Stabilizing Robotics Programming Language

Adam Zimmerman

Overview

- Overview of the StarL Framework
- Distributed Path Planning (DPP) Algorithm
- Application using DPP
- DPP Properties and Verification

StarL Framework

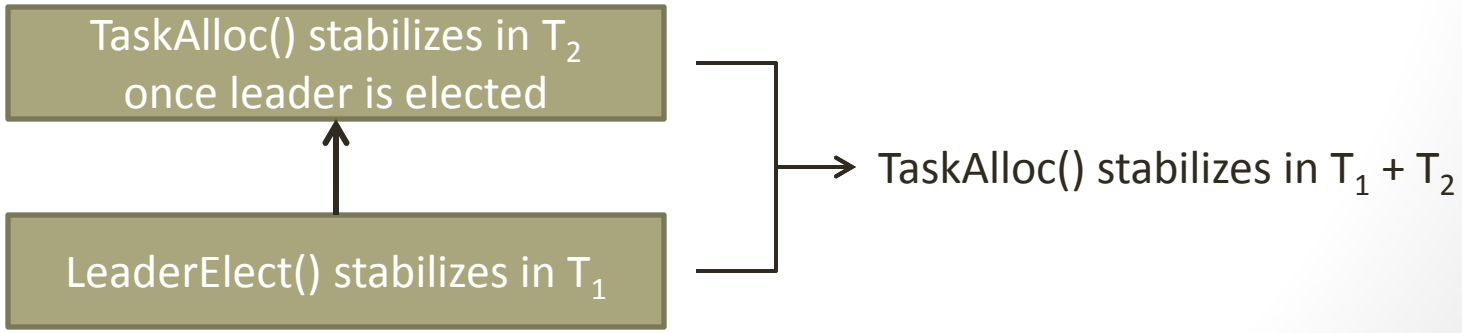
- Collection of software functions for controlling distributed robotic systems
- API for high level application programming
- Functions have well defined properties
- Platform/robot independent

Functionality Provided

- Reliable message passing
- Message loss detection
- Motion controllers
- Built-in algorithms:
  - Geocast
  - Leader election
  - Mutual exclusion
  - Synchronization
- Robust, tolerating failures

Assume-Guaranteed Behavior

- All functions have well defined behavior
  - “Leader election function will elect a leader within 5 rounds or notify all processors of failure”
- Predictable behavior lets us guarantee the entire system

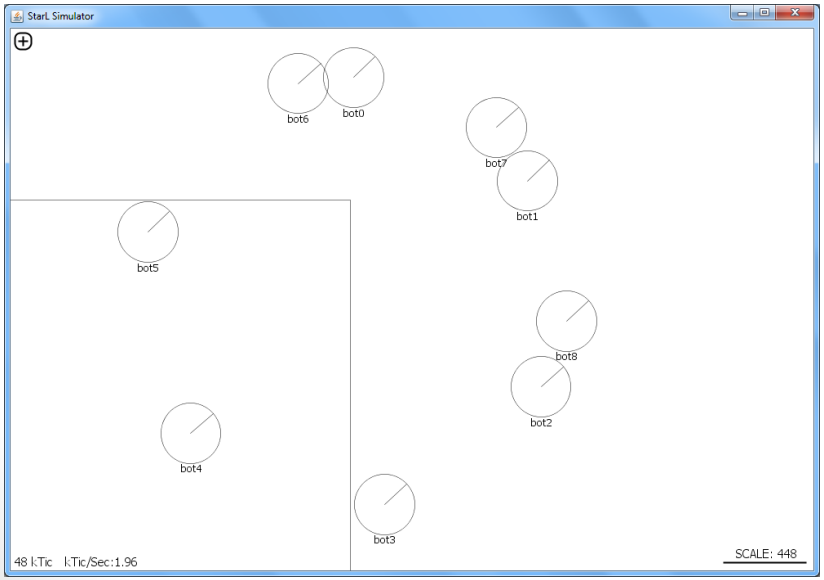


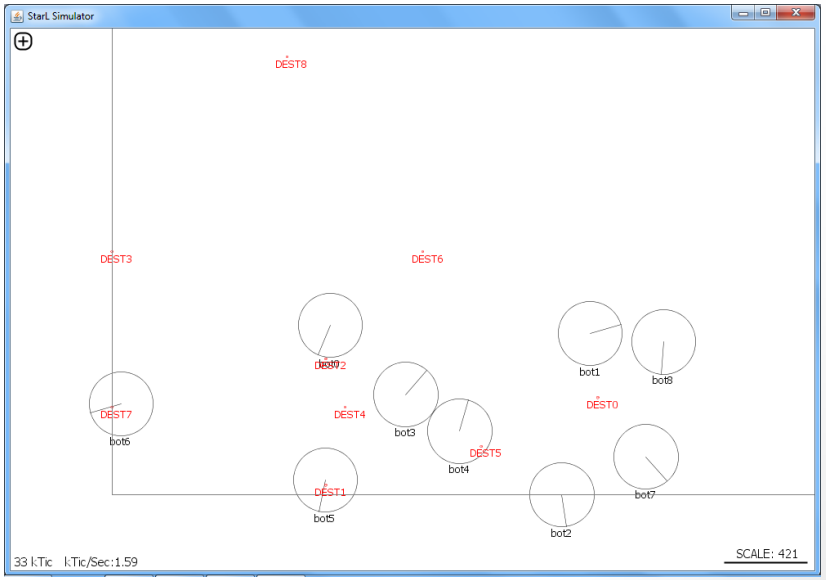
Implementation

- Implemented in Java for Google's Android smartphone OS
- iRobot Create chassis
- IR Camera system for localization

StarL Simulator

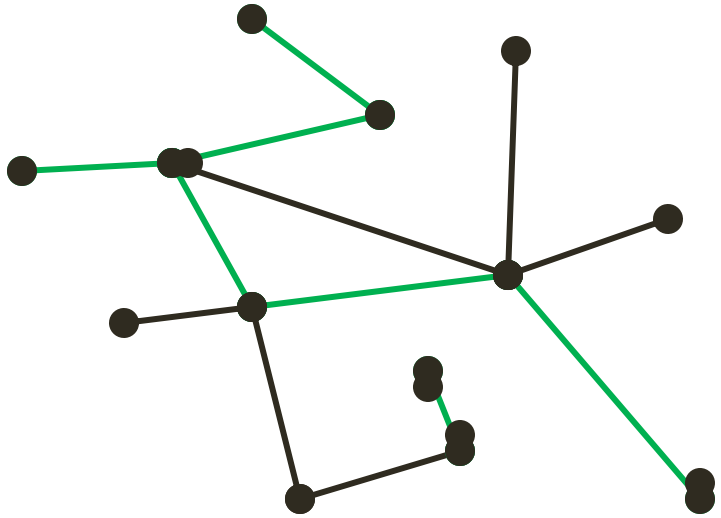
- Any StarL application can be simulated
- Debugging and scalability tests
- Faster and easier than robots





Distributed Path Planning

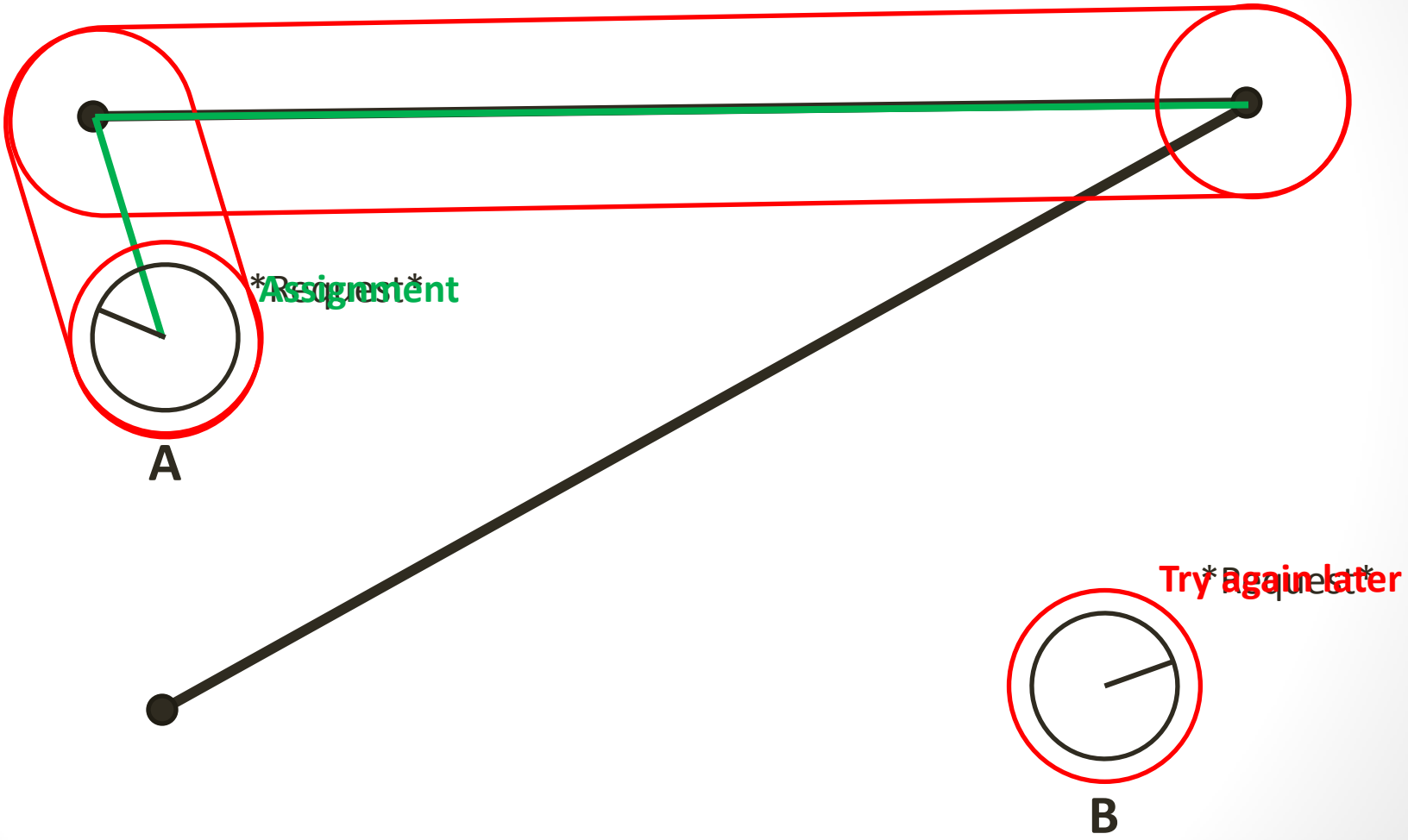
- Algorithm for a collection of robots to compute safe paths to a set of destinations.
- Planar graph $G = (V, E)$ encodes paths and destinations available.
- A subset $T \subseteq E$ are *target edges*



Distributed Path Planning

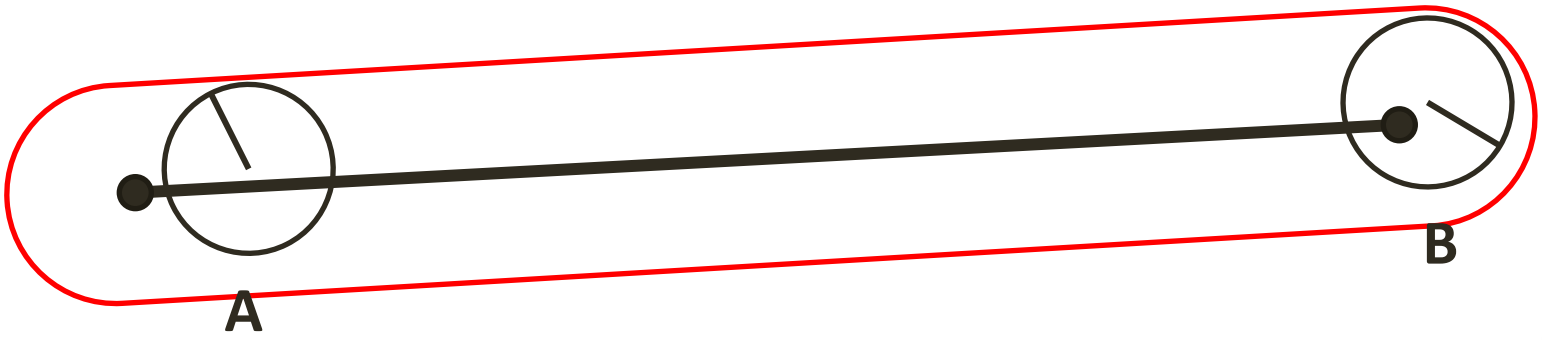
- One robot acts as a *central coordinator* and distributes assignments to each robot
  - Assignment = sequence of points in V
  - At least one edge from T per assignment
  - Each assignment has maximum length H
- Coordinator maintains a reach tube for each robot
  - An over-approximation of a robot's position in the plane
  - Each new assignment must be disjoint from all other reach tubes
- Assignments may be empty
  - Robot remains stationary and requests again later

Distributed Path Planning



Deadlocks

- Condition in which no safe assignments exist and the execution has not completed.
- Formally: An execution of DPP with E arranged such that for each edge $e \in T$ there exist at least two robots within R of e will always deadlock.

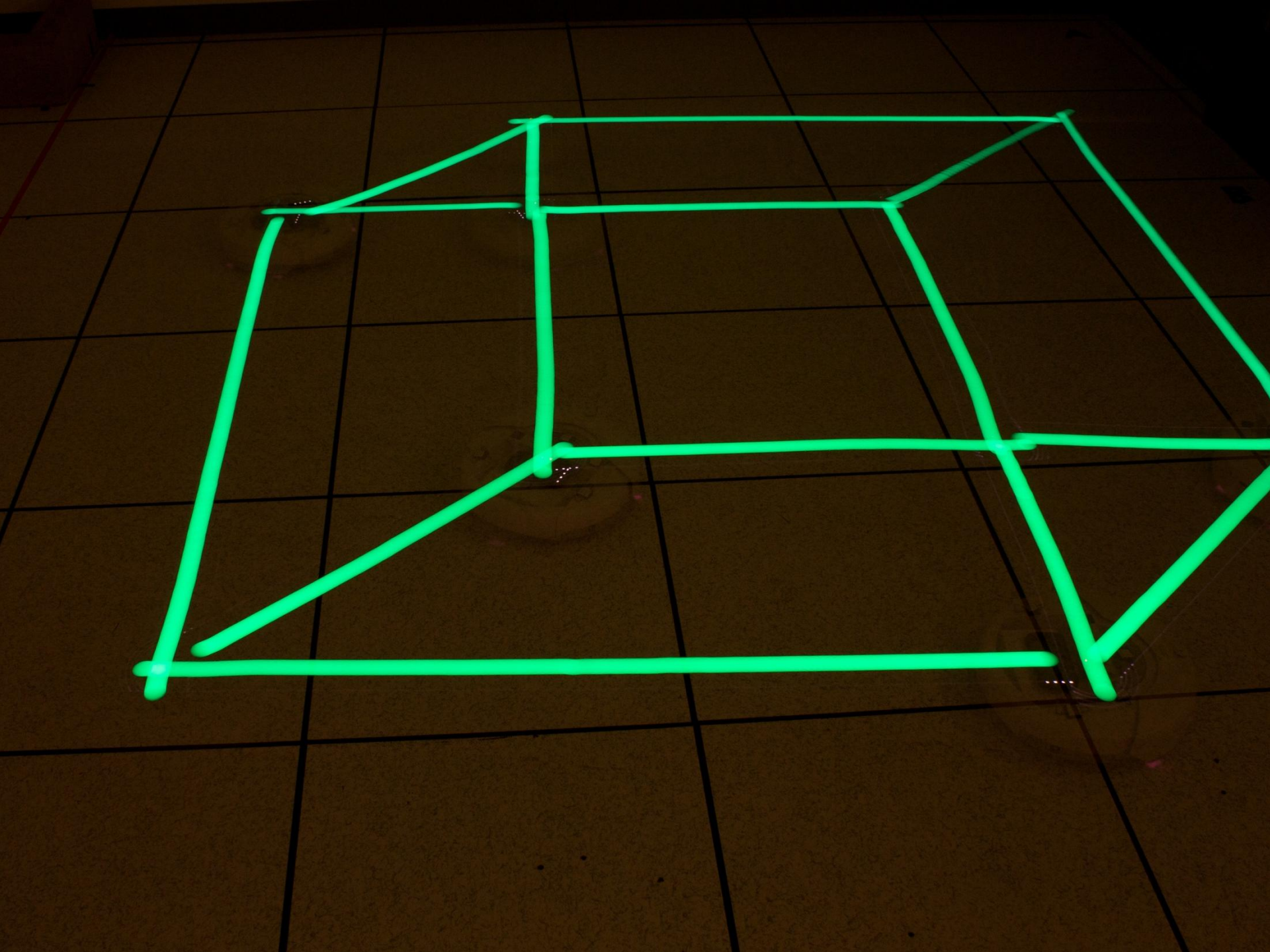


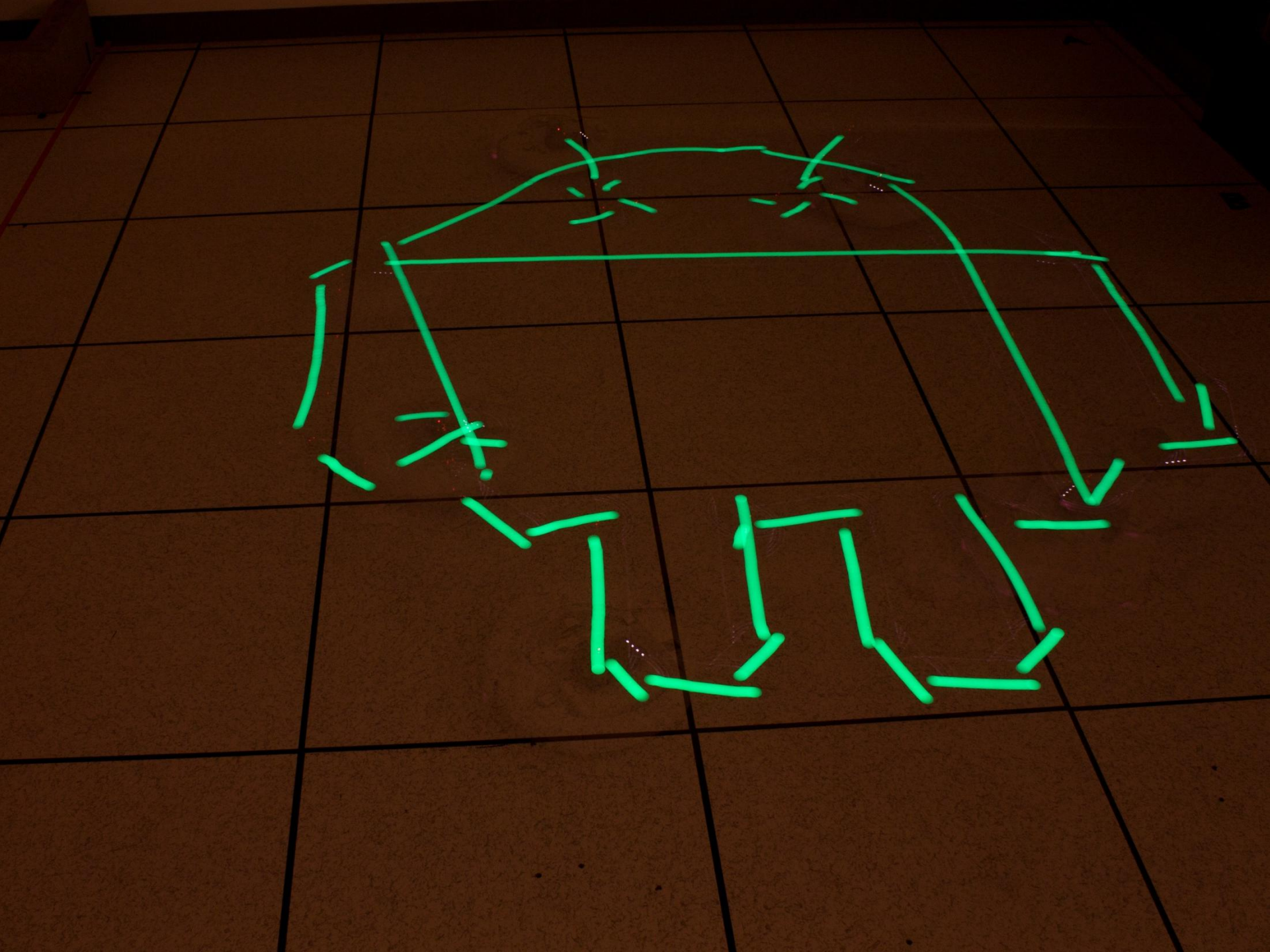
- If an execution never deadlocks, the image is fully drawn.

Collaborative Painting

- T comes from a vector image
- Extra edges in E are generated as needed
  - Probabilistic roadmap path planning
- Output image is recorded using light painting
  - Smartphone screens illuminate when traversing T

Simulator Example





Properties of DPP

- **Safety:** No two robots may be within a safety bound r_s of each other.

- **Progress:** The drawing should be completed as much as possible without violating safety.

DPP Progress

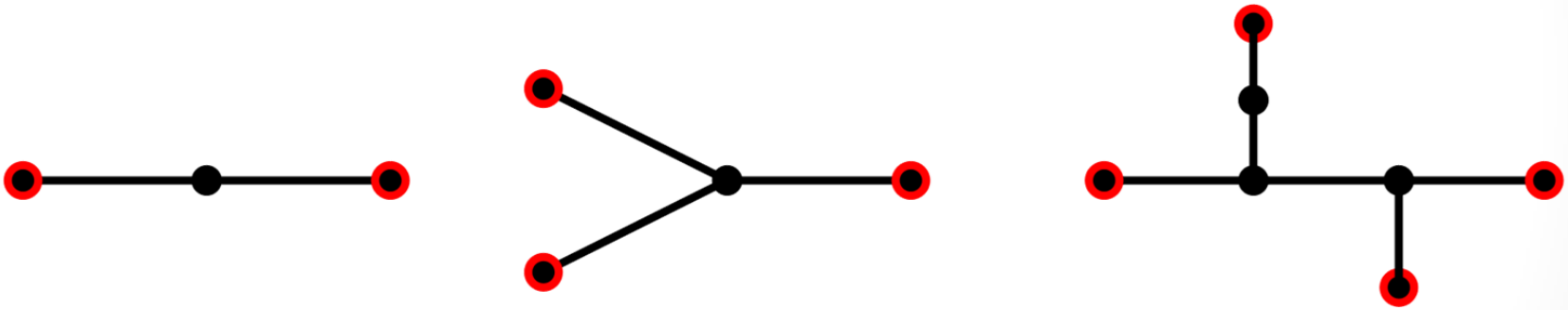
- Formally: At the time of a request from robot i, if there exists a safe edge $e \in FE$ and there exists a safe path between X_i and e, then W_i will be nonempty.
  - The opposite of the deadlock condition
- What if this doesn't hold?
  - Deadlock!
- Can a deadlock be predicted or avoided?

Starting Condition Analysis

- **Progress guarantee:** Can a deadlock be predicted?
  - Can predict only if all sources of indeterminism are removed and G is *completable*
    - Message losses and reception order
    - Assignment calculation must be deterministic
    - Starting locations must be known
  - Special cases exist
    - Single robot executions never deadlock
    - Some graphs are uncompletable

Starting Condition Analysis

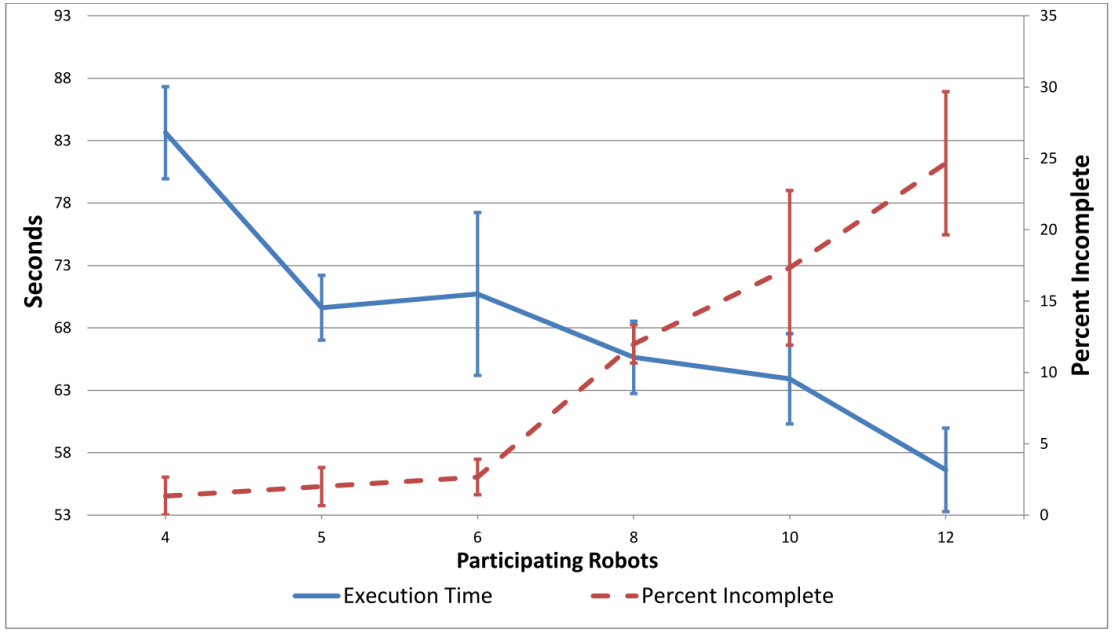
- **Uncompletable graphs** - special cases of G
- A graph G which contains no cycles, n terminal vertices, n participating robots, and $T = E$ is uncompletable.



- Other classes of uncompletable graphs are likely to exist

Starting Condition Analysis

- **In reality -** collaborative painting implementation:
  - Deadlocks are unpredictable
    - Message losses and reception order are unknowable
    - Assignments have nondeterministic components
- Deadlocks are detectable and resolvable



Safety Analysis

- Assumptions used to show safety
  - Assume messages are eventually delivered (no loss)
  - When moving from X_i to X_j at velocity v, a robot is never farther than $r(v)$ from the straight line $\overline{X_i X_j}$
  - Ignore acceleration

- Definition
  - $Unsafe = \bigcup_i ReachTube(X_i, R) + ReachTube(W_i, R)$
  - $R > r_s$
  - $r_s > r(v_{max})$

Safety Analysis

- Robots will always be separated by distance r_s
- Proof sketch:
  - Robots start with minimal separation of r_s
  - Eventually some robot i gets the first assignment W_i
    - Traveling at v_{max}, i is within r_s of W_i while moving
    - W_i is disjoint from $Unsafe$, making the minimum separation distance $R > r_s$
  - Every consecutive assignment will be disjoint from $Unsafe$
  - Empty assignments are disjoint from $Unsafe$

- Holds even if messages are lost!
  - A lost assignment is the same as receiving an empty assignment.

DPP Properties

- **Safety –** Safe even with message losses
- **Progress –** Progress condition identified
  - H and n can be tuned for a specific image or environment size to maximize completion