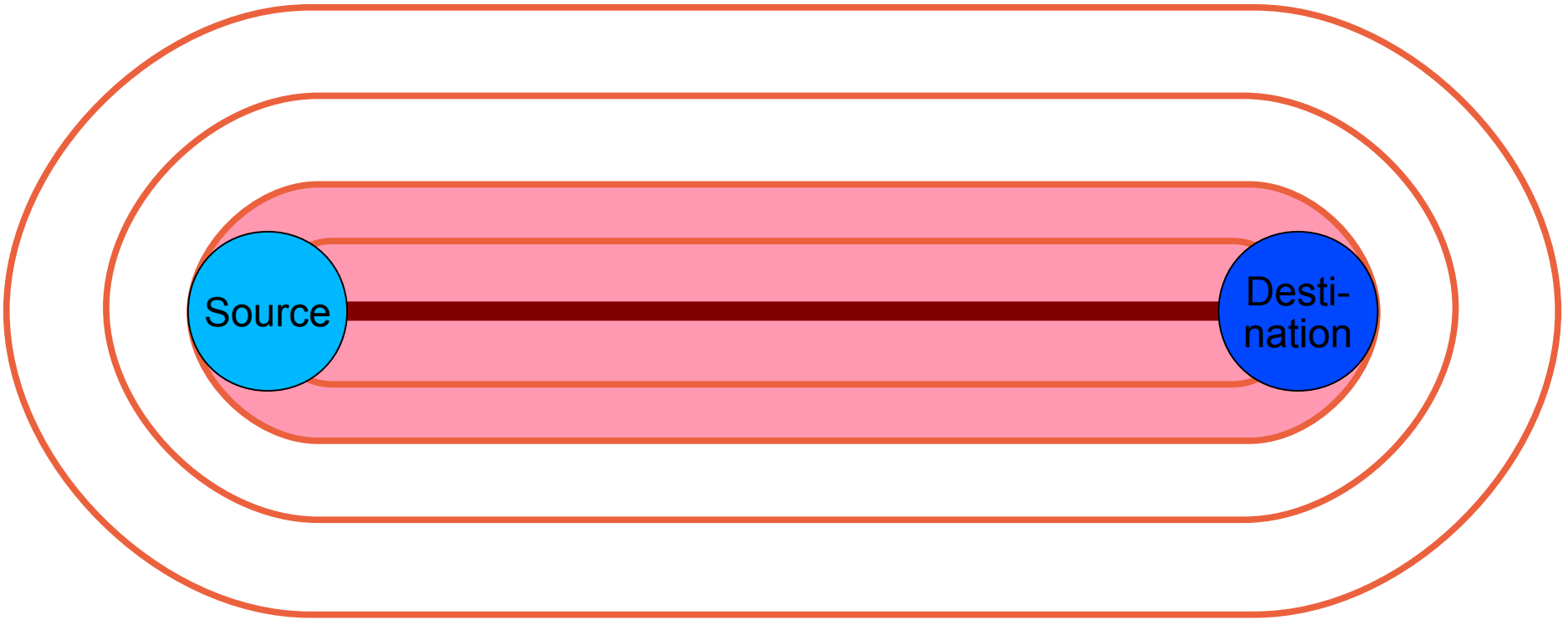
(cf. Butera)

Geometric Program: Channel



(cf. Butera)

Geometric Program: Channel



(cf. Butera)

Geometric Program: Channel

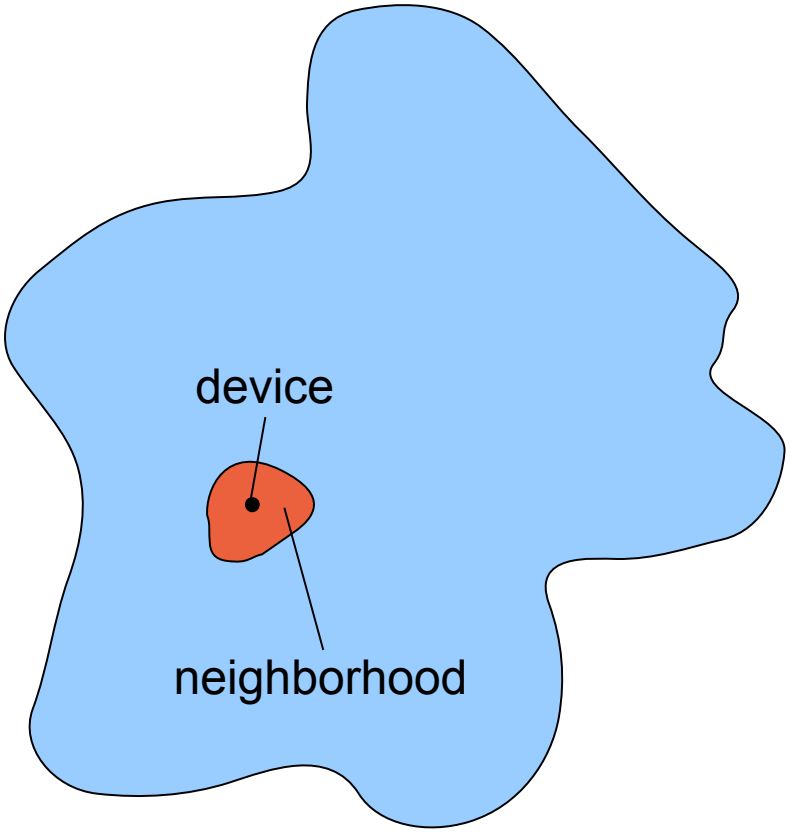
(cf. Butera)

Why use continuous space?

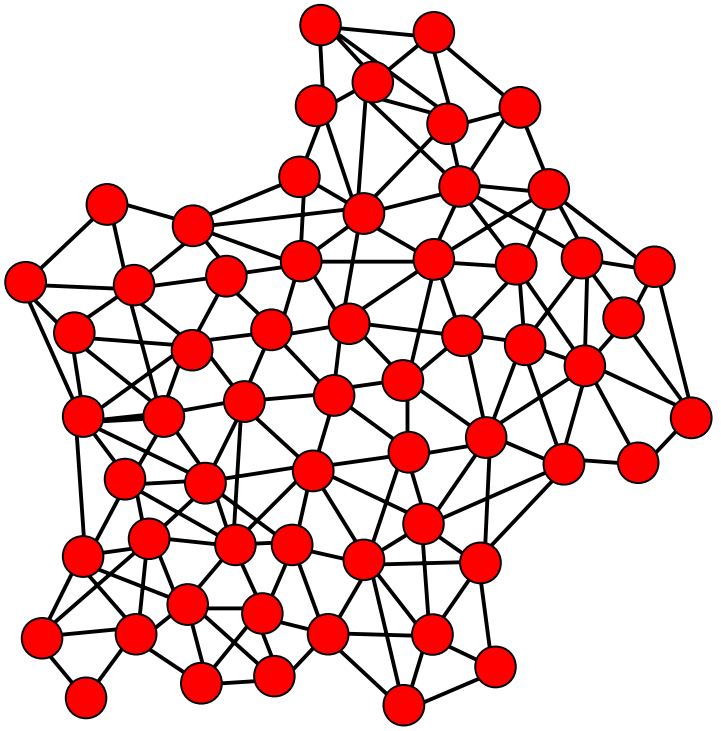
- Simplicity
- Scaling & Portability
- Robustness

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

Amorphous Medium



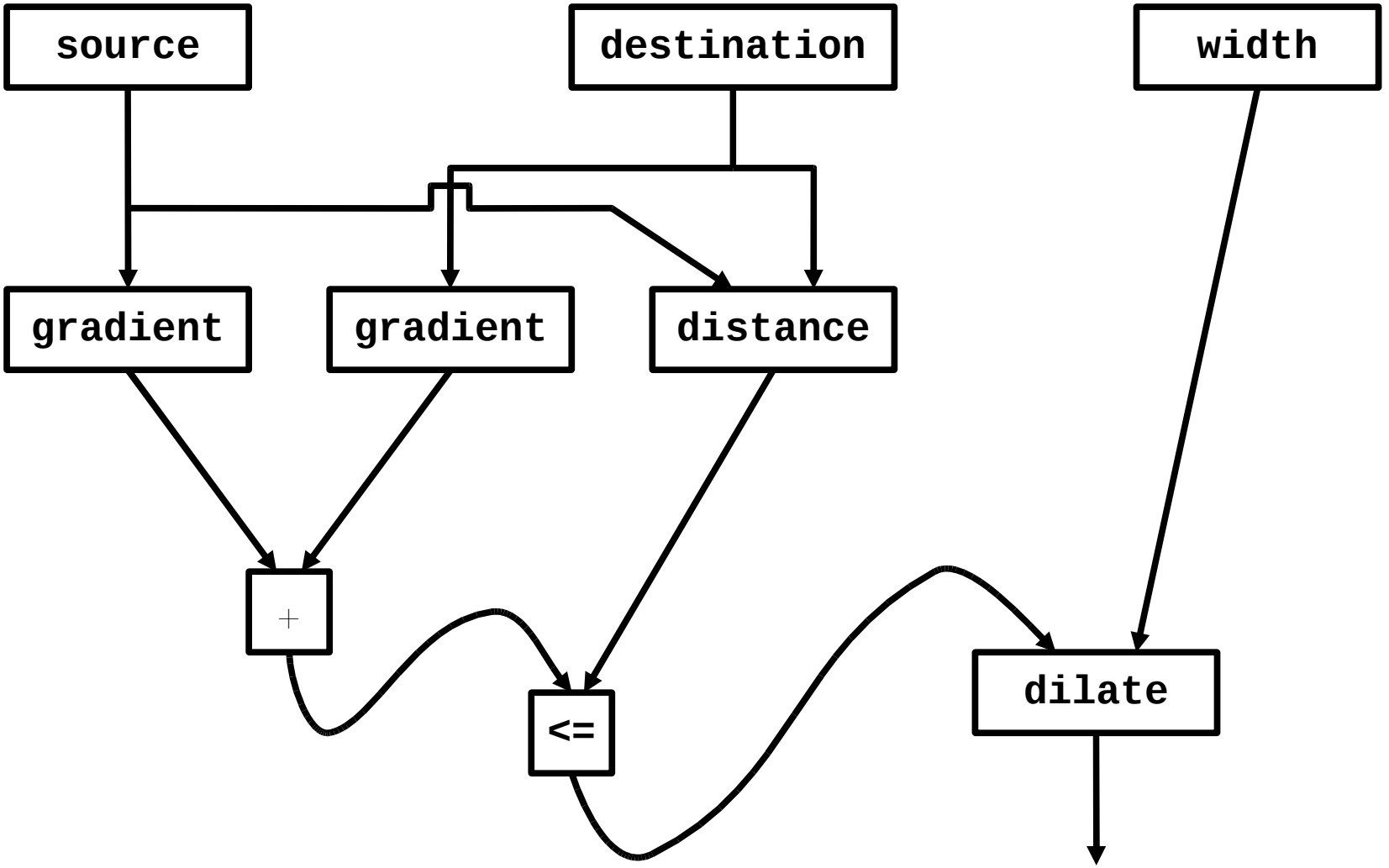
- Continuous space & time
- Infinite number of devices
- See neighbors' past state



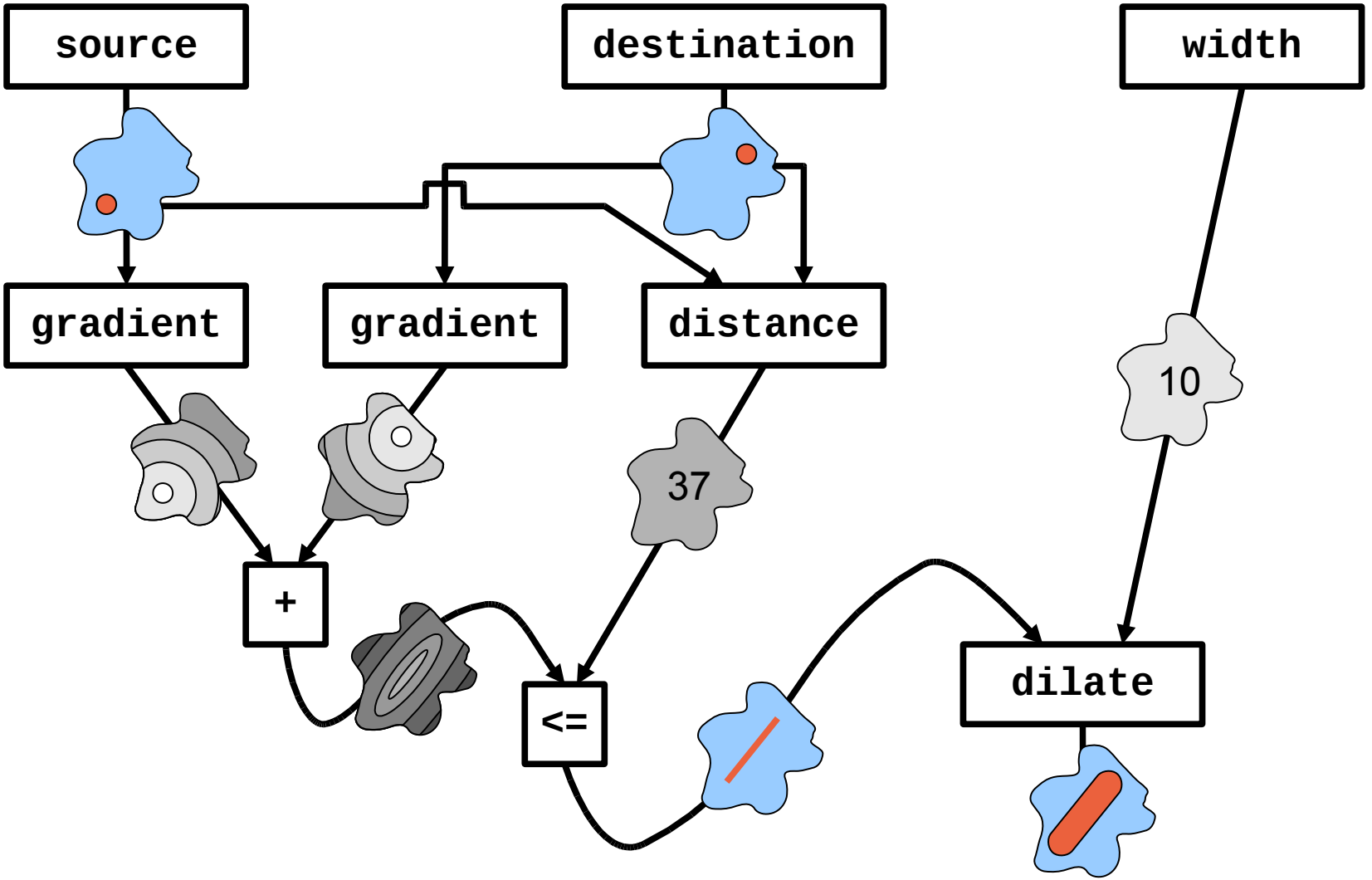
Approximate with:

- Discrete network of devices
- Signals transmit state

Computing with fields

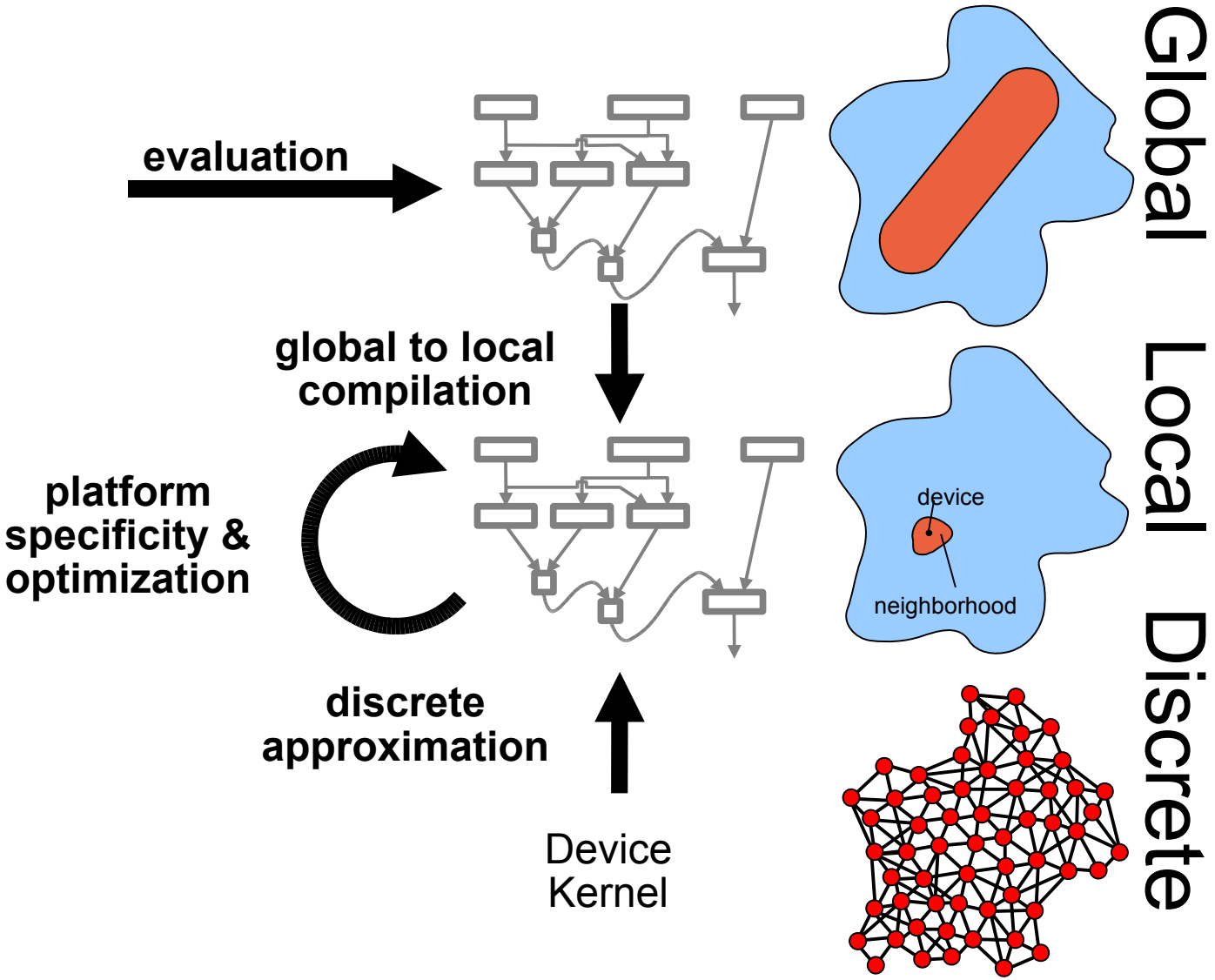


Computing with fields

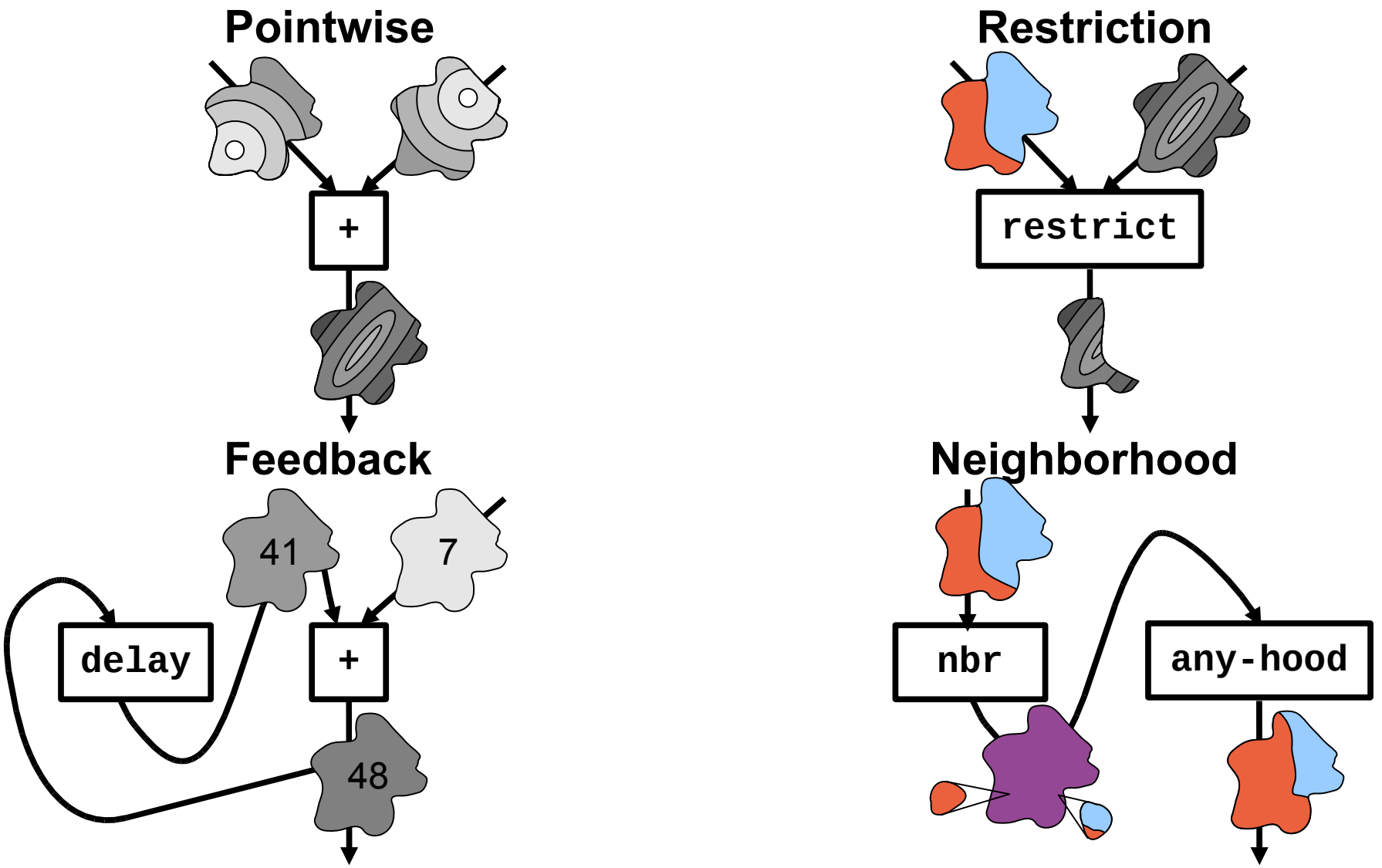


Proto

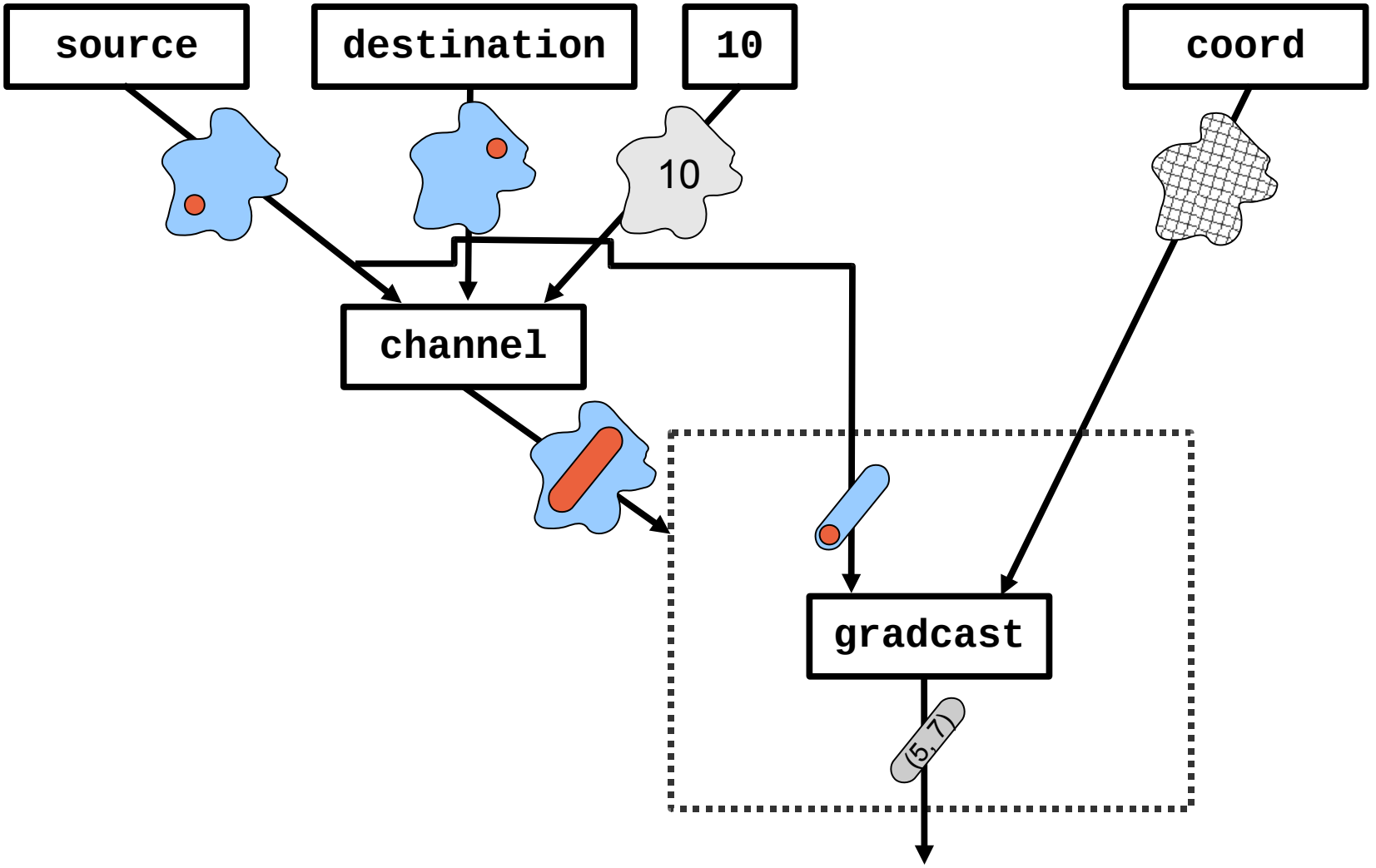
```
(def gradient (src) ...)
(def distance (src dst) ...)
(def dilate (src n)
  (<= (gradient src) n))
(def channel (src dst width)
  (let* ((d (distance src dst))
         (trail (<= (+ (gradient src)
                       (gradient dst))
                    d)))
    (dilate trail width)))
```



Proto's Families of Primitives

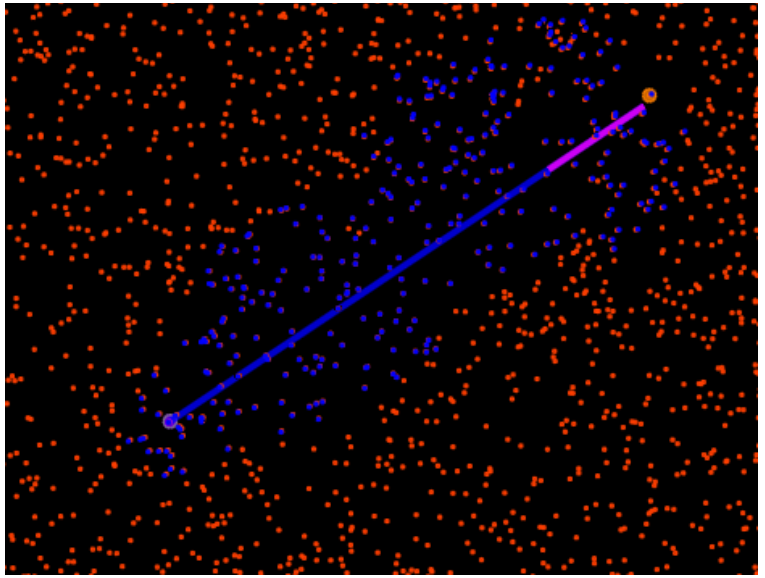


Modulation by Restriction

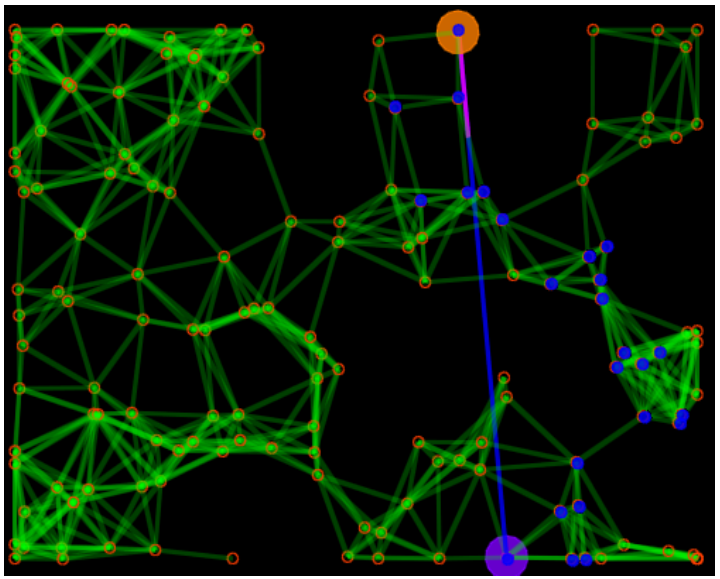


Why use continuous space?

- Simplicity
- Scaling & Portability
- Robustness



2000 devices



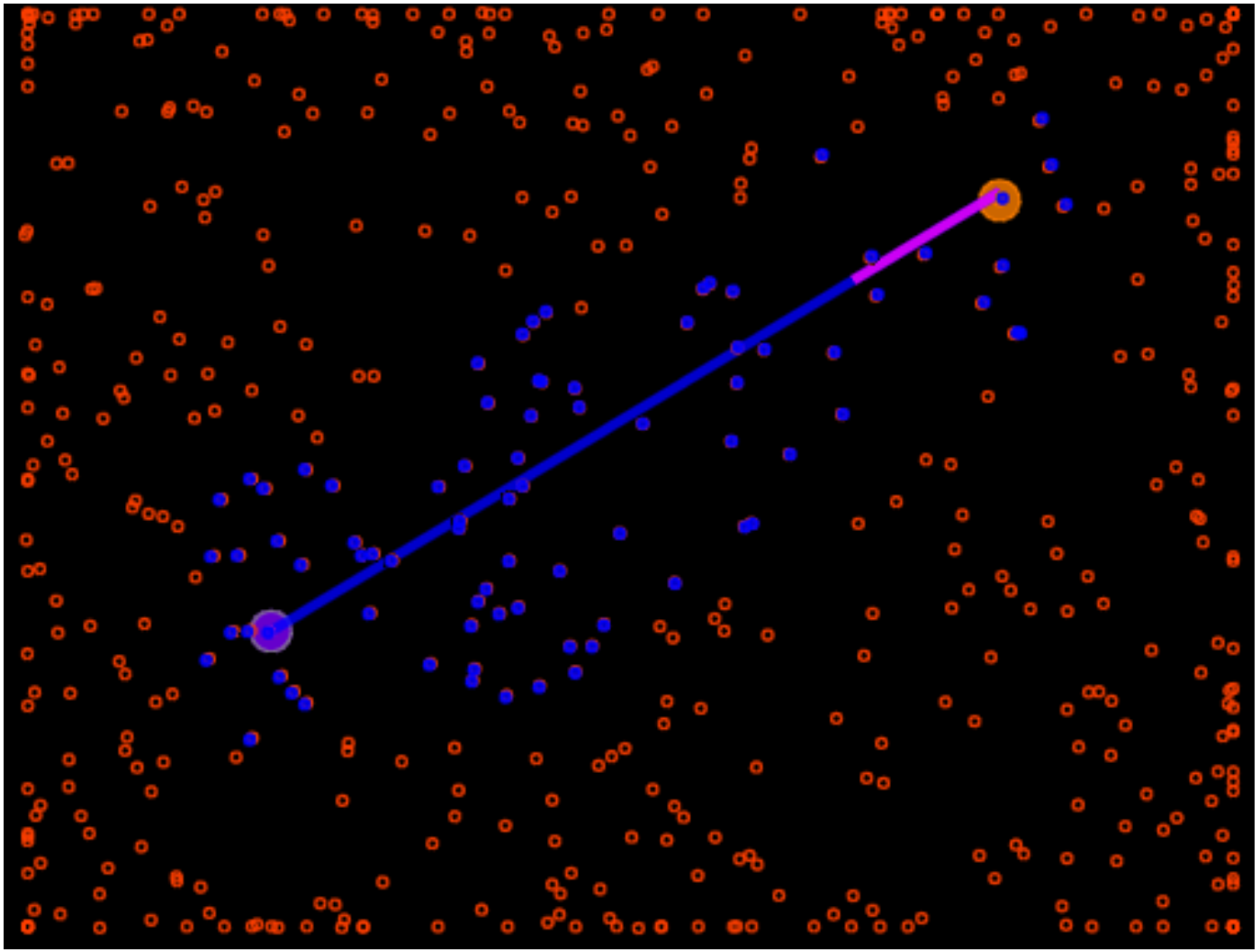
150 devices

Diving into the details

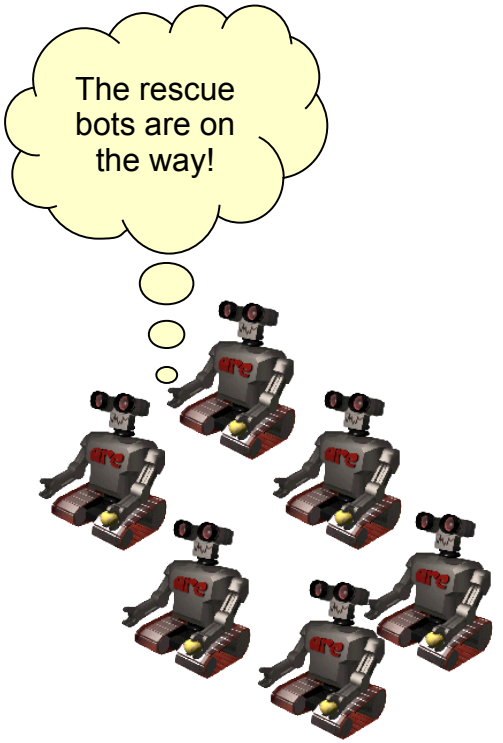
Let's build this up using the Proto simulator, one piece at a time...

(break to work w. simulator)

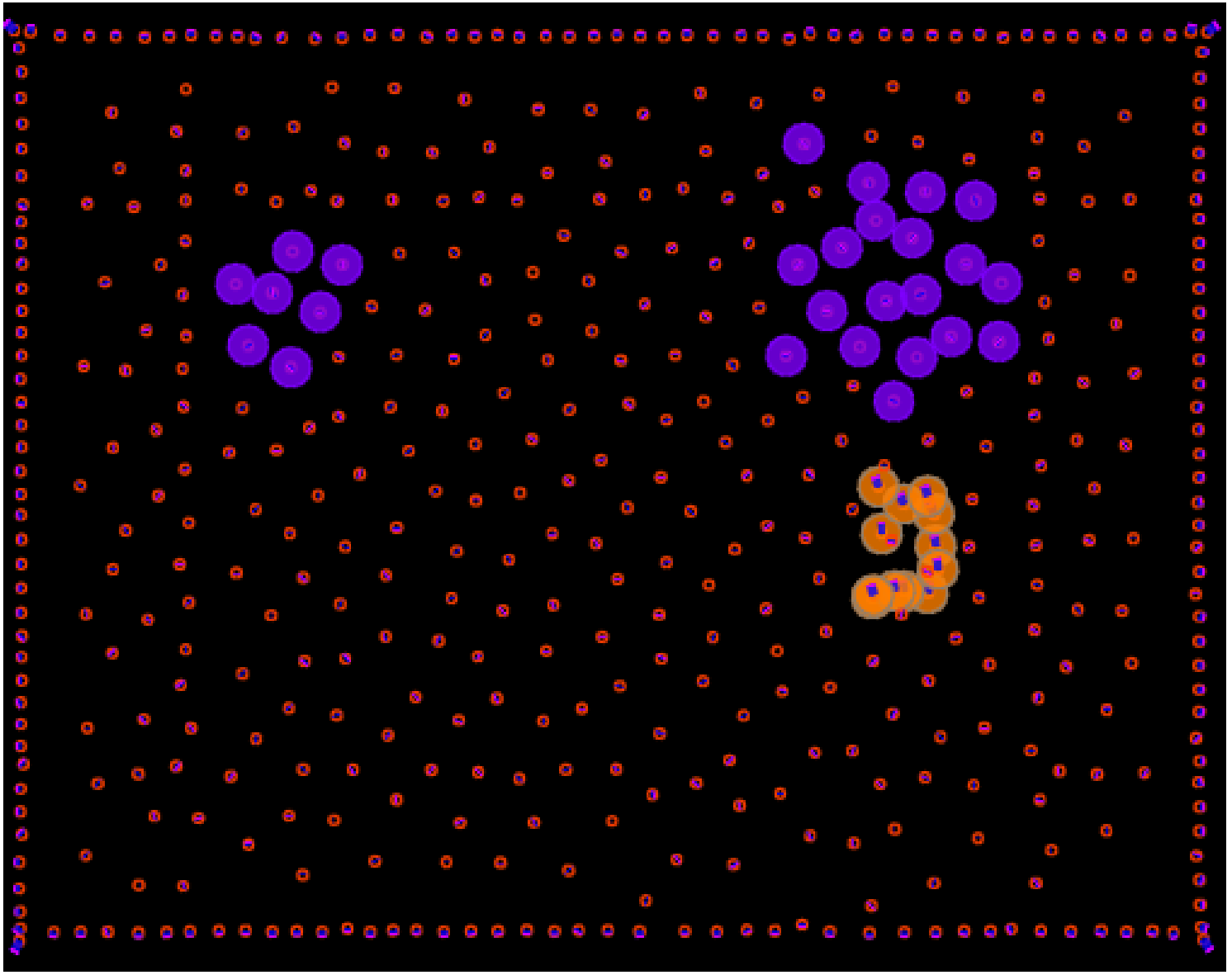
In simulation...



Example: Search & Rescue



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)