# Managing Swarms of Robots: From Centralized to Decentralized Scheduling

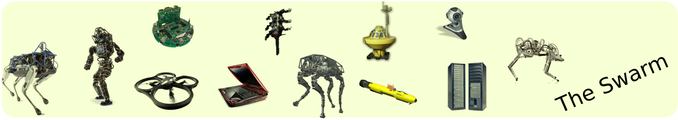
Daniel Graff, Reinhardt Karnapke

FG Kommunikations- und Betriebssysteme
Technische Universität Berlin

13. September 2018

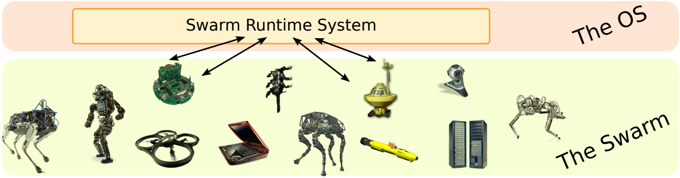
# OS Support for Mobile Robots

- Set of heterogeneous devices (with different capabilities)



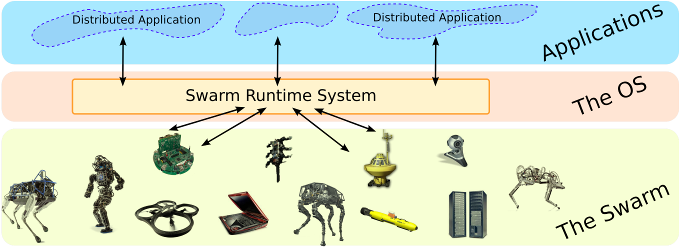
# OS Support for Mobile Robots

- Management, control and coordination by the operating system

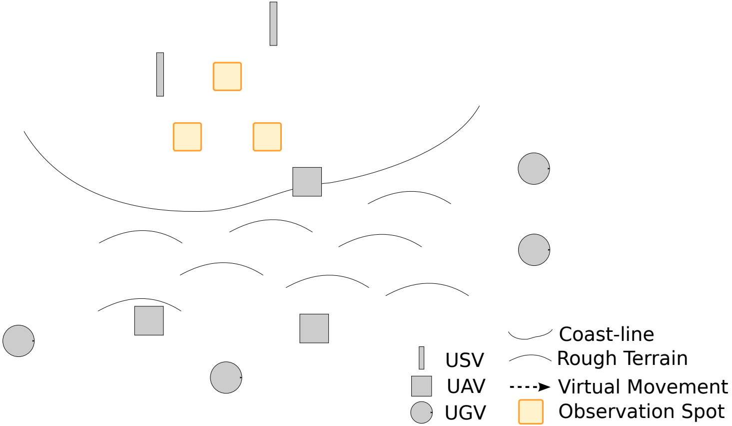


# OS Support for Mobile Robots

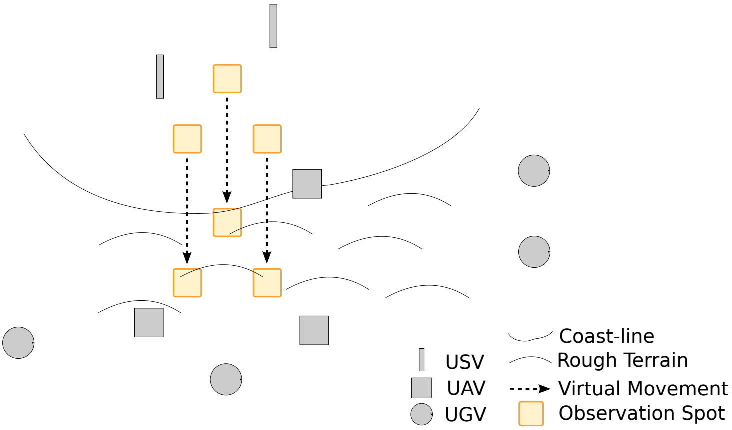
- Multiple (distributed) applications executed (simultaneously on the "swarm"



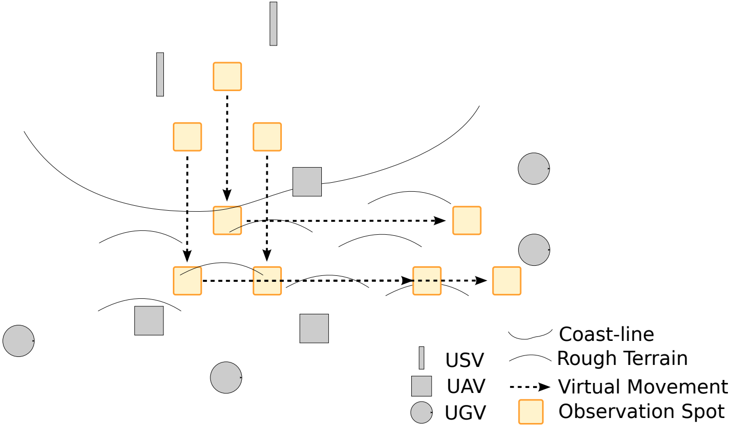
# Example: Exploration



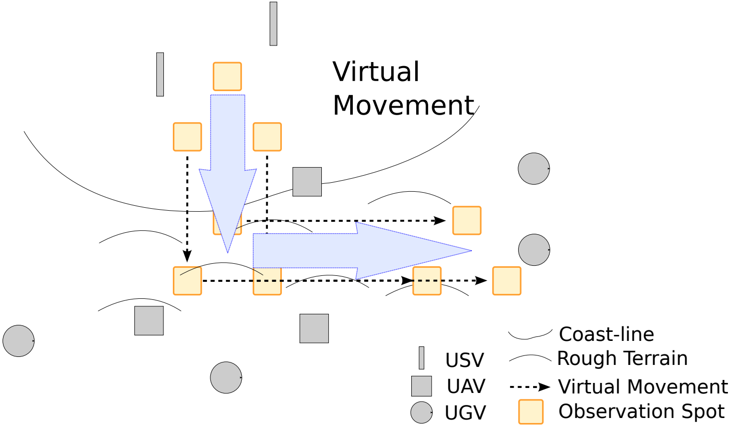
# Example: Exploration



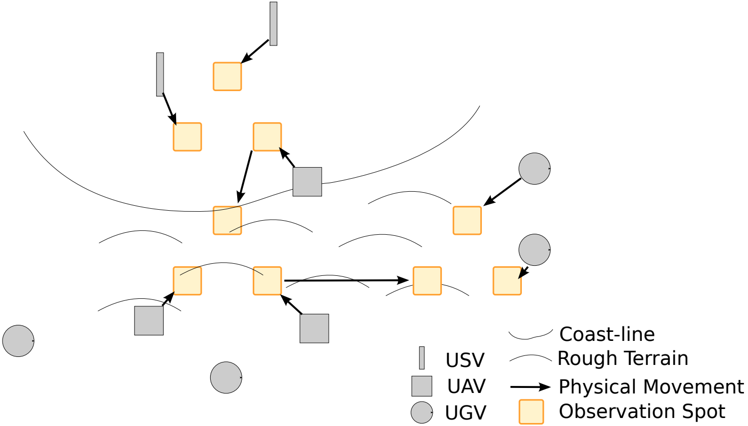
# Example: Exploration



# Example: Exploration



# Example: Exploration



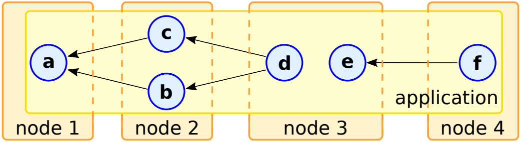
# Types of Movement

## Definition

- Virtual movement: “Logic” movement of application
- Physical movement: Real physical movement

# Application Model

- Building blocks
  - Flexible composition of actions at runtime
  - Connect input / output
- Actions
  - Constrainable in space and time
  - E.g., take picture, measure temperature
- Execution
  - Concurrent / sequential execution of (in-)dependent actions
  - Transparent execution across node boundaries

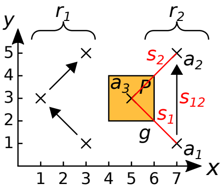


# Scheduling of ActionSuites

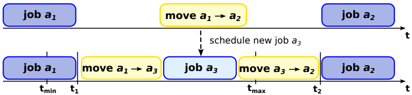
- Input: ActionSuite *as*, dependency graph $g^d$
  - Action $a \in as = (g, t_{min}, t_{max}, d)$
  - $g^d$ is a directed, acyclic graph
- Output: Scheduled job $(j^a, j^t)$ for each $a_i \in as$
  - $j^a = (p, t, r)$
  - $j^t = [(x_1, y_1, t_1), ..]$
- World
  - Static obstacles $O^s$
  - Dynamic obstacles $O^m$, $f : T \to \mathbb{R}^2$

# Determine Slot Candidate

- Determine slot candidate
  - Euclidean distance: $s = ||\vec{x}_{a_i} - \vec{x}_{a_j}||$
  - Overhead of detour: $\Delta s := (s_1 + s_2) - s_{12}$
  - Select slot candidate with minimal detour
- Temporal constraints
  - $t_{min}$, $t_{max}$ are action constraints
  - $t_1$, $t_2$ define begin / end of free slot



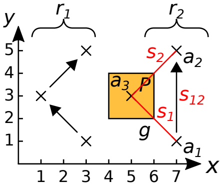

Schedule $a_3$




Schedule of $r_2$

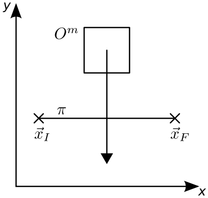
# Path Planner

- Computes trajectory
  - .. from $\vec{x}_{a_1}$ to $\vec{x}_{a_3}(s_1)$
  - .. from $\vec{x}_{a_3}$ to $\vec{x}_{a_2}(s_2)$
- Resulting problem is a Trajectory Planning Problem (TPP)
- Solution: Path-Velocity-Decomposition
  - Static Path Planning Problem (PPP)
  - Dynamic Velocity Planning Problem (VPP)
- Path Planner obtains input from the job scheduler
  - *loc*: location candidate ($P$)
  - *slot*: $(r, [t_1, t_2])$

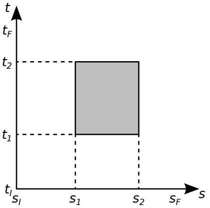


# Forbidden Regions

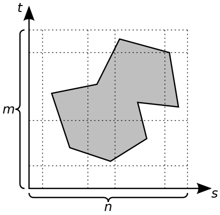
- Compute possible collisions with dynamic obstacles and mark them as forbidden regions
- $O^m$ crosses robot path $\pi$
- The result is a forbidden region in $s \times t$ space
- $n$ robot segment and $m$ segments of $O^m$ creates $n \times m$ tiles




$O^m$ crosses $\pi$



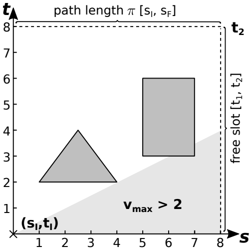

Forbidden region




$n \times m$ tiles

# Velocity Planning Problem (VPP)

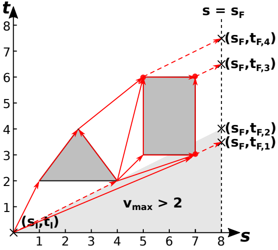
- Computes velocity profile along path $\pi$
- $s \times t$-space shows forbidden regions (dynamic obstacles)
- Find time based mapping $s : T \rightarrow S$, $S = [s_I, s_F] = [0, 8]$
  $s(2) = 1$ or $s(2) = 4$
- Extend to: $\vec{x}_\pi : T \rightarrow \vec{x} \quad \Leftrightarrow \quad \vec{x}_\pi : (T \rightarrow S) \rightarrow \vec{x}$




Intial with $v_{max} = 2$

# Velocity Planning Problem (VPP)

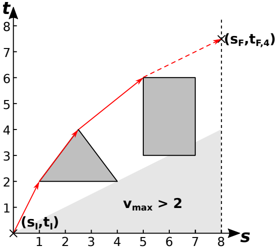
- Computes velocity profile along path $\pi$
- $s \times t$-space shows forbidden regions (dynamic obstacles)
- Find time based mapping $s : T \to S$, $S = [s_I, s_F] = [0, 8]$
  $s(2) = 1$ or $s(2) = 4$
- Extend to: $\vec{x}_\pi : T \to \vec{x} \quad \Leftrightarrow \quad \vec{x}_\pi : (T \to S) \to \vec{x}$




Graph Construction

# Velocity Planning Problem (VPP)

- Computes velocity profile along path $\pi$
- $s \times t$-space shows forbidden regions (dynamic obstacles)
- Find time based mapping $s : T \rightarrow S$, $S = [s_I, s_F] = [0, 8]$
  $s(2) = 1$ or $s(2) = 4$
- Extend to: $\vec{x}_\pi : T \rightarrow \vec{x} \quad \Leftrightarrow \quad \vec{x}_\pi : (T \rightarrow S) \rightarrow \vec{x}$




Shortest Path

# Complexity and Scalability

- VPP induces high complexity
  - Meshing to obtain velocity graph
  - Check of visibility

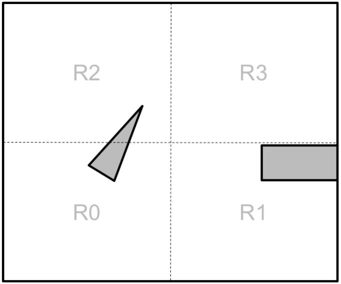
  $\rightarrow$ Execution time depends on $O^m$, $\pi^m$, $\pi$ (segments) planning
- However, scalability problem still remains
  - Very large worlds
  - Huge amount of robots

# Possible Solutions

- GPGPU computing suitable for trajectory planning
- Approaches from autonomous driving, TP with time horizon
  $\rightarrow$ Loose guarantees and predictability, deadlocks
- Scalability $\rightarrow$ Decentralized Approaches
  - Static Approach
  - Dynamic Approach

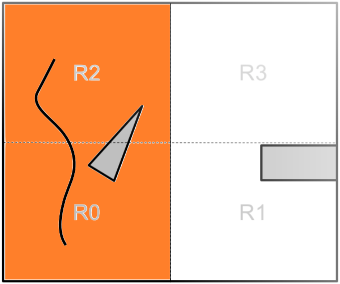
# Static Approach

- Splitting the world into static regions
- Each regions has its own scheduler
- Scheduling decentralized and in parallel



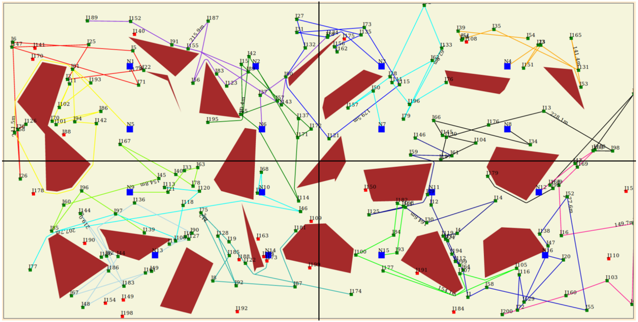
# Static Approach

- Cross-boundary movement possible
- .. requires locking
- Full locking $\rightarrow$ degenerates to centralized version $\rightarrow$ unlikely



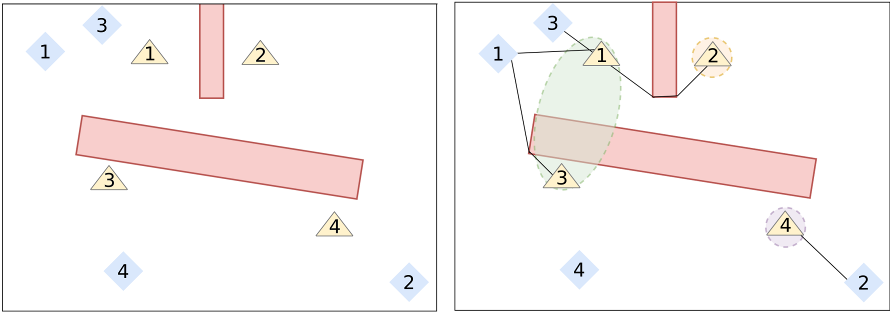
# Static Approach

- ~250 trajectory segments
- 17 segments with cross-boundary movement
- $\approx$ 93.2%



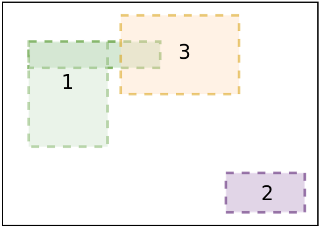
# Dynamic Approach

- Regions emerge dynamically based on scheduling requests
- Formation of “scheduler groups”
- Run distributed and in parallel
- Getting destroyed after request



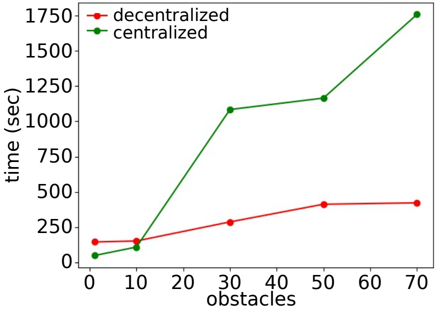
# Dynamic Approach

- Each group creates a region lock
- Bounding box of group
- If overlap $\rightarrow$ dependency (waiting) graph
- Variants
  - One node becomes scheduler
  - All nodes become schedulers (first one wins)
  - All nodes become schedulers (consensus / majority voting)



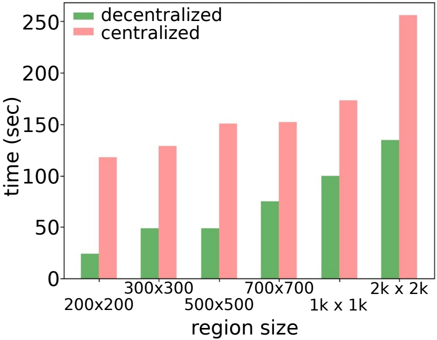
# Preliminary Results

- 25 nodes: Core i3-6100U CPU @ 2.30GHz, dual core
- Time: Centralized vs. decentralized
- $f(o)$, number of obstacles
- 10,000 jobs
- More obstacles create more path segments
  $\rightarrow$ Complexity of forbidden regions increase



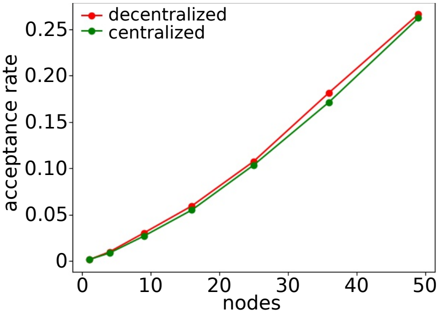
# Preliminary Results

- 25 nodes: Core i3-6100U CPU @ 2.30GHz, dual core
- Time: Centralized vs. decentralized
- $f(r)$, region size
- 10,000 jobs

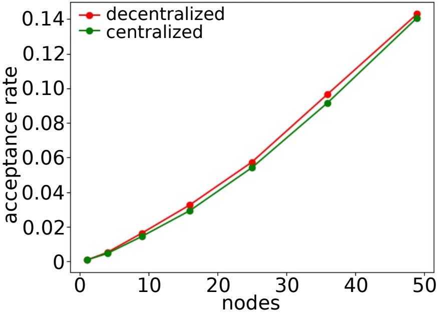


# Preliminary Results

- 50 nodes: Core i3-6100U CPU @ 2.30GHz, dual core
- Acceptance rate: Centralized vs. decentralized
- $f(n)$, number of nodes
- Non-linear increase of acceptance rate
- (De-)centralized approach has only little impact




5k jobs




10k jobs

# Conclusion / Future Work

- OS support for mobile robot swarms
- TPP has high complexity
- Address scalability by decentralization
  - Static approach
  - Dynamic approach
- Locking vs non-locking
- Distributed data structures: transfer only delta schedule
  - After scheduling
  - On request