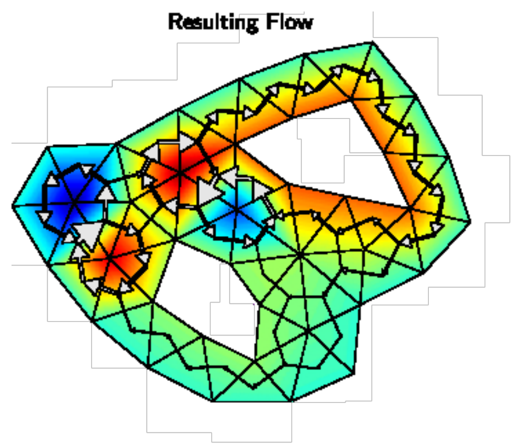
# A Fluid-Based Approach to Human-Swarm Interactions


**Magnus Egerstedt**

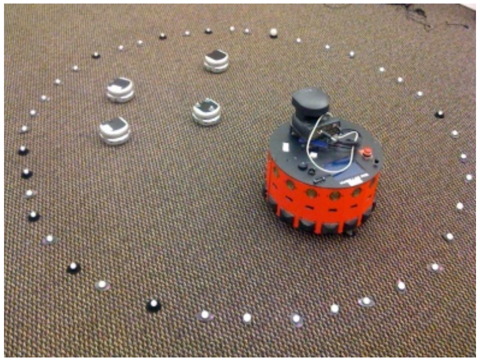
**GRITSLab**
**Electrical and Computer Engineering**
**Georgia Institute of Technology**
**www.ece.gatech.edu/~magnus**


Outline:

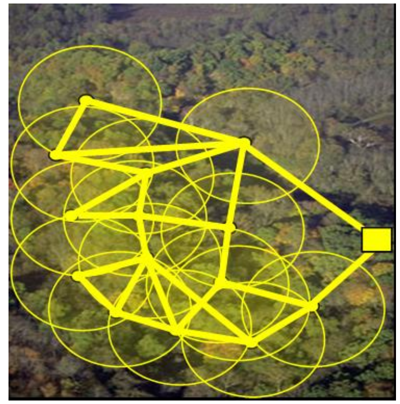
1. Leader-Based Interactions
2. Infrastructure Routing
3. Fluid-Based HSI



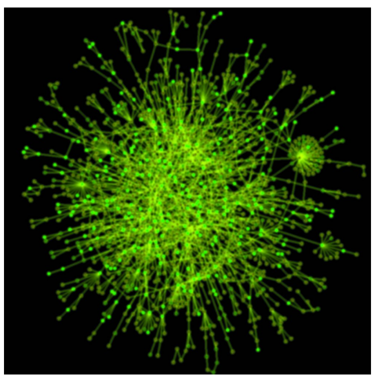
# Why Interacting with Dynamic Networks?



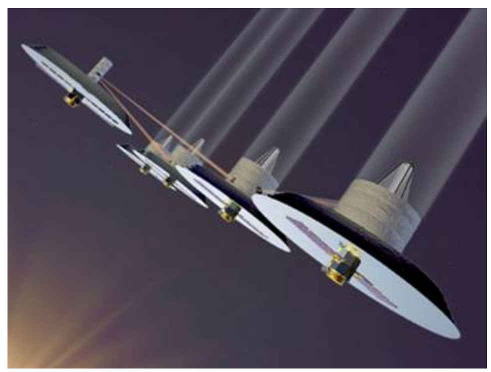
Multi-agent robotics



Sensor and communications networks



Biological networks

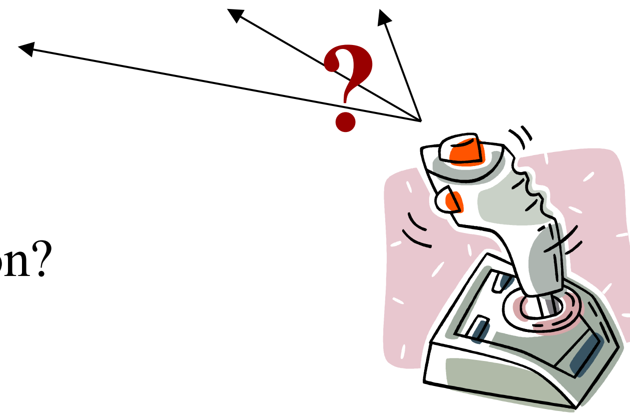


Coordinated control

# Inspiration: The Mandatory Bio-Slide

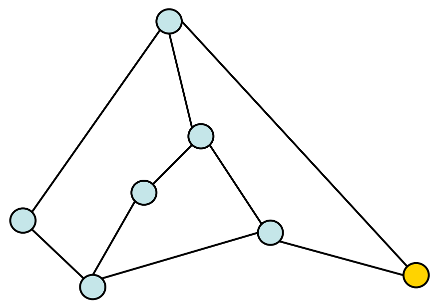
- As sensor webs, large-scale robot teams, and networked embedded devices emerge, algorithms are needed for inter-connected systems with ***limited communication, computation, and sensing capabilities***

- How to effectively control such systems?
  – What is the correct model?
  – What is the correct mode of interaction?
  – Does every individual matter?



# Standard Model: Leader (Anchor) Nodes

- **Key idea**: Let some subset of the agents act as control inputs and let the rest run some cohesion ensuring control protocol



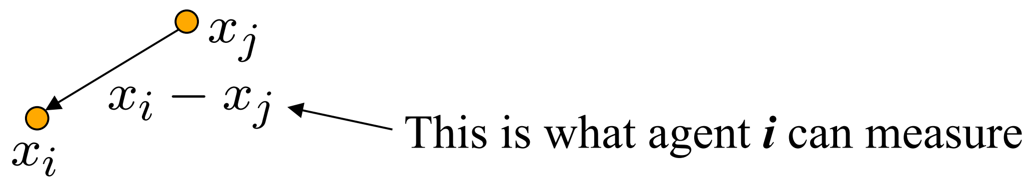
# Leader-Based Interactions

# The Agenda

- Decentralized Interactions
- Leader-Based Swarm-Interactions
- Reinterpreting the Standard Model
- Fluid-Based Swarm-Interactions

# Rendezvous – A Canonical Problem

- Given a collection of mobile agents who can only measure the relative displacement of their neighbors (no global coordinates)




- Problem: Have all the agents meet at the same (unspecified) position

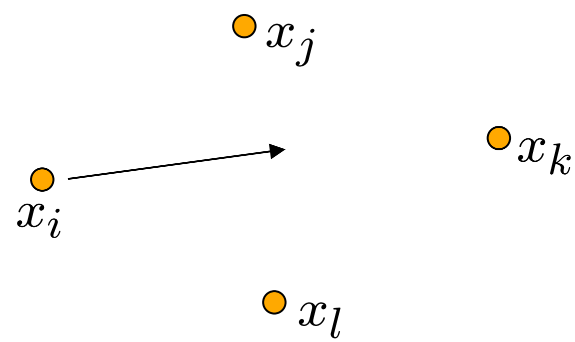
- If there are only two agents, it makes sense to have them drive towards each other, i.e.

$$\dot{x}_1 = -\gamma_1(x_1 - x_2)$$
$$\dot{x}_2 = -\gamma_2(x_2 - x_1)$$

- If $\gamma_1 = \gamma_2$ they should meet halfway

# Rendezvous – A Canonical Problem

- If there are more than two agents, they should probably aim towards the centroid of their neighbors (or something similar)




$$\dot{x}_i = -\gamma \sum_{j \in \mathcal{N}_i} (x_i - x_j)$$

**Fact**: The ***consensus equation*** drives all agents to the same state value iff the interaction network is connected

$$\lim_{t \to \infty} x_i(t) = \bar{x} = \frac{1}{N} \sum_{j=1}^{N} x_j(0)$$

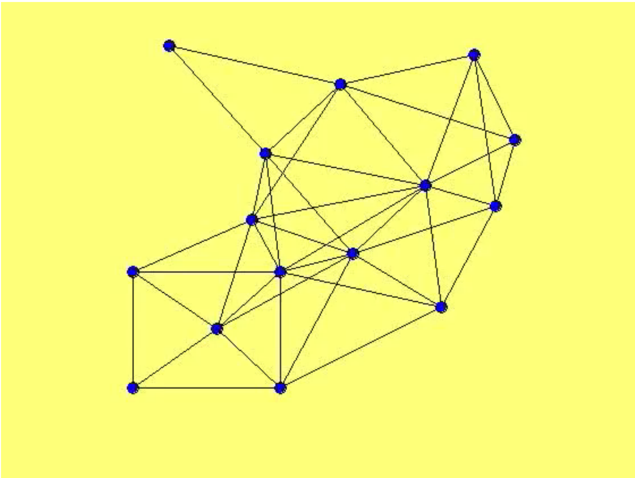
# Beyond Static Consensus

- The consensus equation will drive the node states to the same value if the graph is static and connected.
- But, this is clearly not the case in a number of situations:
  - **Edges = communication links**
    - Random failures
    - Dependence on the position (shadowing,…)
    - Interference
    - Bandwidth issues
  - **Edges = sensing**
    - Range-limited sensors
    - Occlusions
    - Weirdly shaped sensing regions

# Switched Consensus

**Theorem**: As long as the graph stays connected, the ***consensus equation*** drives all agents to the same state value

$$\lim_{t\to\infty} x_i(t) = \bar{x} = \frac{1}{N}\sum_{j=1}^{N} x_j(0)$$

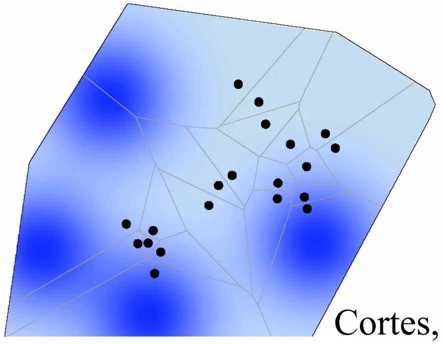


# Adding Weights

- Sometimes it makes sense to add weights

$$\dot{x}_i = -\sum_{j \in N_i} w(\|x_i - x_j\|)(x_i - x_j)$$

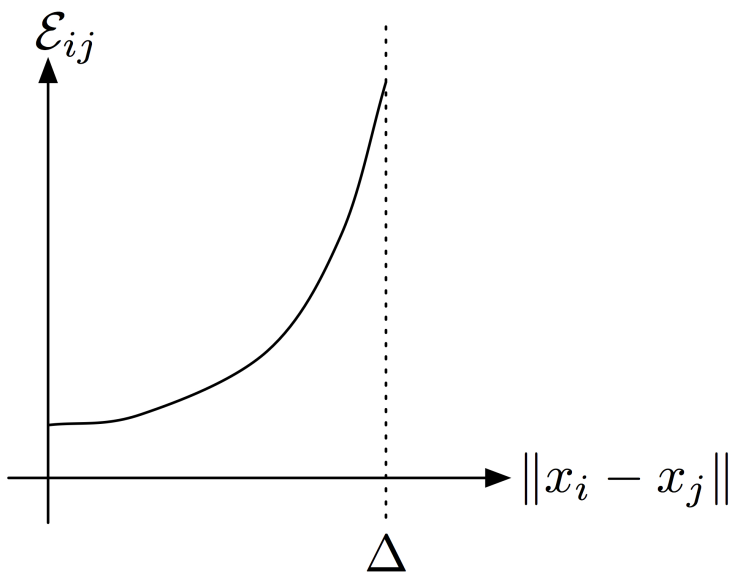
- Collision avoidance
- Coverage
- Connectivity maintenance



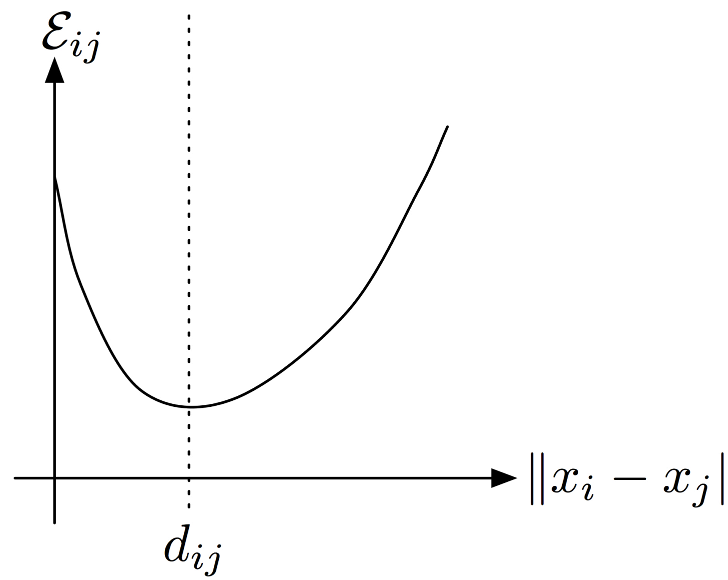
Cortes, Martinez, Bullo

# Weights Through Edge Tensions

- How select appropriate weights?
- Let an edge tension be given by $\mathcal{E} = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{i,j} \mathcal{E}_{i,j}(\|x_i - x_j\|)$



Connectivity Maintenance



Formation Control

# Weights Through Edge Tensions

- How select appropriate weights?
- Let an edge tension be given by $\mathcal{E} = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{i,j} \mathcal{E}_{i,j}(\|x_i - x_j\|)$
- We get

$$\frac{\partial \mathcal{E}_{i,j}}{\partial x_i} = w_{i,j}(\|x_i - x_j\|)(x_i - x_j)$$

- Gradient descent

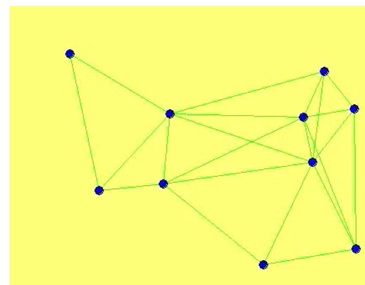
$$\dot{x}_i = -\frac{\partial \mathcal{E}}{\partial x_i} = -\sum_{j \in N_i} w_{i,j}(\|x_i - x_j\|)(x_i - x_j)$$

$$\frac{d\mathcal{E}}{dt} = \frac{\partial \mathcal{E}}{\partial x}\dot{x} = -\left\| \frac{\partial \mathcal{E}}{\partial x} \right\|^2$$

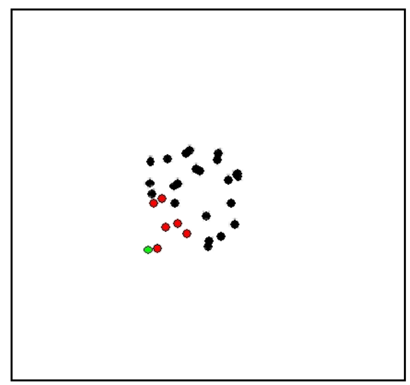
***Energy is non-increasing!***

# Graph-Based Control

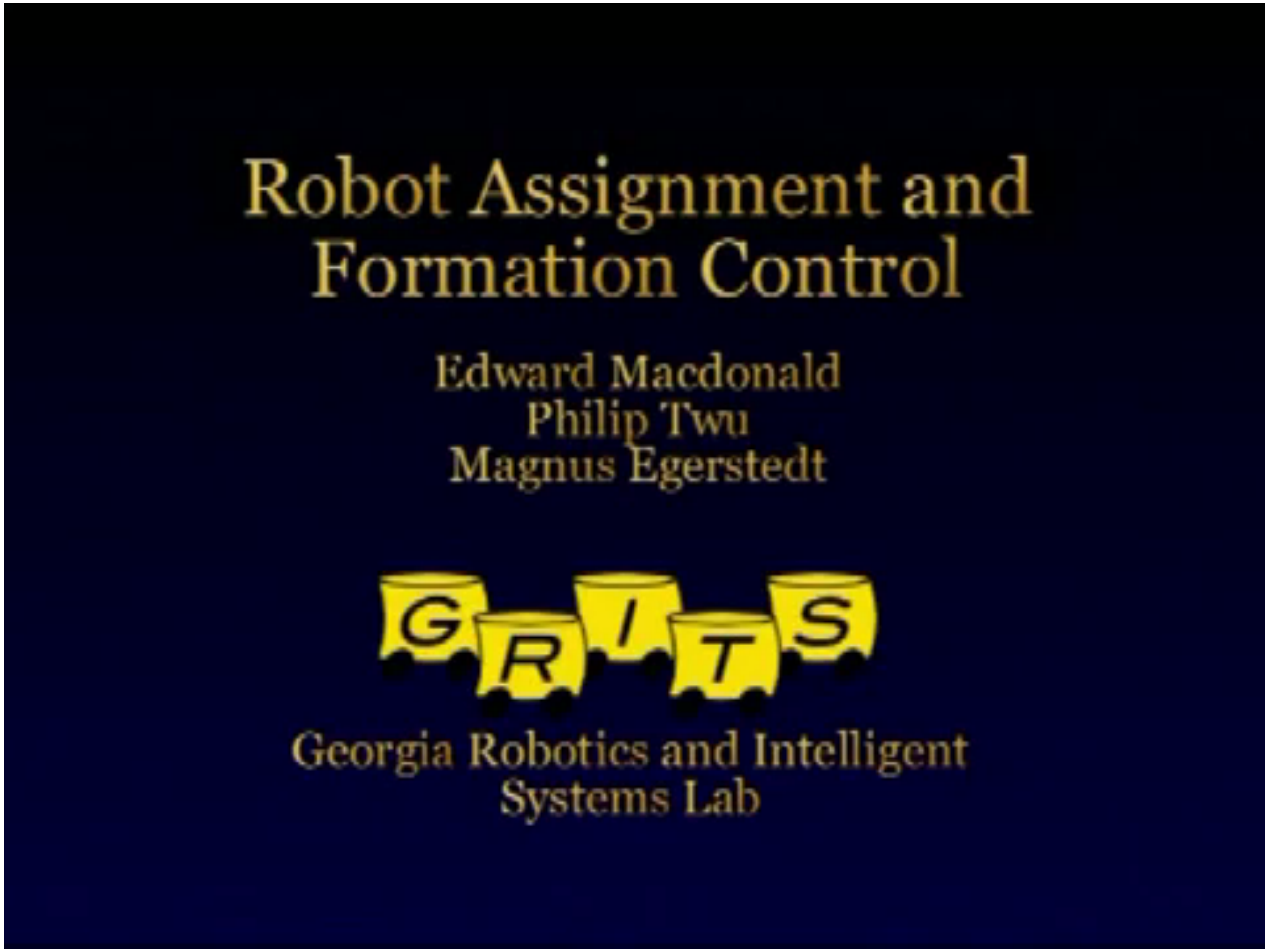
- In fact, based on variations of the consensus equation, a number of different multi-agent problems have been "solved", e.g.
  – **Formation control** (How drive the collection to a predetermined configuration?)
  – **Coverage control** (How produce triangulations or other regular structures?)



- ***OK – fine. Now what?***
- Need to be able to **reprogram and redeploy** multi-agent systems (**HSI = Human-Swarm Interactions**)
- This has traditionally been achieved through active control of the "leader-nodes"



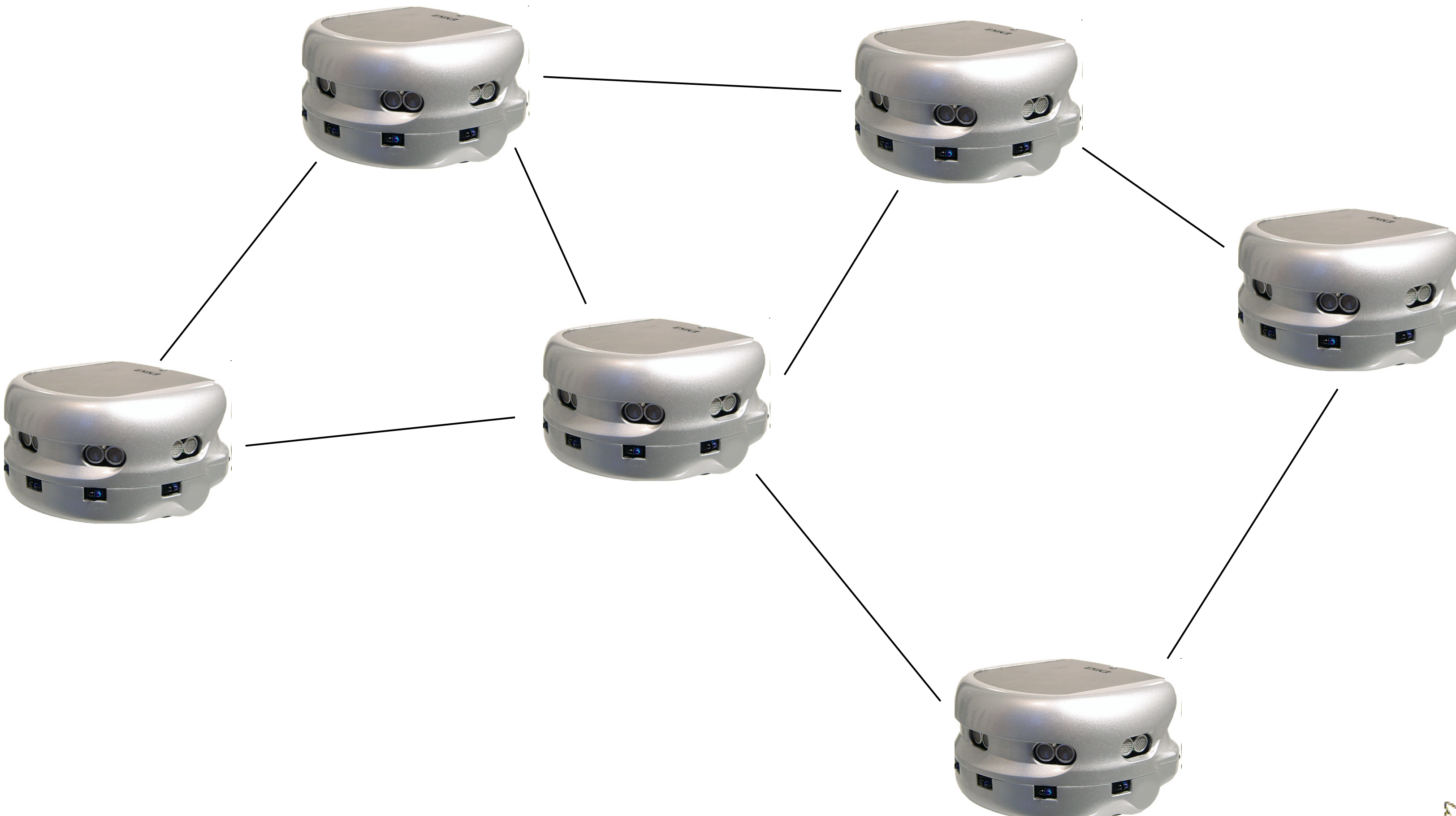
# Heterogeneous Networks



# But, What About Other Types of Interactions?

## Fluid-Based Interactions?!?

# Multiple Robots…

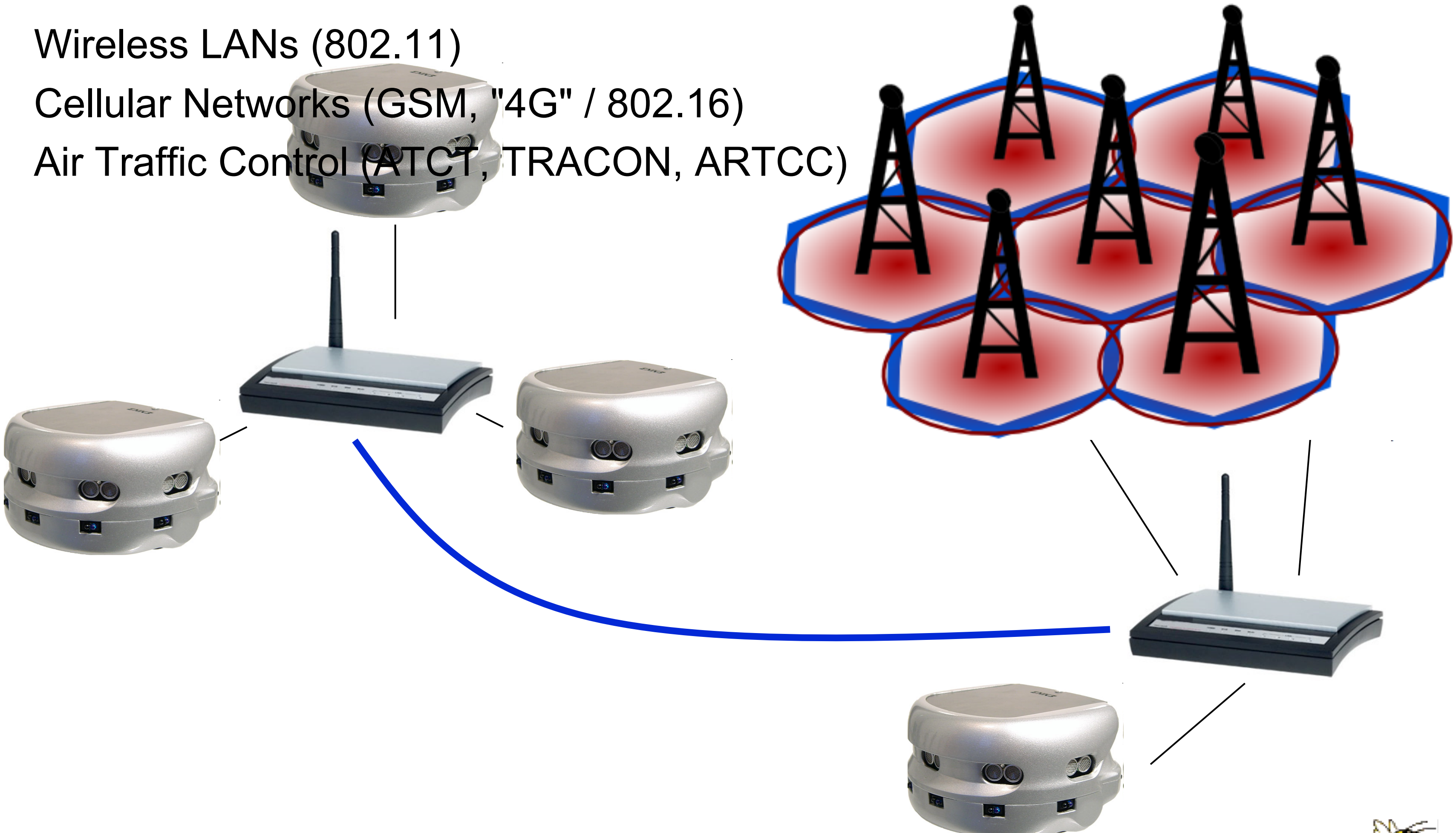


# …With Infrastructure

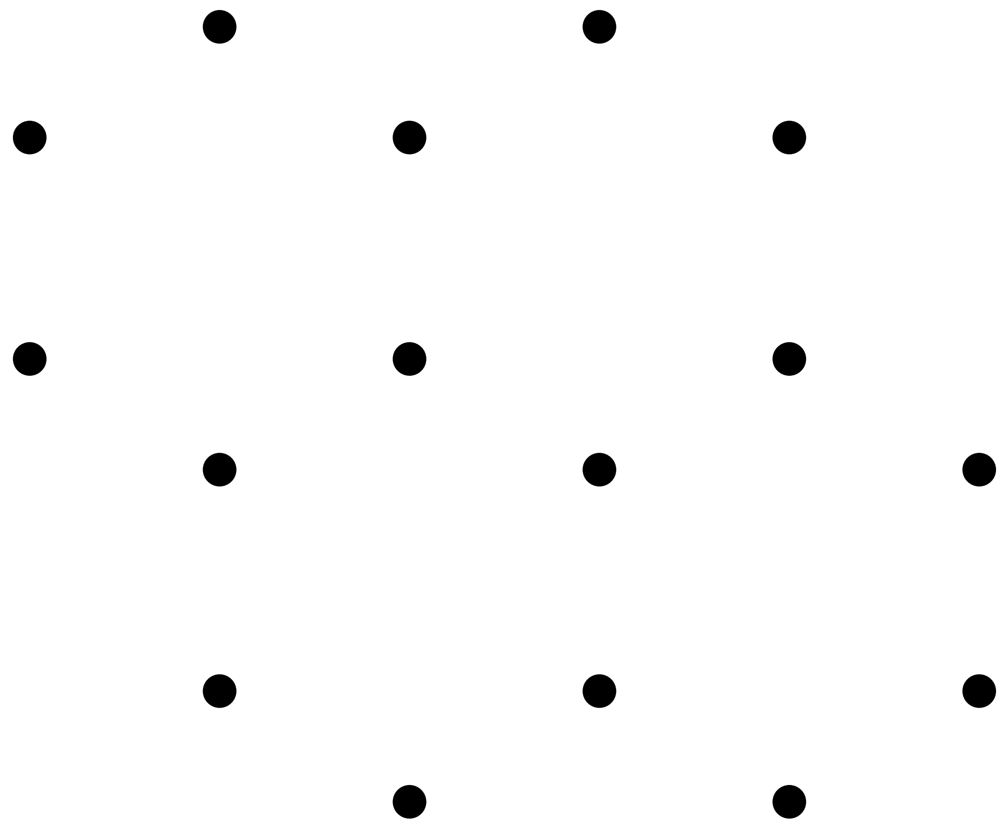
Wireless LANs (802.11)

Cellular Networks (GSM, "4G" / 802.16)

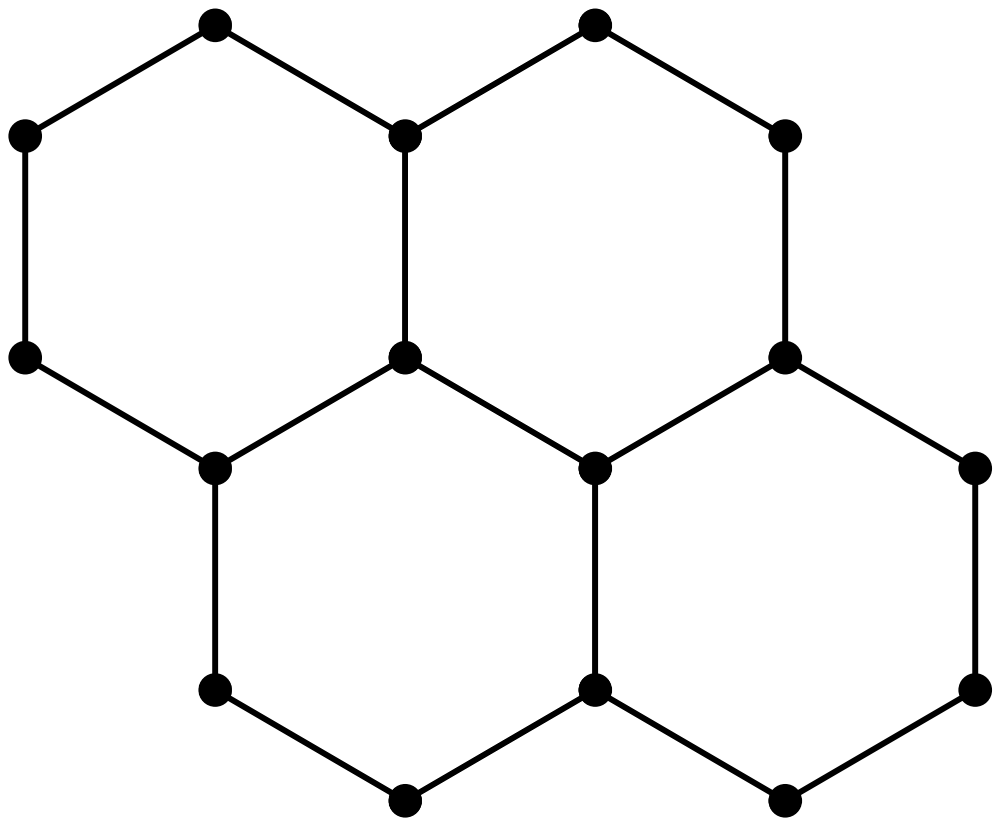
Air Traffic Control (ATCT, TRACON, ARTCC)



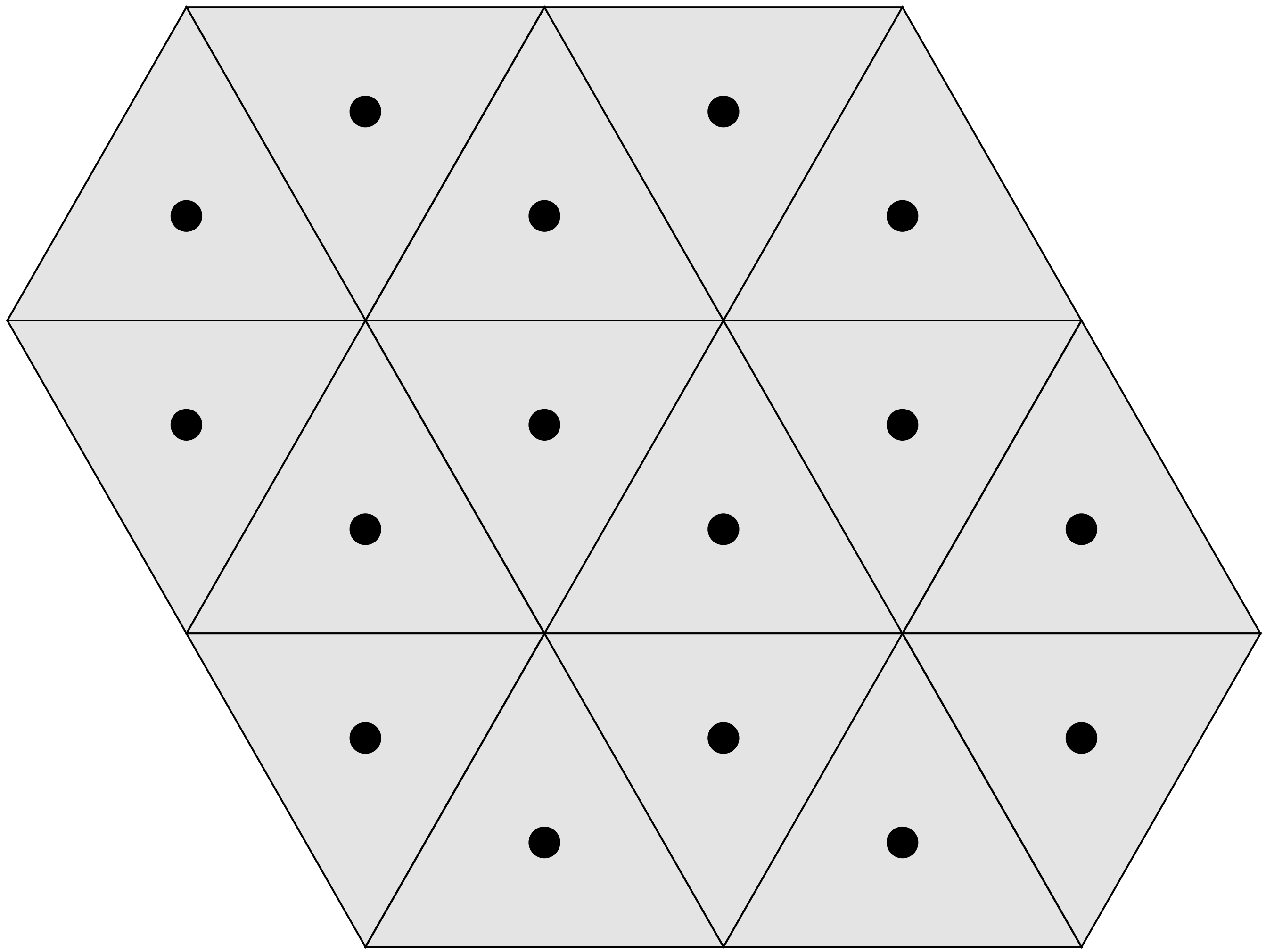
# The Infrastructure Network



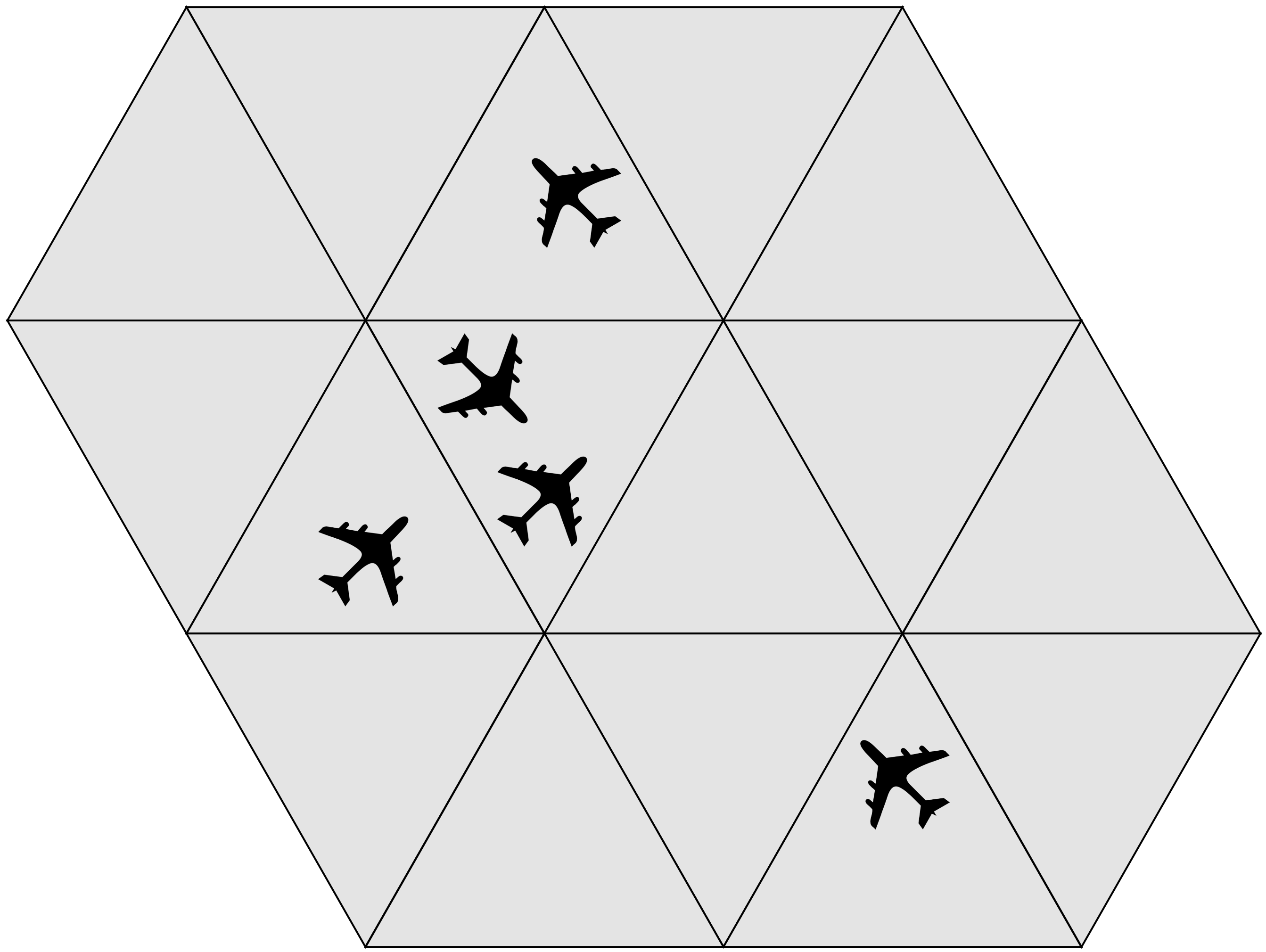
# The Infrastructure Network



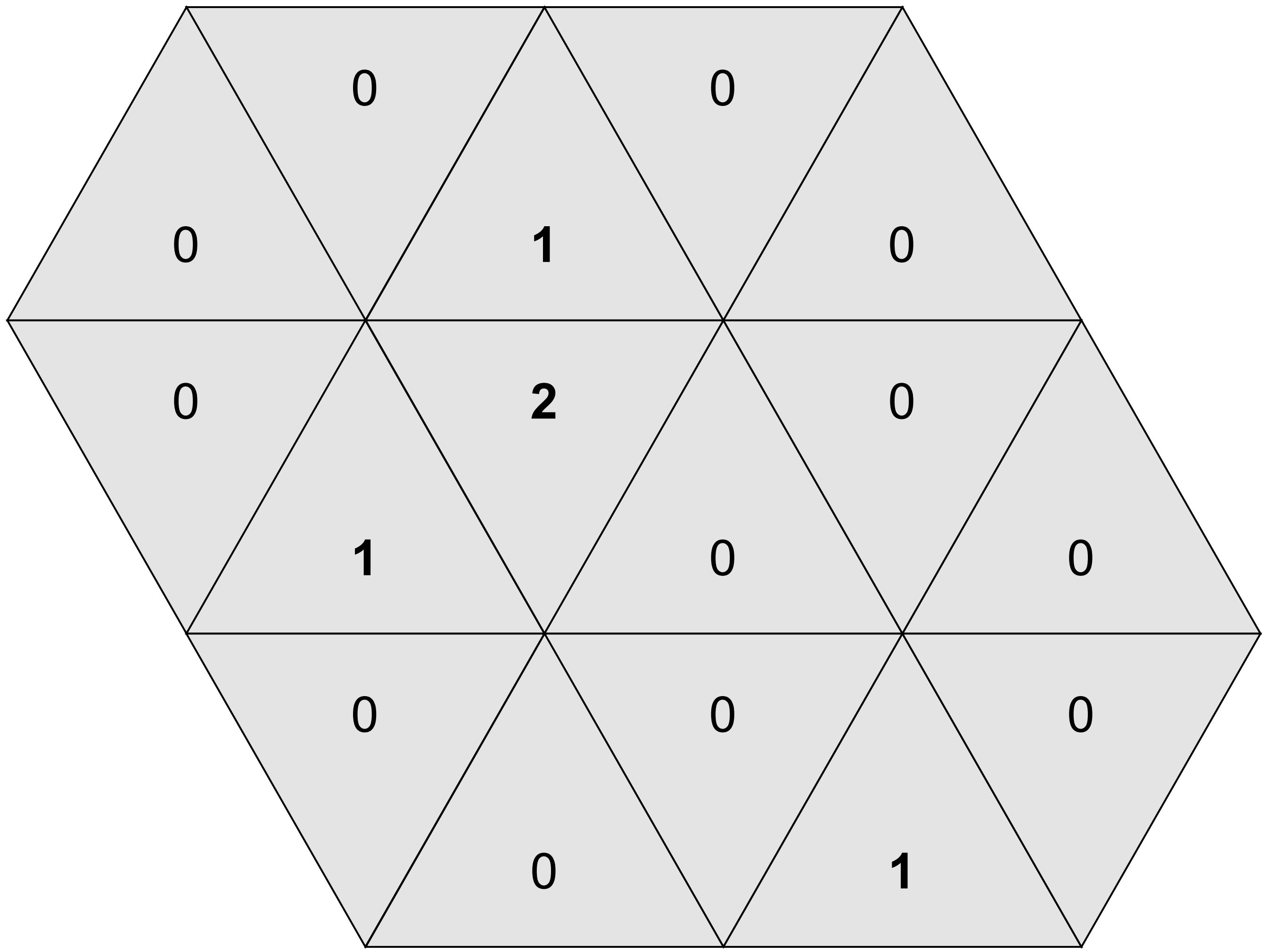
# The Infrastructure Network



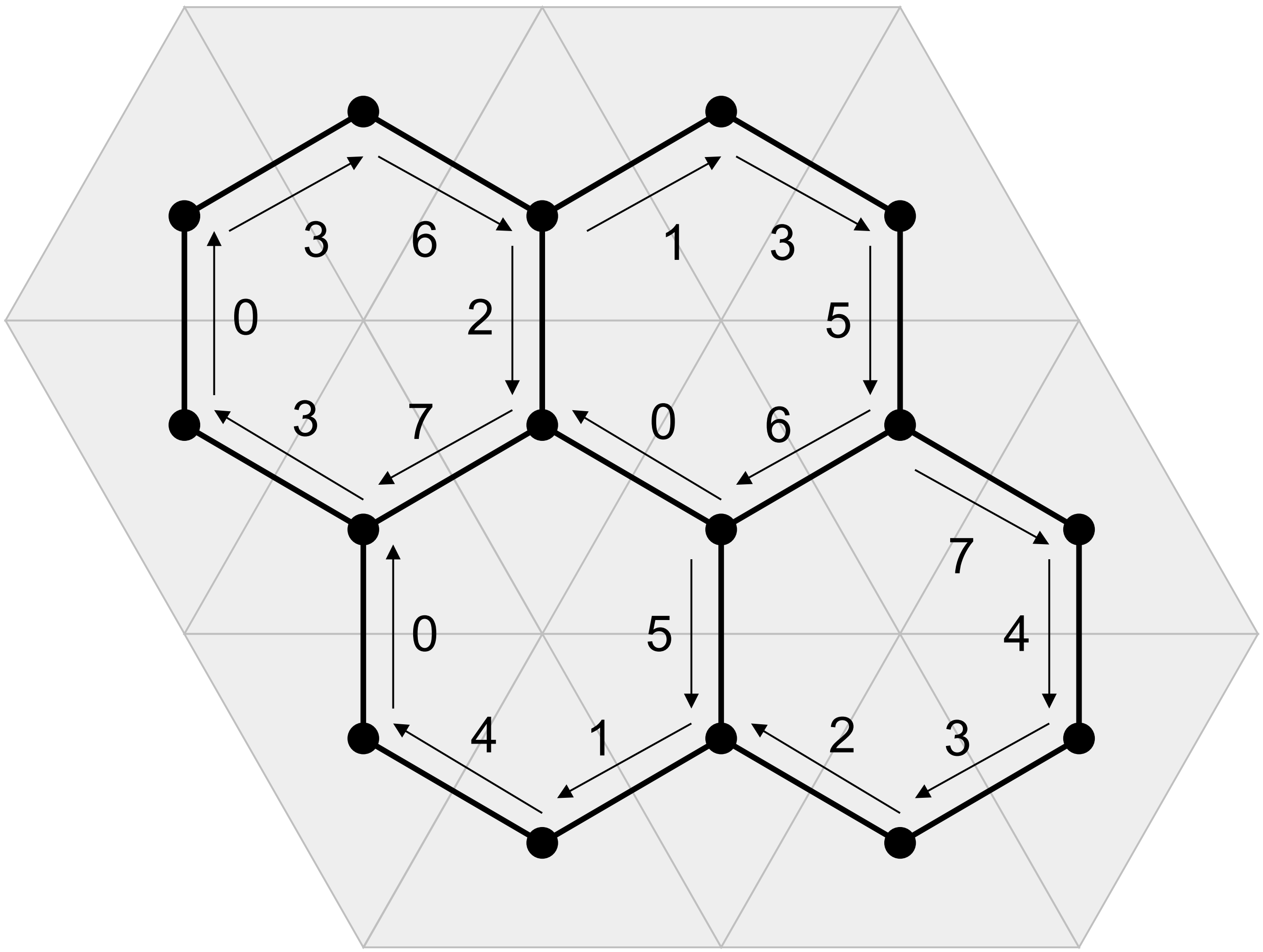
# The Infrastructure Network



# The Infrastructure Network



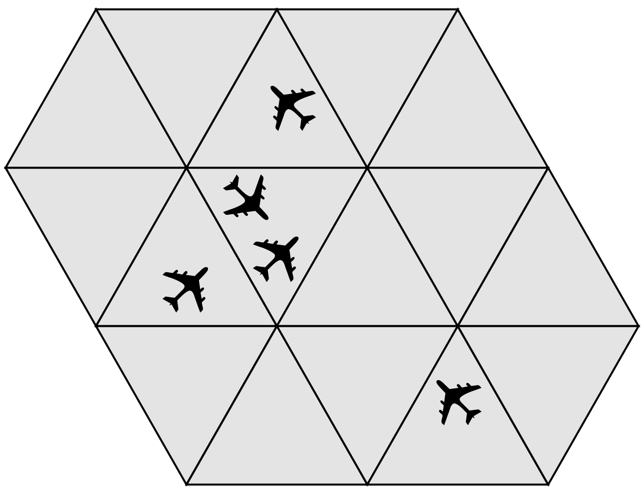
# The Infrastructure Network



# Two Views of the World

- Lagrangian

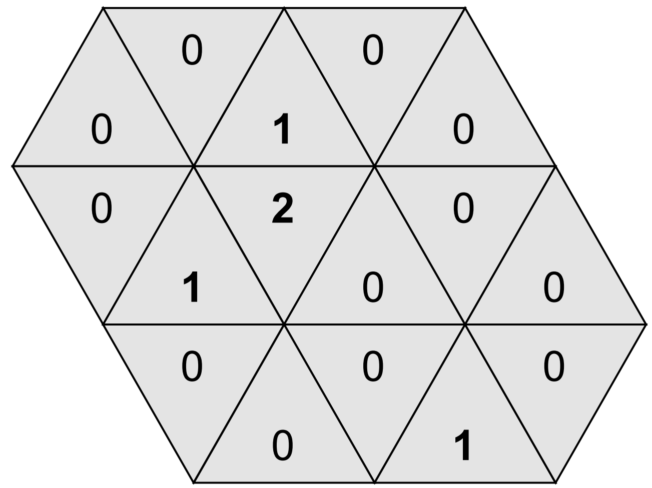
$$\dot{x}_i = f(x_i, u_i)$$



- Eulerian

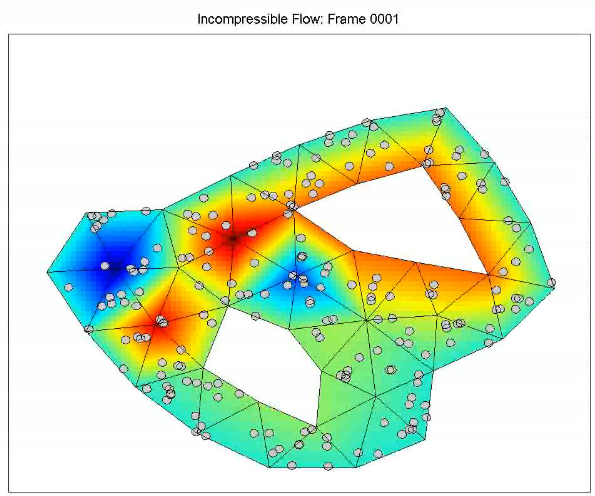
$$\dot{m}_i = v_{ij}$$
$$\dot{m}_j = -v_{ij}$$

(incompressibility)

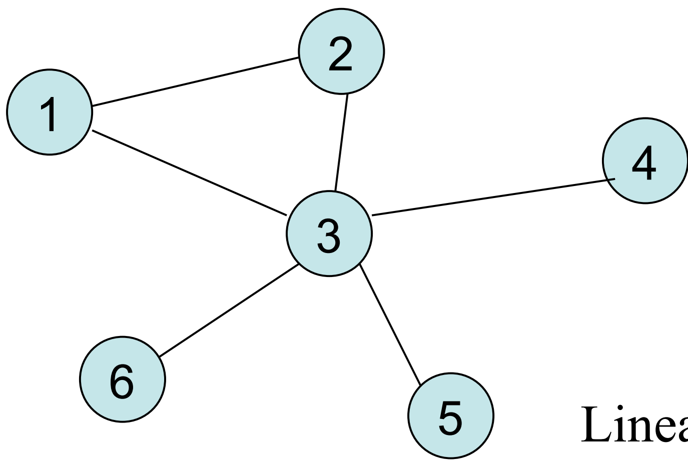


# What We'll Do…

- Let users specify "flows" through the network
- Distribute the flows across the network so vehicles don't "pile up" anywhere
  - by solving a problem on the dual graph
  - in a distributed way.
- Produce, from these flows, continuous control laws
  - "no piling up"
  - collision avoidance
  - in a distributed way.



# Back to Basics: Controlled Laplacian Dynamics




Dynamics

$$\dot{x}_i = -\sum_{j \in \mathcal{N}(i)} (x_i - x_j) + u_i$$

Linear system

$$\dot{x} = -Lx + u$$

????

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}$$

$$u = \begin{bmatrix} u_1 \\ \vdots \\ u_N \end{bmatrix}$$

where $L$, the *Graph Laplacian*, is defined s.t.,

$$L_{ij} = \begin{cases} \deg(i) & i = j \\ -1 & j \in \mathcal{N}(i) \\ 0 & \text{o.w.} \end{cases}$$

# Graph Laplacian

Laplacian factors as…

$$L = DD^T \quad (\nabla = \text{div grad})$$

where,

$$D = \begin{array}{cc} \text{\# edges} & \\ \begin{bmatrix} 0 & 1 & & \\ 1 & -1 & & \\ 0 & 0 & \cdots & \\ -1 & 0 & & \\ 0 & 0 & & \end{bmatrix} & \text{\# vertices} \end{array}$$

# A Simple Problem: Least Squares

$$\begin{bmatrix} \\ & & A & & \\ \\ \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} \\ b \\ \\ \end{bmatrix} \qquad \min_x \|Ax - b\|^2$$

- Just make sure inner products w. columns of *A* are right…

$$\underbrace{A^T A}_{\text{Grammian of columns of A}} x = A^T b$$

# Another Grammian

- Graph Laplacian: $L = DD^T$

- Get the least-squares solution to

$$D^T p = f \qquad (\min_p \|D^T p - f\|^2)$$

by solving $\boxed{Lp = Df}$

- Gradient descent:

The input!!

$$\dot{p} = -\frac{d}{dp}\left(\frac{1}{2}\|D^T p - f\|^2\right)^T = -Lp + Df$$

# Punchline

- The forced consensus dynamics

$$\dot{p} = -Lp + Df$$

…solve the normal equations

$$Lp = Df$$

# What Does This Mean?

$$D^T p = f$$

Computes differences across edges

**Gradient**

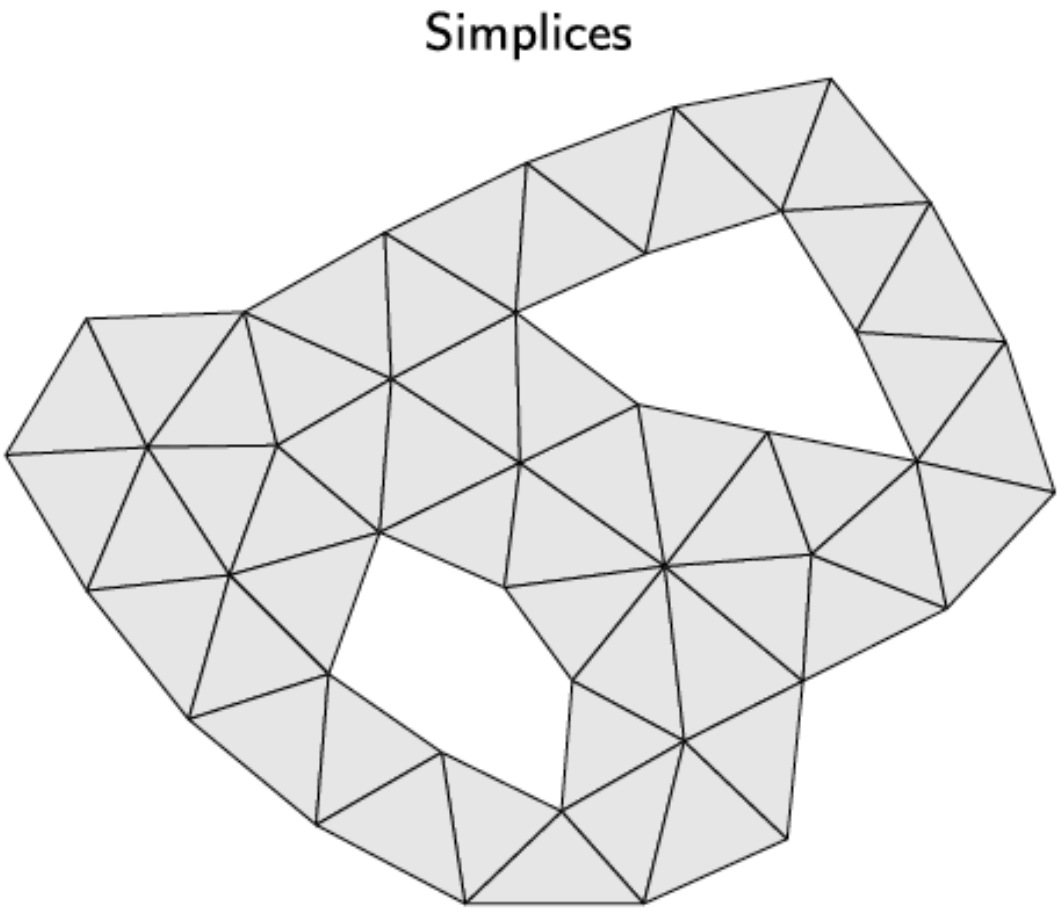
Assigns number to each vertex

**Scalar Field (PRESSURE)**

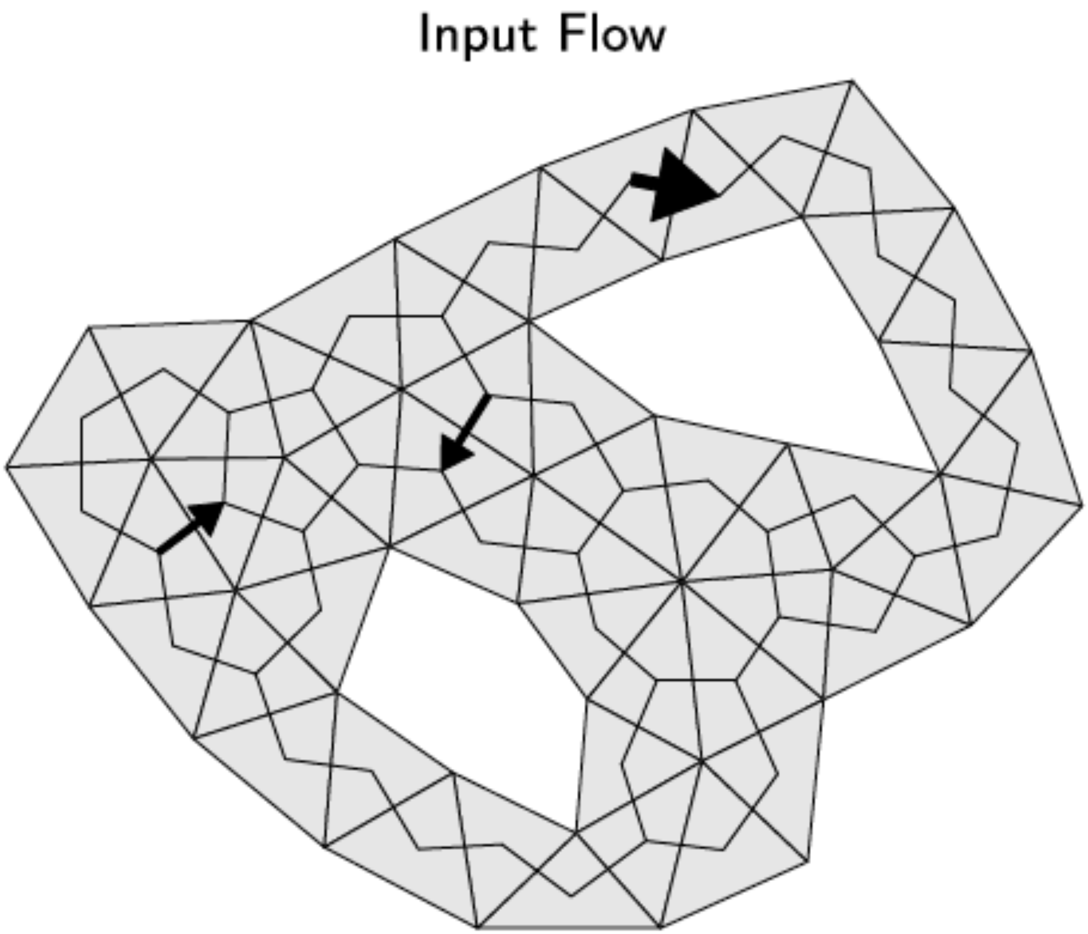
Assigns number to each edge

**Vector Field (FLOW)**

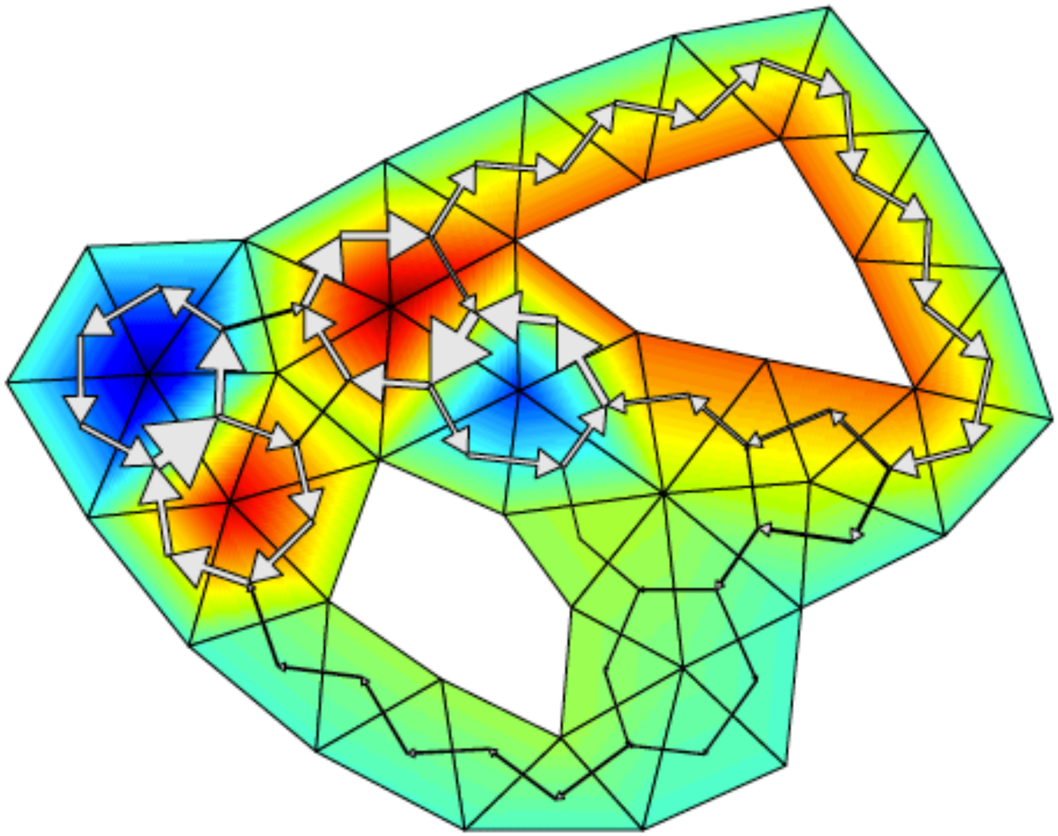
# Example



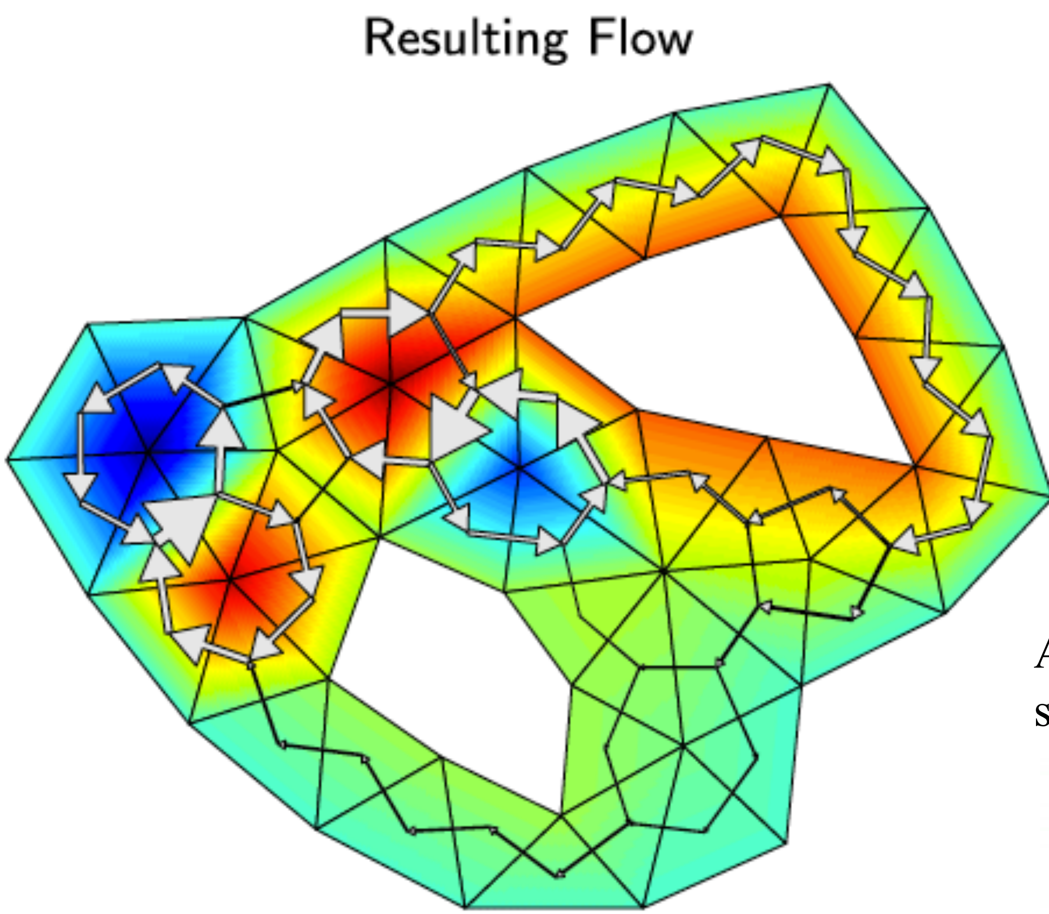
# Example



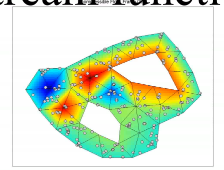
# Example



# Example



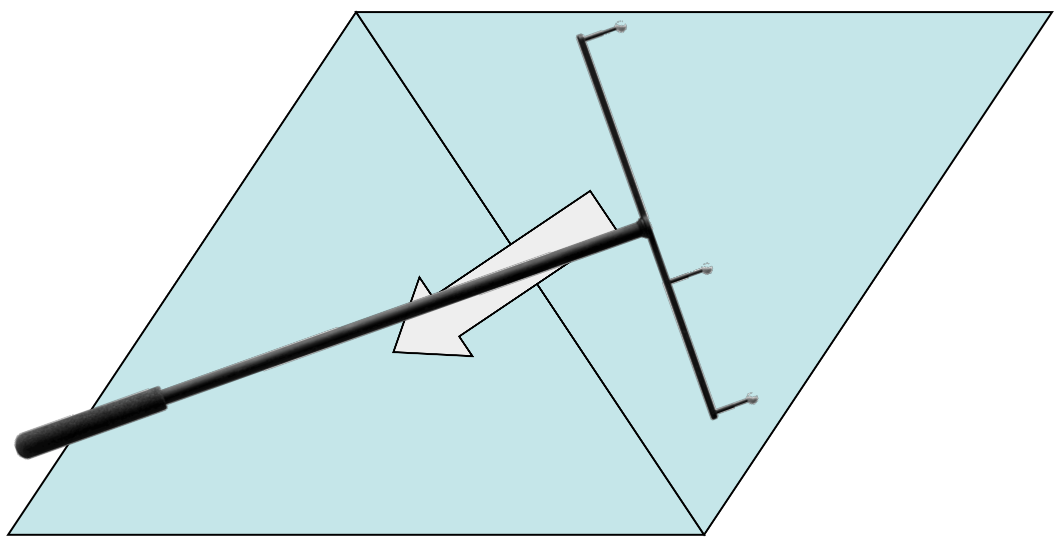

Add local, hybrid stream functions



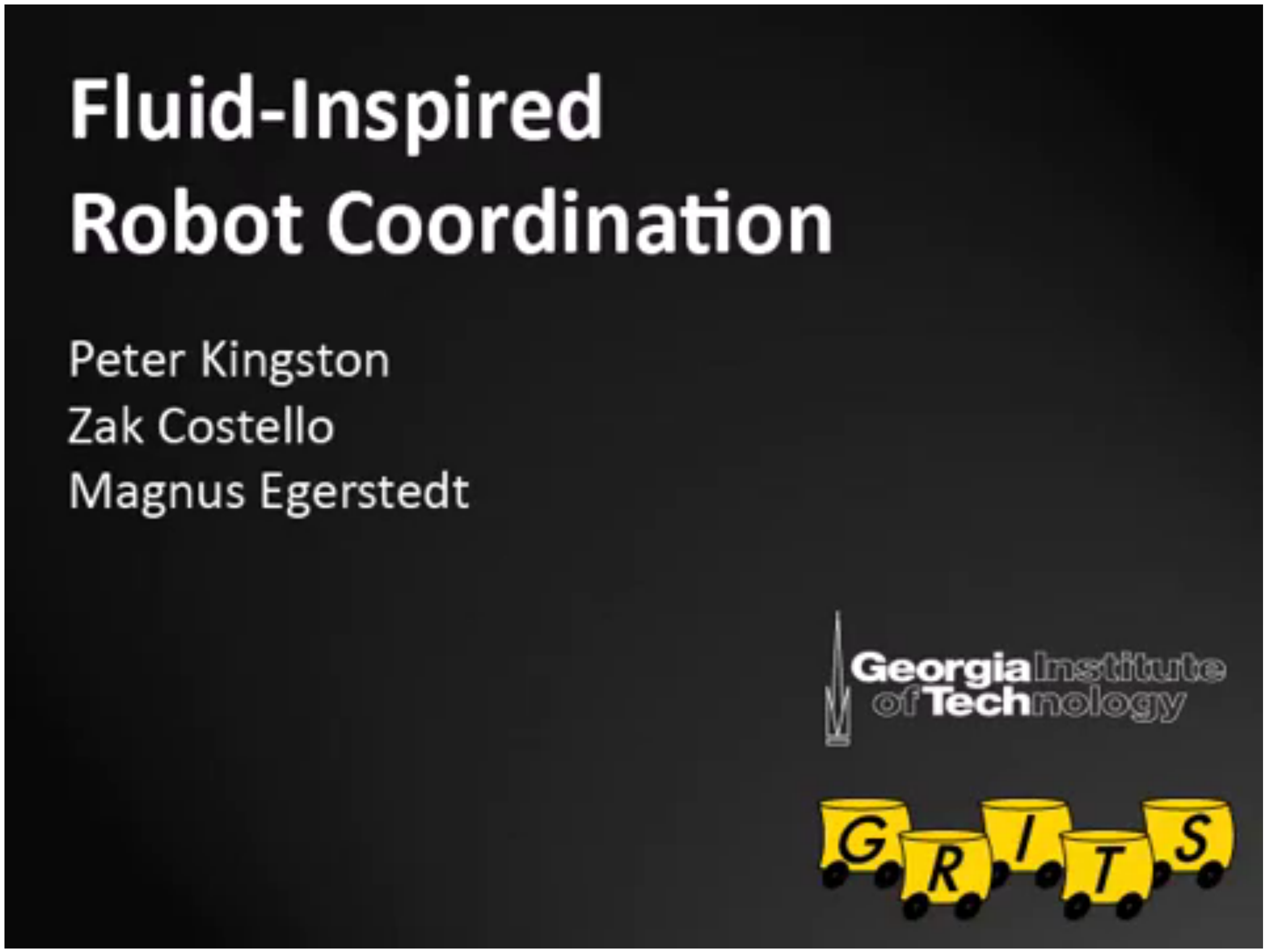
# But, What About This Picture?

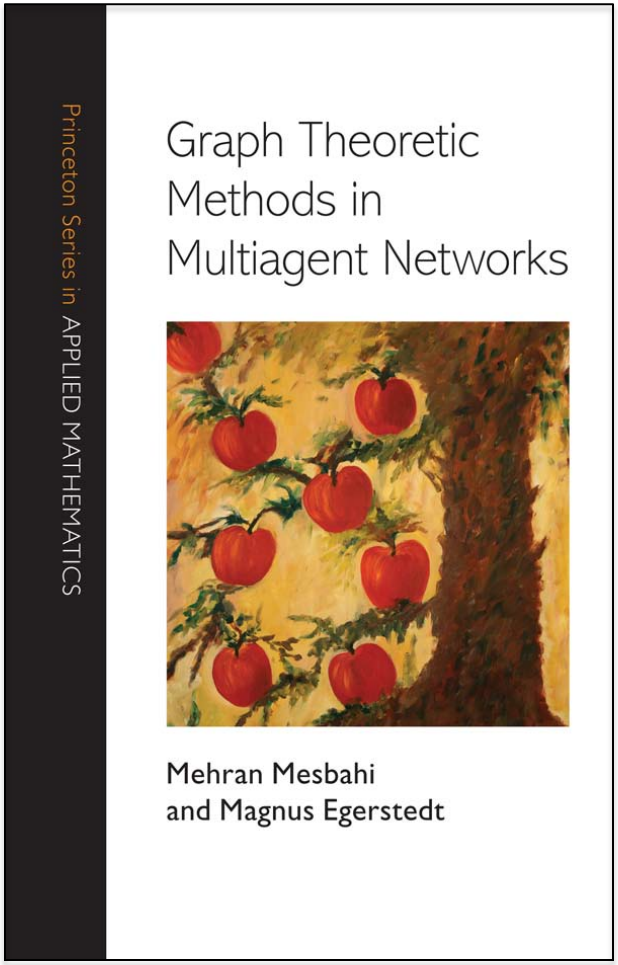
# Swarm Conducting

- Interface: Motion capture wand



# Swarm Conducting






# THANK YOU!

Peter Kingston