BeeCluster: Drone Orchestration via Predictive Optimization

Presenter: Songtao He (songtao@mit.edu)

Present at ACM MobiSys 2020

Joint work with Favyen Bastani, Arjun Balasingam, Karthik Gopalakrishnan, Ziwen Jiang, Mohammad Alizadeh, Hari Balakrishnan, Michael Cafarella, Tim Kraska, Sam Madden

Massachusetts Institute of Technology

CSAIL

Aerial Drones Enable Many Applications

Aerial Mapping	**Inspection**	**Environmental Monitoring**	**Agriculture**
Delivery	**Disaster Management**	**Wildlife Monitoring**	**Search and Rescue**

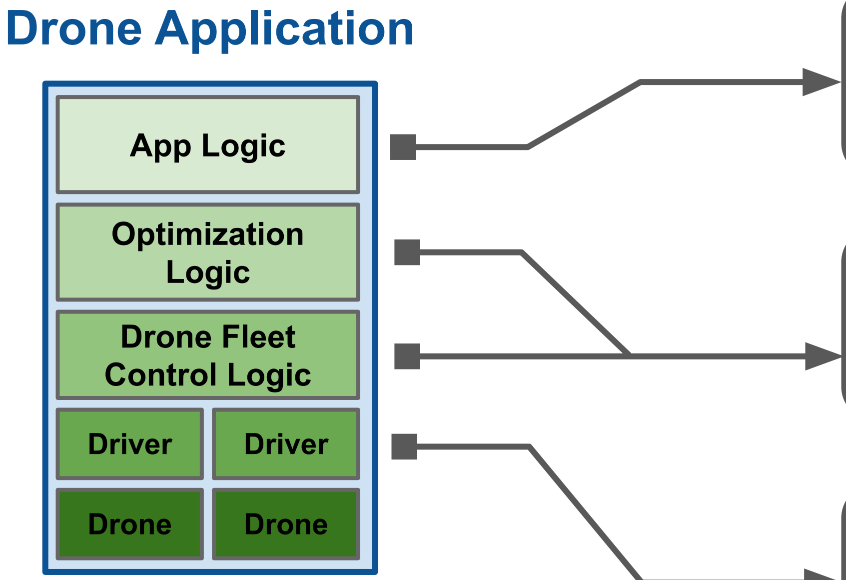
Emerging Autonomous Drone Applications

Applications which require human operators

Autonomous Drone Applications

https://www.dronepilotgroundschool.com/full-time-drone-jobs/
https://www.information-age.com/ai-powered-drone-swarms-123479202/

Developing Autonomous Drone Applications is Challenging

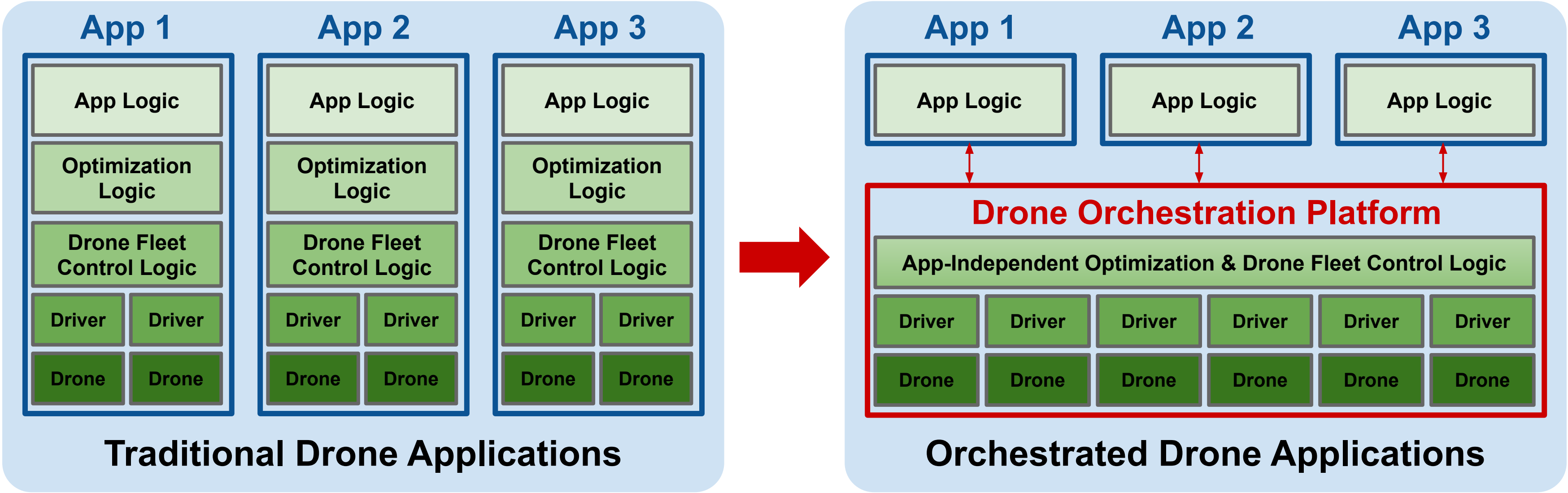


- Where to collect sensing data?
- How to process the data?
- How to react to the processing result?

- Task-to-drone scheduling.
- Routes planning.
- Drone monitoring, e.g., battery time

- Access raw sensors
- Drone control
- Telemetry

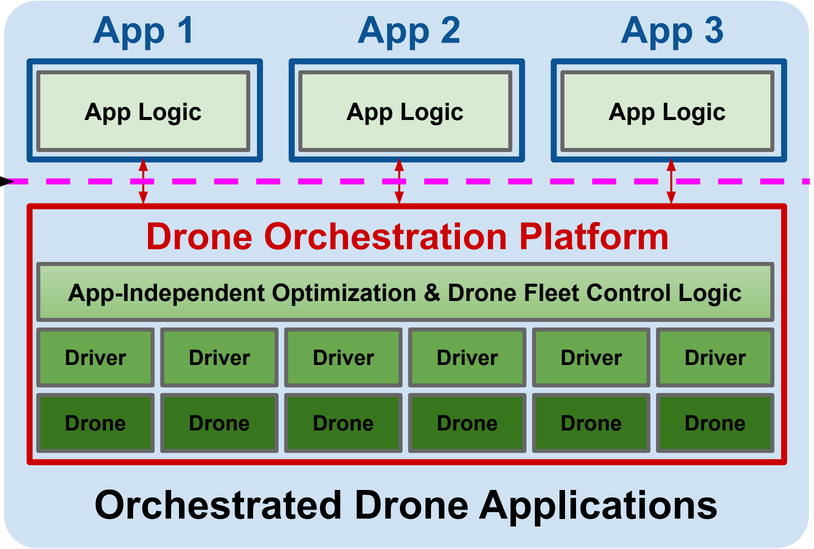
Drone Orchestration Comes to Help



BeeCluster Drone Orchestration Platform

Two Interesting Design Questions

- Programming Interface

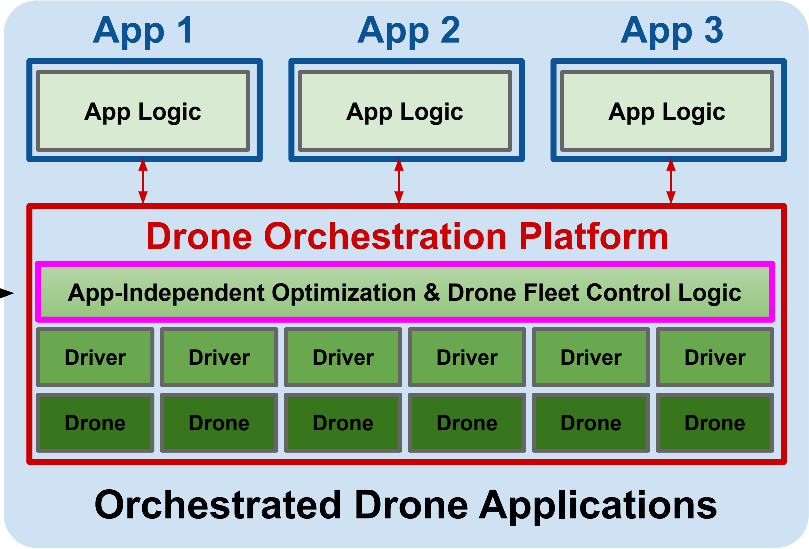


BeeCluster Drone Orchestration Platform

Two Interesting Design Questions

- Programming Interface

- Application-independent Optimization

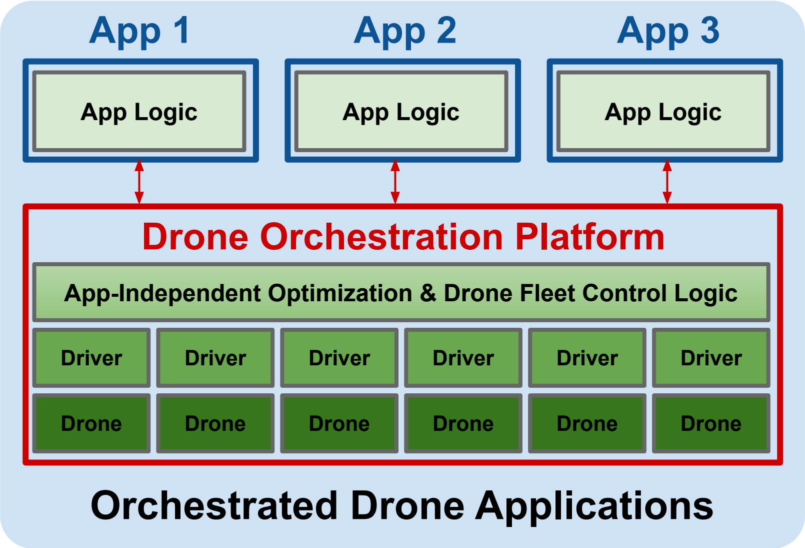


BeeCluster Drone Orchestration Platform

Two Interesting Design Questions

- Programming Interface

- Application-independent Optimization

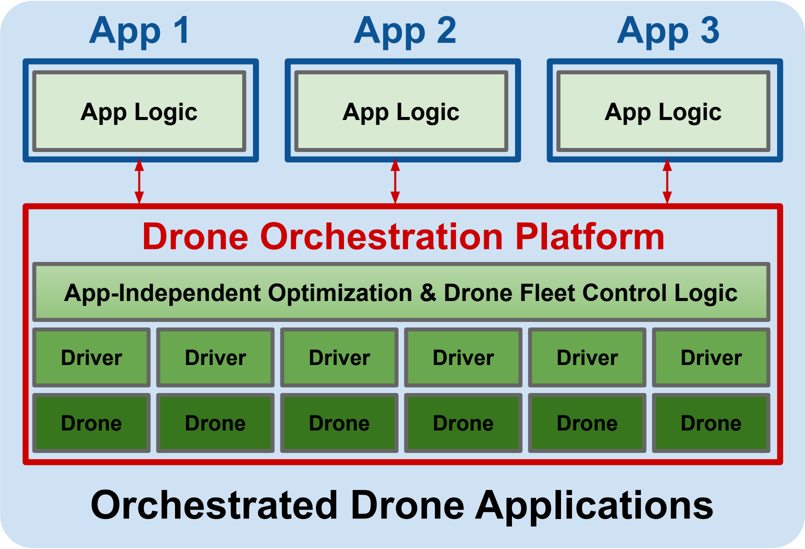


BeeCluster Drone Orchestration Platform

Two Interesting Design Questions

- Programming Interface
  - (1) Dynamic Tasking
  - (2) Rich Task-to-Drone Mapping Options

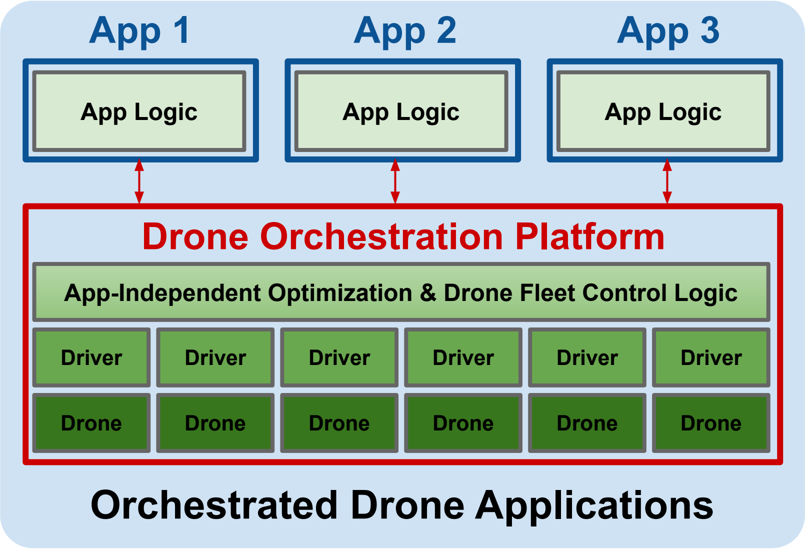
- Application-independent Optimization
  - (3) Predictive Optimization
    - Forecast applications' future behavior and use the forecast to speed up the execution of the app.



BeeCluster Drone Orchestration Platform

Two Interesting Design Questions

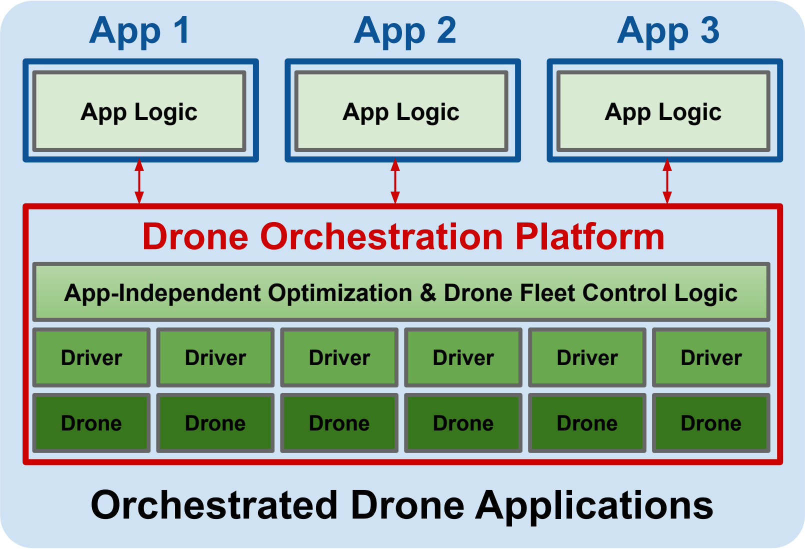
- Programming Interface
  - **(1) Dynamic Tasking**
  - **(2) Rich Task-to-Drone Mapping Options**
- Application-independent Optimization
  - (3) Predictive Optimization
    - Forecast applications' future behavior and use the forecast to speed up the execution of the app.



BeeCluster Drone Orchestration Platform

Two Interesting Design Questions

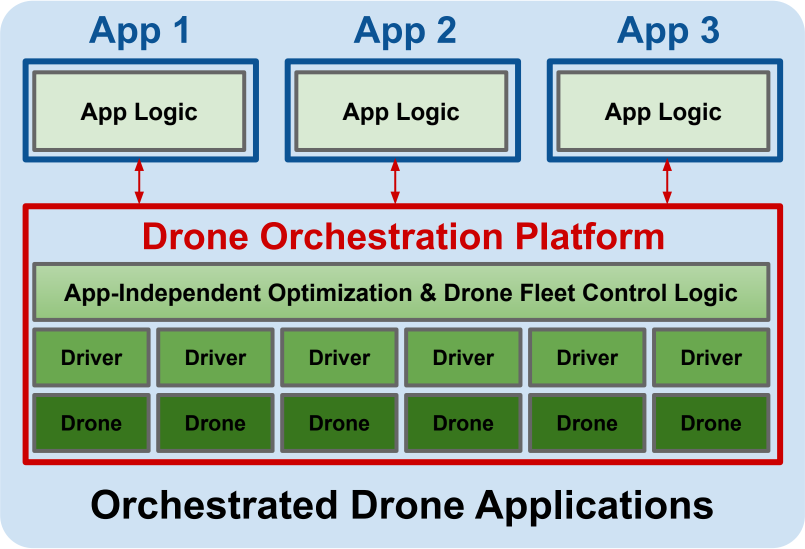
- Programming Interface
  - (1) Dynamic Tasking
  - (2) Rich Task-to-Drone Mapping Options
- Application-independent Optimization
  - **(3) Predictive Optimization**
    - **Forecast applications' future behavior and use the forecast to speed up the execution of the app.**



BeeCluster Drone Orchestration Platform

Two Interesting Design Questions

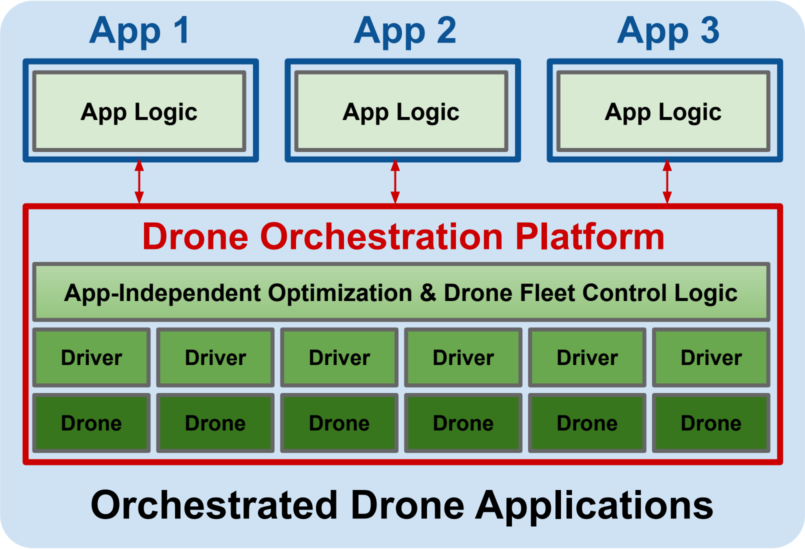
- Programming Interface
  - (1) Dynamic Tasking
  - (2) Rich Task-to-Drone Mapping Options
- Application-independent Optimization
  - (3) Predictive Optimization
    - Forecast applications' future behavior and use the forecast to speed up the execution of the app.



BeeCluster Drone Orchestration Platform

Two Interesting Design Questions

- Programming Interface
  - Dynamic Tasking
  - Rich Task-to-Drone Binding Options
- Application-independent Optimization
  - **Predictive Optimization**
    - Forecast applications' future behavior and use the forecast to speed up the execution of the app.

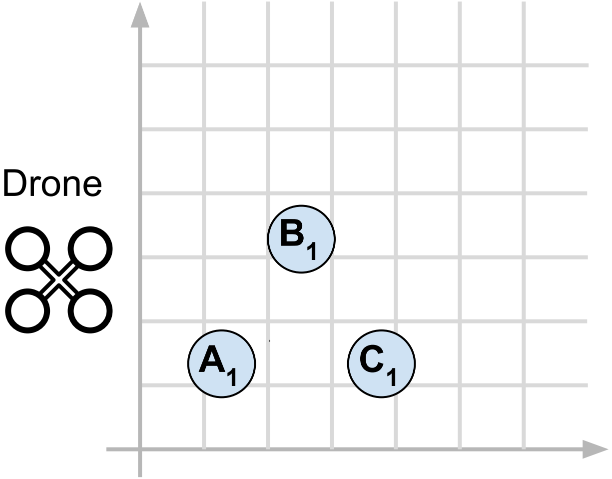


Example of Predictive Optimization

```
A,B,C = initial locations
while True:
  values = SenseAtLocations([A,B,C])
  A,B,C = UpdateLocations(A,B,C,values)
```

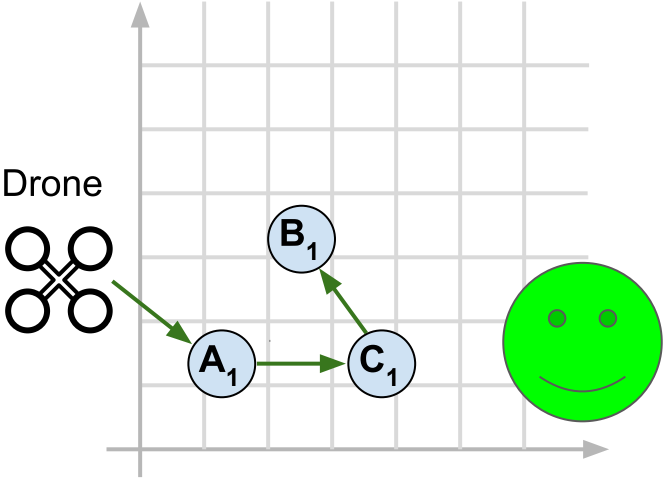
Example of Predictive Optimization

```
A,B,C = initial locations
while True:
  values = SenseAtLocations([A,B,C])
  A,B,C = UpdateLocations(A,B,C,values)
```

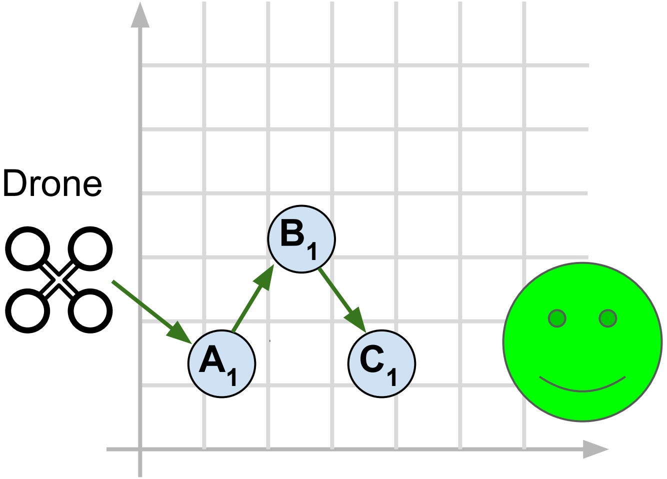


Example of Predictive Optimization

```
A,B,C = initial locations
while True:
  values = SenseAtLocations([A,B,C])
  A,B,C = UpdateLocations(A,B,C,values)
```



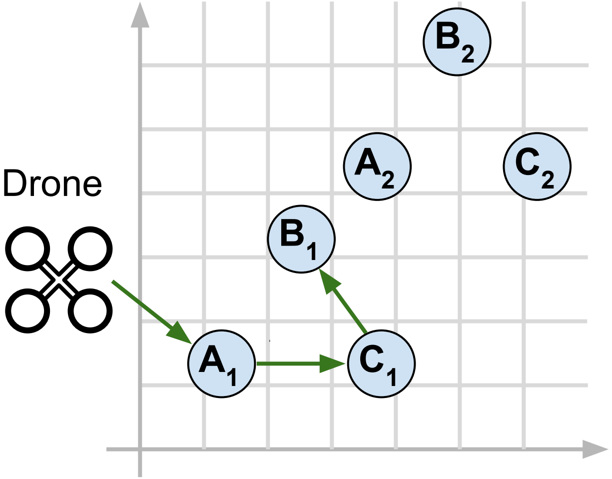
A→ C→ B (Counter-Clockwise)



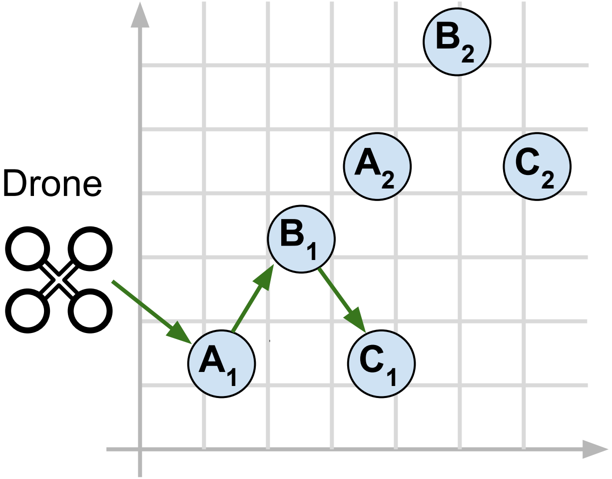
A→ B→ C (Clockwise)

Example of Predictive Optimization

```
A,B,C = initial locations
while True:
  values = SenseAtLocations([A,B,C])
  A,B,C = UpdateLocations(A,B,C,values)
```



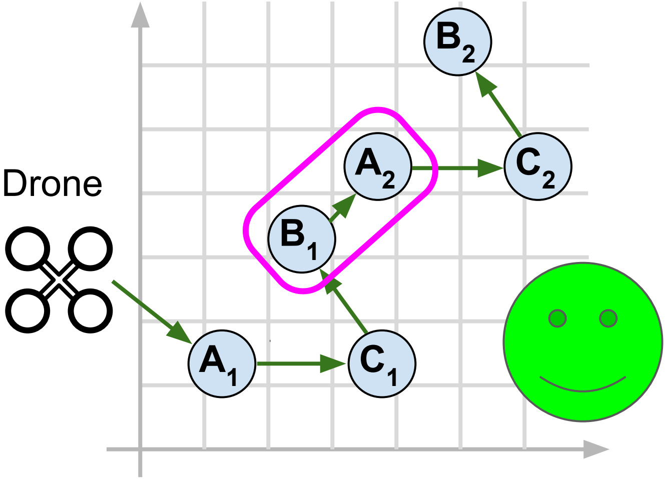
A→ C→ B (Counter-Clockwise)



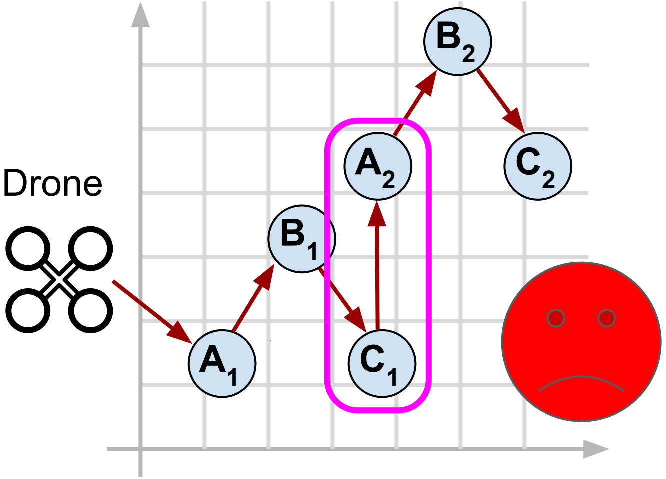
A→ B→ C (Clockwise)

Example of Predictive Optimization

```
A,B,C = initial locations
while True:
  values = SenseAtLocations([A,B,C]) # Second Iteration
  A,B,C = UpdateLocations(A,B,C,values)
```



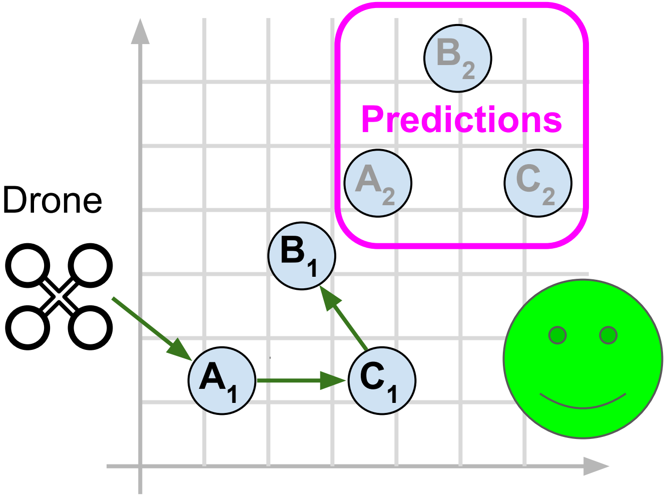
Case 1 (Better)



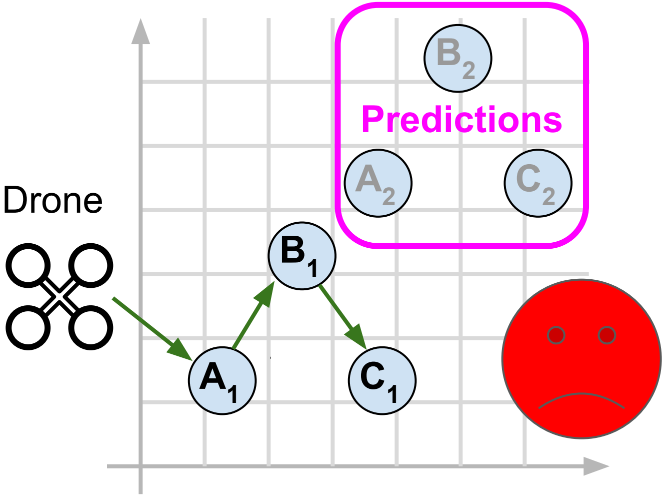
Case 2

Example of Predictive Optimization

```
A,B,C = initial locations
while True:
  values = SenseAtLocations([A,B,C]) # First Iteration
  A,B,C = UpdateLocations(A,B,C,values)
```



A→ C→ B (Counter-Clockwise)



A→ B→ C (Clockwise)

Outline

- Introduction
- Motivation
- **Design of BeeCluster**
  - Programming Interface
    - Programming model (dynamic tasking)
    - Task-to-drone mapping options
  - Predictive Optimization
- Implementation
- Case Studies

Outline

- Introduction
- Motivation
- Design of BeeCluster
  - Programming Interface
    - **Programming model (dynamic tasking)**
    - Task-to-drone mapping options
  - Predictive Optimization
- Implementation
- Case Studies

BeeCluster Programming Model

- Programming Model
  - An imperative programming model based on dynamic task graph (DTG)

BeeCluster Programming Model

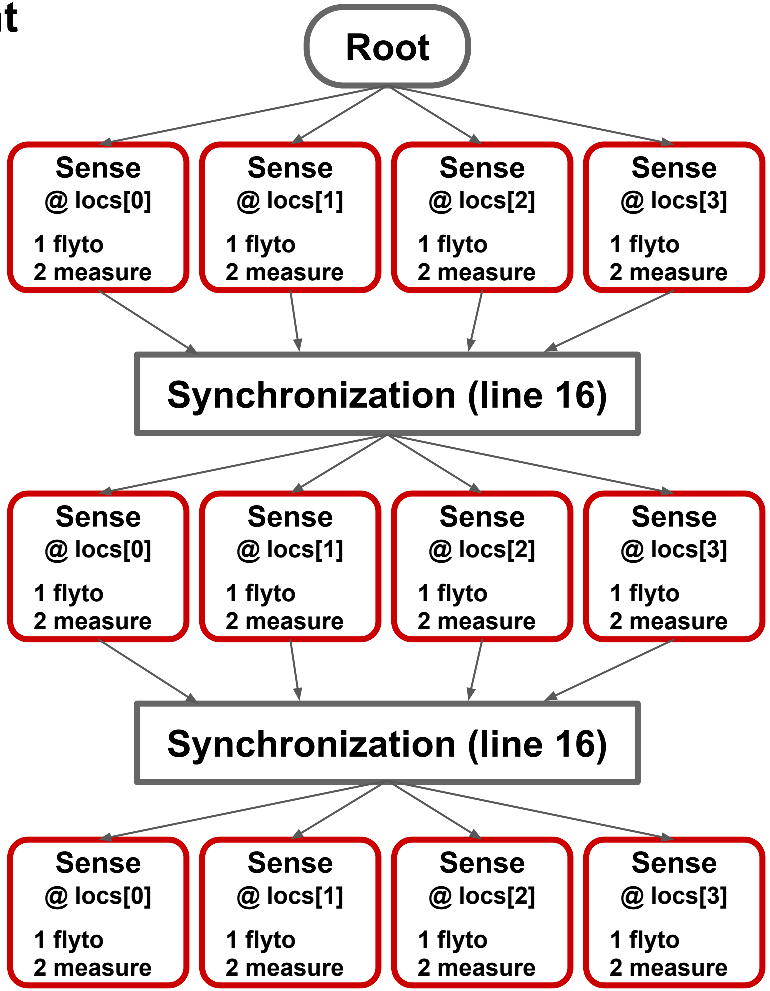
- Programming Model
  - An imperative programming model based on dynamic task graph (DTG)
- Programming Primitives (Building Blocks)
  - Actions - E.g., fly to a location, and take a photo.
  - Tasks - Sequence of actions, multiple tasks run concurrently.

BeeCluster Programming Model

- Programming Model
  - An imperative programming model based on dynamic task graph (DTG)
- Programming Primitives (Building Blocks)
  - Actions - E.g., fly to a location, and take a photo.
  - Tasks - Sequence of actions, multiple tasks run concurrently.
- Basic API
  - ***Non-blocking*** creations of new actions and new tasks:
    - handler = act(args)
    - handler = newTask(function_entry, args, ...)
  - Retrieve values from the handler (***blocking***): result = handler.value
  - Cancel tasks: cancelTask(task_handler)

Example: Locate WiFi Hotspot Using Gradient Descent

```
1   def sense(loc):
2     act("flyto",loc)
3     handle = act("measure_WiFi_signal")
4     return handle.value # blocking
5
6   if __name__ == "__main__":
7     loc = initial_loc # e.g., (0,0)
8     for i in range(10):
9       locs = four_corners(loc)
10
11      task0 = newTask(sense, locs[0])
12      task1 = newTask(sense, locs[1])
13      task2 = newTask(sense, locs[2])
14      task3 = newTask(sense, locs[3])
15
16      values = [ task0.value, task1.value,
                   task2.value, task2.value ]
        # blocking

17      loc = GradientDescent(locs, values)
```



Example: Locate WiFi Hotspot Using Gradient Descent

```
def sense(loc):
  act(“flyto”,loc)
  handle = act(“measure_WiFi_signal”)
  return handle.value # blocking

if __name__ == “__main__”:
  loc = initial_loc # e.g., (0,0)
  for i in range(10):
    locs = four_corners(loc)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```

Example: Locate WiFi Hotspot Using Gradient Descent

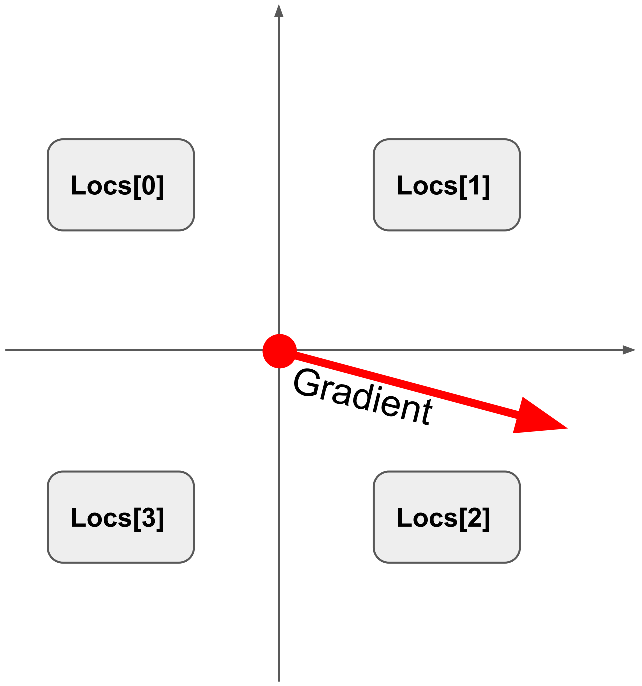
```
def sense(loc):
  act(“flyto”,loc)
  handle = act(“measure_WiFi_signal”)
  return handle.value # blocking

if __name__ == “__main__”:
  loc = initial_loc # e.g., (0,0)
  for i in range(10):
    locs = four_corners(loc)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```

Example: Locate WiFi Hotspot Using Gradient Descent

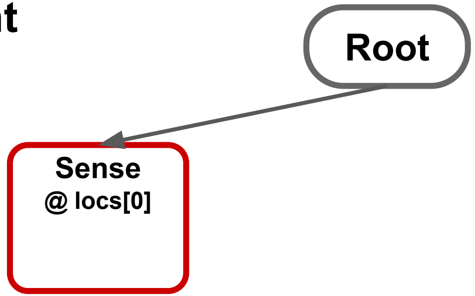
```
def sense(loc):
  act("flyto",loc)
  handle = act("measure_WiFi_signal")
  return handle.value # blocking

if __name__ == "__main__":
  loc = initial_loc # e.g., (0,0)
  for i in range(10):
    locs = four_corners(loc)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```

Example: Locate WiFi Hotspot Using Gradient Descent

```
def sense(loc):
  act(“flyto”,loc)
  handle = act(“measure_WiFi_signal”)
  return handle.value # blocking

if __name__ == “__main__”:
  loc = initial_loc # e.g., (0,0)
  for i in range(10):
    locs = four_corners(loc)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```

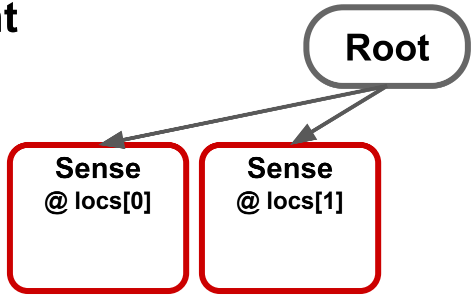


Example: Locate WiFi Hotspot Using Gradient Descent



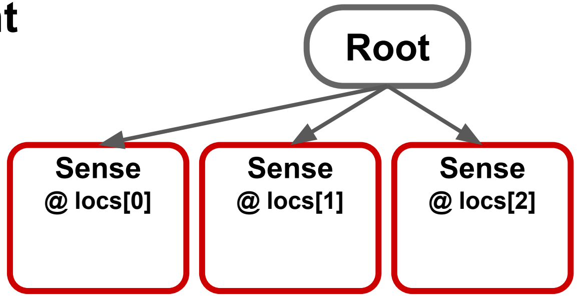

```
def sense(loc):
  act("flyto",loc)
  handle = act("measure_WiFi_signal")
  return handle.value # blocking

if __name__ == "__main__":
  loc = initial_loc # e.g., (0,0)
  for i in range(10):
    locs = four_corners(loc)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```

Example: Locate WiFi Hotspot Using Gradient Descent




```
def sense(loc):
  act("flyto",loc)
  handle = act("measure_WiFi_signal")
  return handle.value # blocking

if __name__ == "__main__":
  loc = initial_loc # e.g., (0,0)
  for i in range(10):
    locs = four_corners(loc)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```

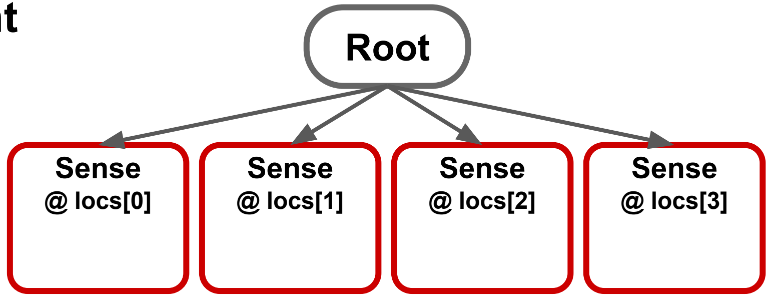
Example: Locate WiFi Hotspot Using Gradient Descent



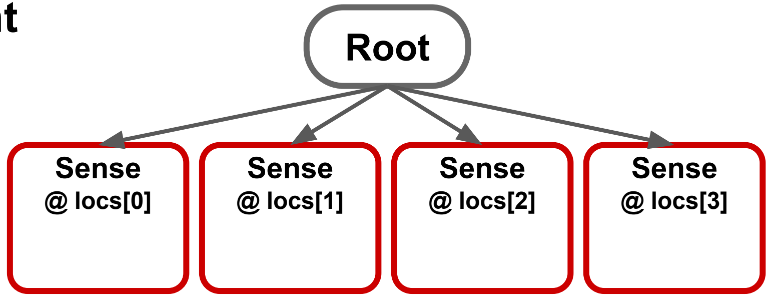

```
def sense(loc):
  act(“flyto”,loc)
  handle = act(“measure_WiFi_signal”)
  return handle.value # blocking

if __name__ == “__main__”:
  loc = initial_loc # e.g., (0,0)
  for i in range(10):
    locs = four_corners(loc)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```

Example: Locate WiFi Hotspot Using Gradient Descent

```
def sense(loc):
  act(“flyto”,loc)
  handle = act(“measure_WiFi_signal”)
  return handle.value # blocking

if __name__ == “__main__”:
  loc = initial_loc # e.g., (0,0)
  for i in range(10):
    locs = four_corners(loc)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```

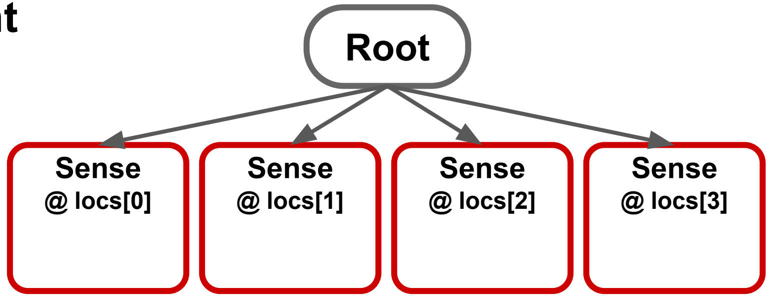


Example: Locate WiFi Hotspot Using Gradient Descent




```
def sense(loc):
  act(“flyto”,loc)
  handle = act(“measure_WiFi_signal”)
  return handle.value # blocking

if __name__ == “__main__”:
  loc = initial_loc # e.g., (0,0)
  for i in range(10):
    locs = four_corners(loc)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```

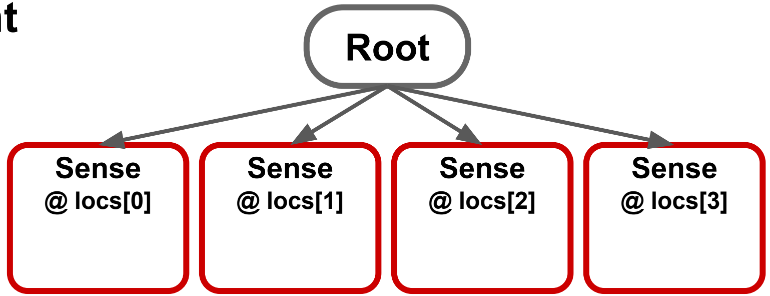
Example: Locate WiFi Hotspot Using Gradient Descent




```
def sense(loc):
  act(“flyto”,loc)
  handle = act(“measure_WiFi_signal”)
  return handle.value # blocking

if __name__ == “__main__”:
  loc = initial_loc # e.g., (0,0)
  for i in range(10):
    locs = four_corners(loc)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```

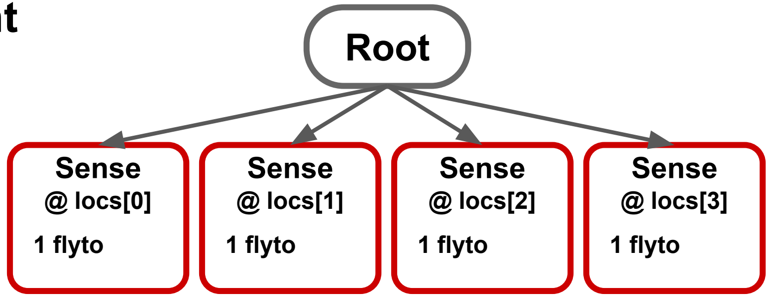
Example: Locate WiFi Hotspot Using Gradient Descent

```
def sense(loc):
    act("flyto",loc)
    handle = act("measure_WiFi_signal")
    return handle.value # blocking

if __name__ == "__main__":
  loc = initial_loc # e.g., (0,0)
  for i in range(10):
    locs = four_corners(loc)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```



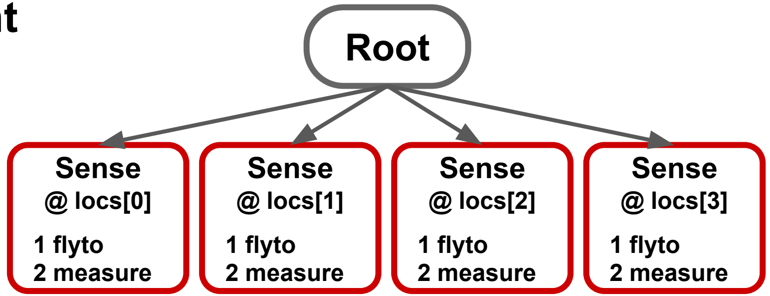
Example: Locate WiFi Hotspot Using Gradient Descent

```
def sense(loc):
    act("flyto",loc)
    handle = act("measure_WiFi_signal")
    return handle.value # blocking

if __name__ == "__main__":
   loc = initial_loc # e.g., (0,0)
   for i in range(10):
      locs = four_corners(loc)

      task0 = newTask(sense, locs[0])
      task1 = newTask(sense, locs[1])
      task2 = newTask(sense, locs[2])
      task3 = newTask(sense, locs[3])

      values = [ task0.value, task1.value,
                 task2.value, task2.value ]
      # blocking

      loc = GradientDescent(locs, values)
```



Example: Locate WiFi Hotspot Using Gradient Descent

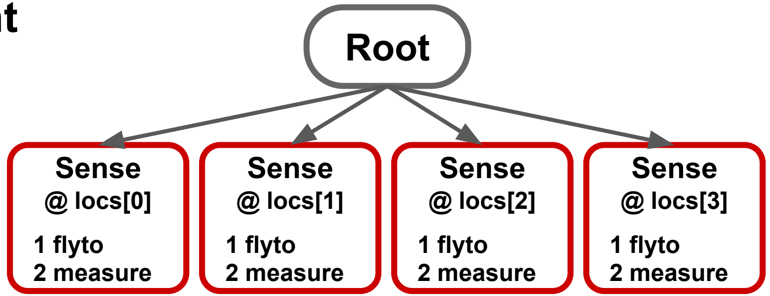
```
def sense(loc):
  act("flyto",loc)
  handle = act("measure_WiFi_signal")
  return handle.value # blocking

if __name__ == "__main__":
  loc = initial_loc # e.g., (0,0)
  for i in range(10):
    locs = four_corners(loc)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```



Example: Locate WiFi Hotspot Using Gradient Descent

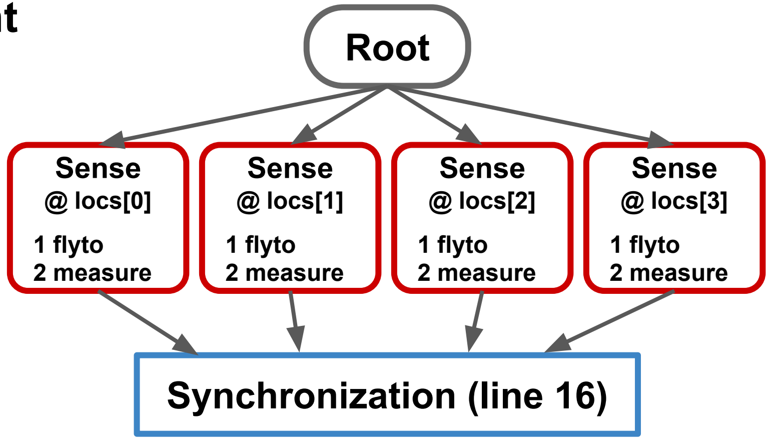
```
def sense(loc):
  act("flyto",loc)
  handle = act("measure_WiFi_signal")
  return handle.value # blocking
```

```
if __name__ == "__main__":
  loc = initial_loc # e.g., (0,0)
  for i in range(10):
    locs = four_corners(loc)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```



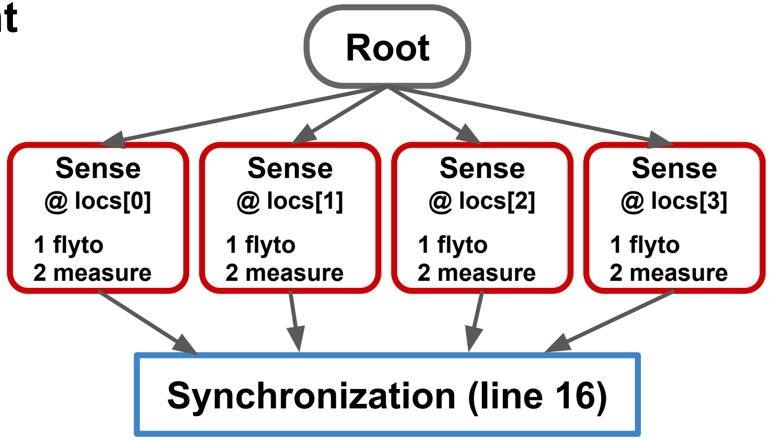
Example: Locate WiFi Hotspot Using Gradient Descent

```
1   def sense(loc):
2     act(“flyto”,loc)
3     handle = act(“measure_WiFi_signal”)
4     return handle.value # blocking
5
6   if __name__ == “__main__”:
7     loc = initial_loc # e.g., (0,0)
8     for i in range(10):
9       locs = four_corners(loc)
10
11      task0 = newTask(sense, locs[0])
12      task1 = newTask(sense, locs[1])
13      task2 = newTask(sense, locs[2])
14      task3 = newTask(sense, locs[3])
15
16      values = [ task0.value, task1.value,
                   task2.value, task2.value ]
        # blocking

17      loc = GradientDescent(locs, values)
```



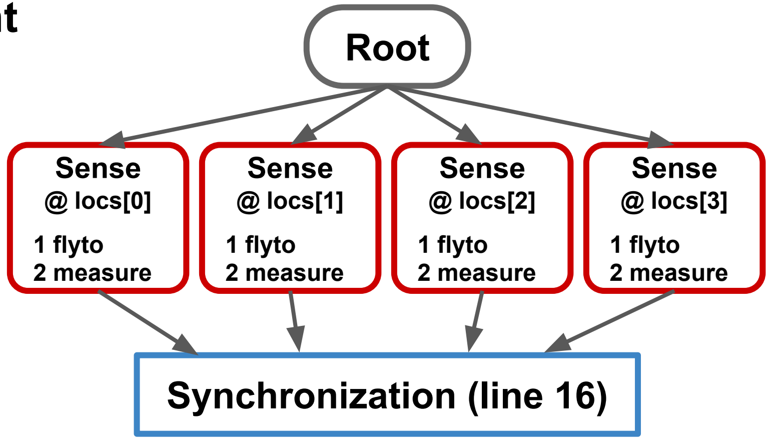
Example: Locate WiFi Hotspot Using Gradient Descent

```
1   def sense(loc):
2     act("flyto",loc)
3     handle = act("measure_WiFi_signal")
4     return handle.value # blocking
5
6   if __name__ == "__main__":
7     loc = initial_loc # e.g., (0,0)
8     for i in range(10):
9       locs = four_corners(loc)
10
11      task0 = newTask(sense, locs[0])
12      task1 = newTask(sense, locs[1])
13      task2 = newTask(sense, locs[2])
14      task3 = newTask(sense, locs[3])
15
16      values = [ task0.value, task1.value,
                   task2.value, task2.value ]
        # blocking

17      loc = GradientDescent(locs, values)
```



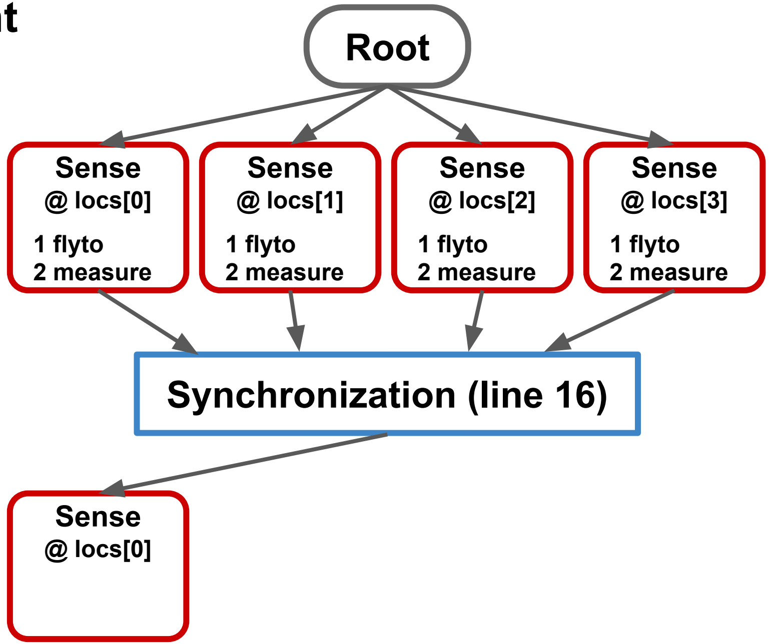
Example: Locate WiFi Hotspot Using Gradient Descent

```
1   def sense(loc):
2     act("flyto",loc)
3     handle = act("measure_WiFi_signal")
4     return handle.value # blocking
5
6   if __name__ == "__main__":
7     loc = initial_loc # e.g., (0,0)
8     for i in range(10):
9       locs = four_corners(loc)
10
11      task0 = newTask(sense, locs[0])
12      task1 = newTask(sense, locs[1])
13      task2 = newTask(sense, locs[2])
14      task3 = newTask(sense, locs[3])
15
16      values = [ task0.value, task1.value,
                   task2.value, task2.value ]
        # blocking

17      loc = GradientDescent(locs, values)
```



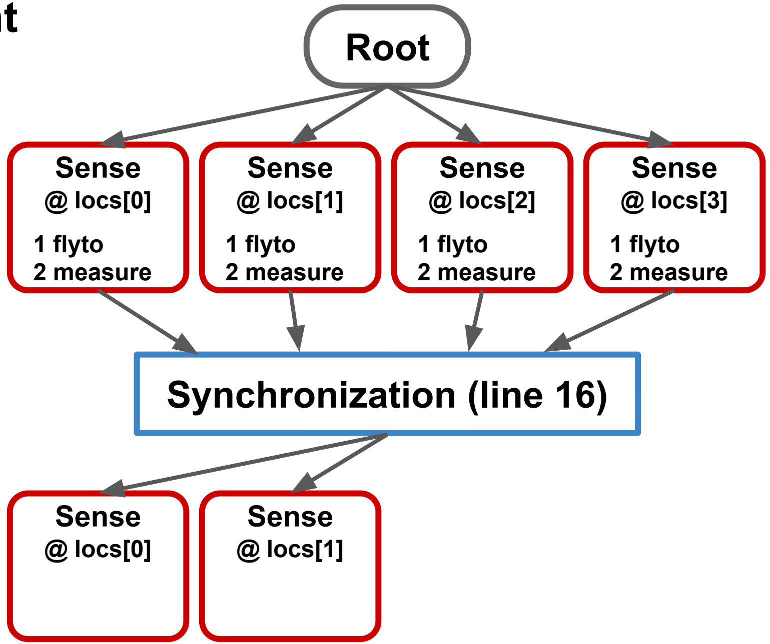
Example: Locate WiFi Hotspot Using Gradient Descent

```
1   def sense(loc):
2     act(“flyto”,loc)
3     handle = act(“measure_WiFi_signal”)
4     return handle.value # blocking
5
6   if __name__ == “__main__”:
7     loc = initial_loc # e.g., (0,0)
8     for i in range(10):
9       locs = four_corners(loc)
10
11      task0 = newTask(sense, locs[0])
12      task1 = newTask(sense, locs[1])
13      task2 = newTask(sense, locs[2])
14      task3 = newTask(sense, locs[3])
15
16      values = [ task0.value, task1.value,
                   task2.value, task2.value ]
        # blocking

17      loc = GradientDescent(locs, values)
```



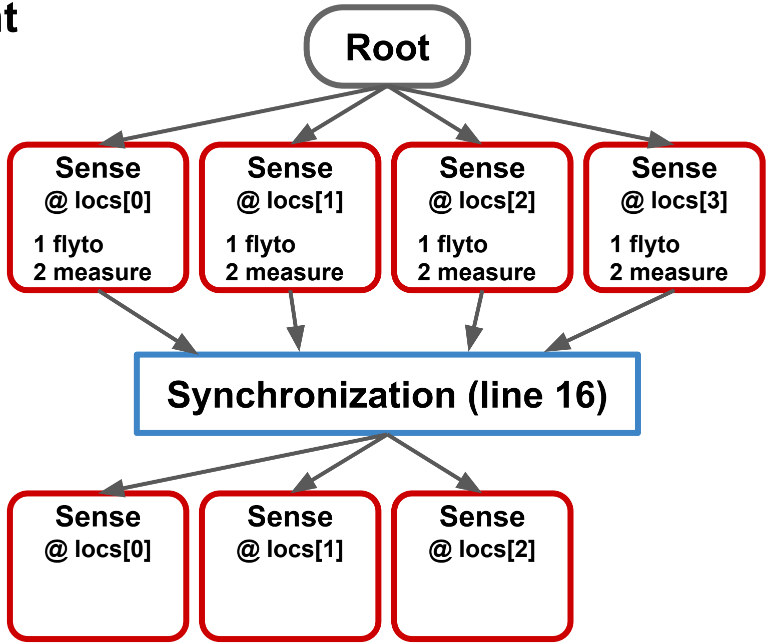
Example: Locate WiFi Hotspot Using Gradient Descent

```
1   def sense(loc):
2     act(“flyto”,loc)
3     handle = act(“measure_WiFi_signal”)
4     return handle.value # blocking
5
6   if __name__ == “__main__”:
7     loc = initial_loc # e.g., (0,0)
8     for i in range(10):
9       locs = four_corners(loc)
10
11      task0 = newTask(sense, locs[0])
12      task1 = newTask(sense, locs[1])
13      task2 = newTask(sense, locs[2])
14      task3 = newTask(sense, locs[3])
15
16      values = [ task0.value, task1.value,
                   task2.value, task2.value ]
        # blocking

17      loc = GradientDescent(locs, values)
```



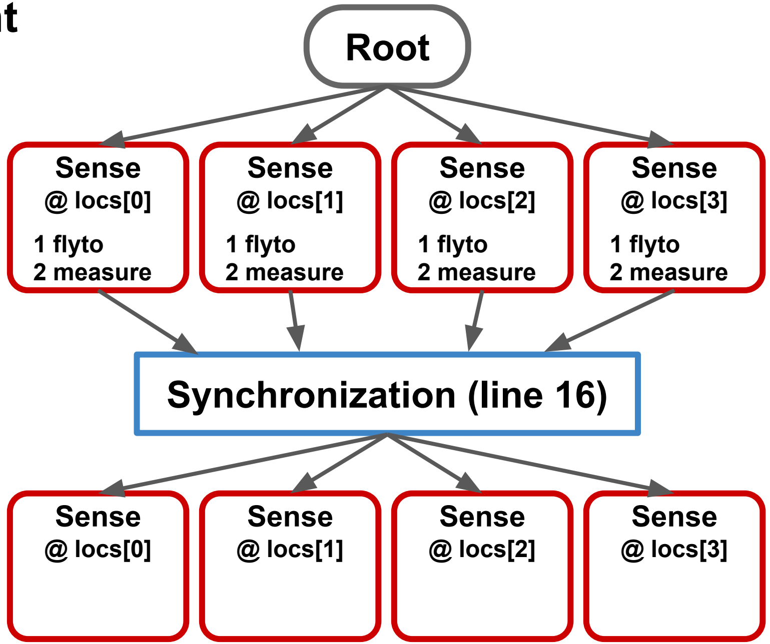
Example: Locate WiFi Hotspot Using Gradient Descent

```
1   def sense(loc):
2     act("flyto",loc)
3     handle = act("measure_WiFi_signal")
4     return handle.value # blocking
5
6   if __name__ == "__main__":
7     loc = initial_loc # e.g., (0,0)
8     for i in range(10):
9       locs = four_corners(loc)
10
11      task0 = newTask(sense, locs[0])
12      task1 = newTask(sense, locs[1])
13      task2 = newTask(sense, locs[2])
14      task3 = newTask(sense, locs[3])
15
16      values = [ task0.value, task1.value,
                   task2.value, task2.value ]
        # blocking

17      loc = GradientDescent(locs, values)
```



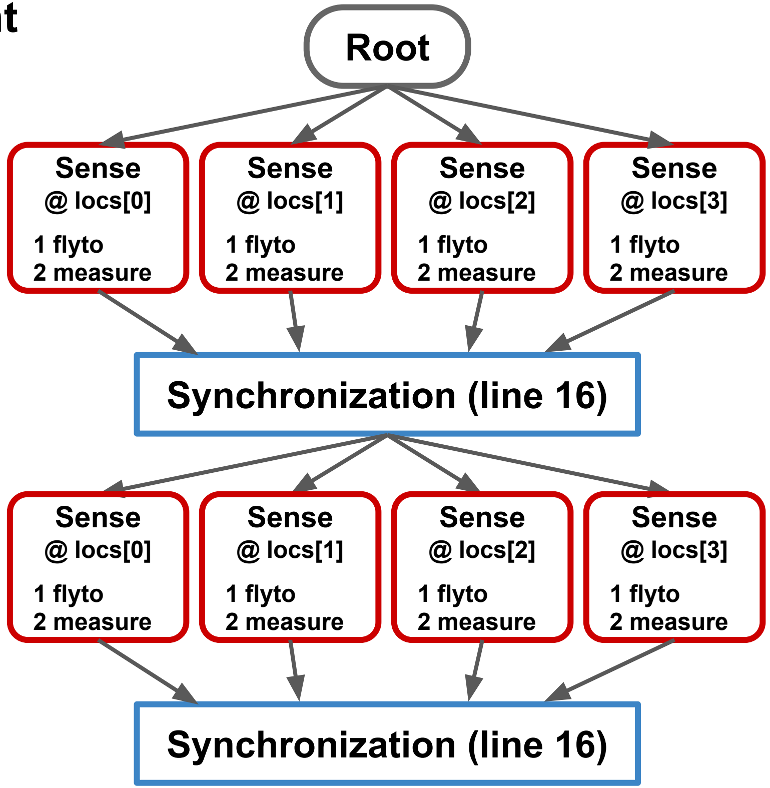
Example: Locate WiFi Hotspot Using Gradient Descent

```
1   def sense(loc):
2     act(“flyto”,loc)
3     handle = act(“measure_WiFi_signal”)
4     return handle.value # blocking
5
6   if __name__ == “__main__”:
7     loc = initial_loc # e.g., (0,0)
8     for i in range(10):
9       locs = four_corners(loc)
10
11      task0 = newTask(sense, locs[0])
12      task1 = newTask(sense, locs[1])
13      task2 = newTask(sense, locs[2])
14      task3 = newTask(sense, locs[3])
15
16      values = [ task0.value, task1.value,
                   task2.value, task2.value ]
        # blocking

17      loc = GradientDescent(locs, values)
```



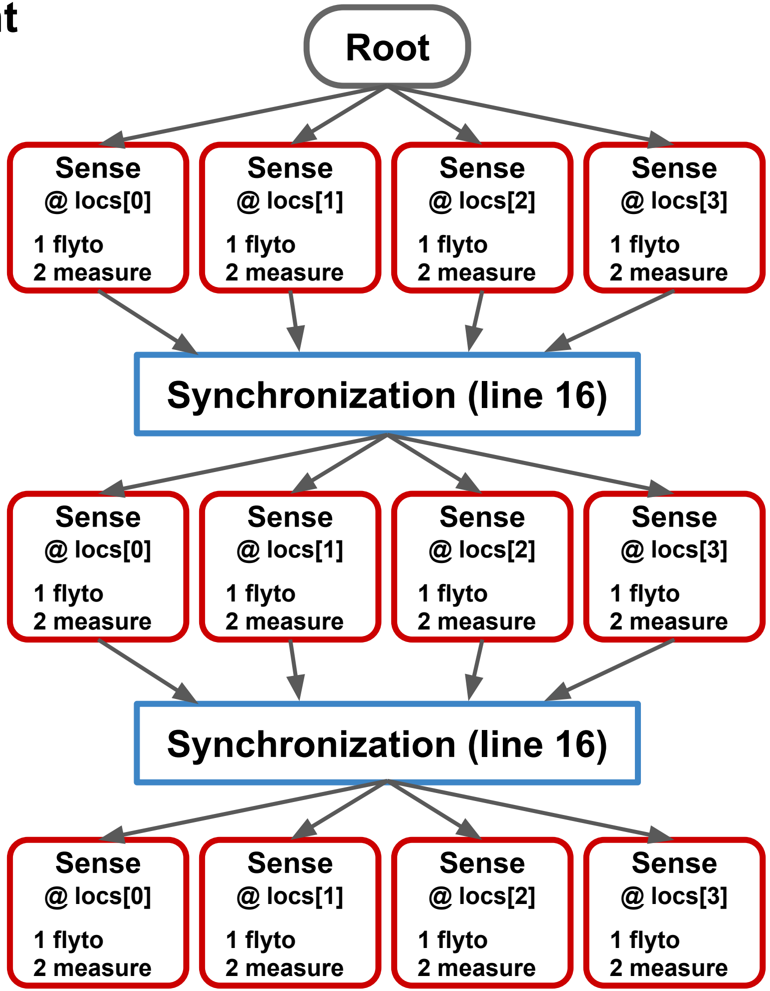
Example: Locate WiFi Hotspot Using Gradient Descent

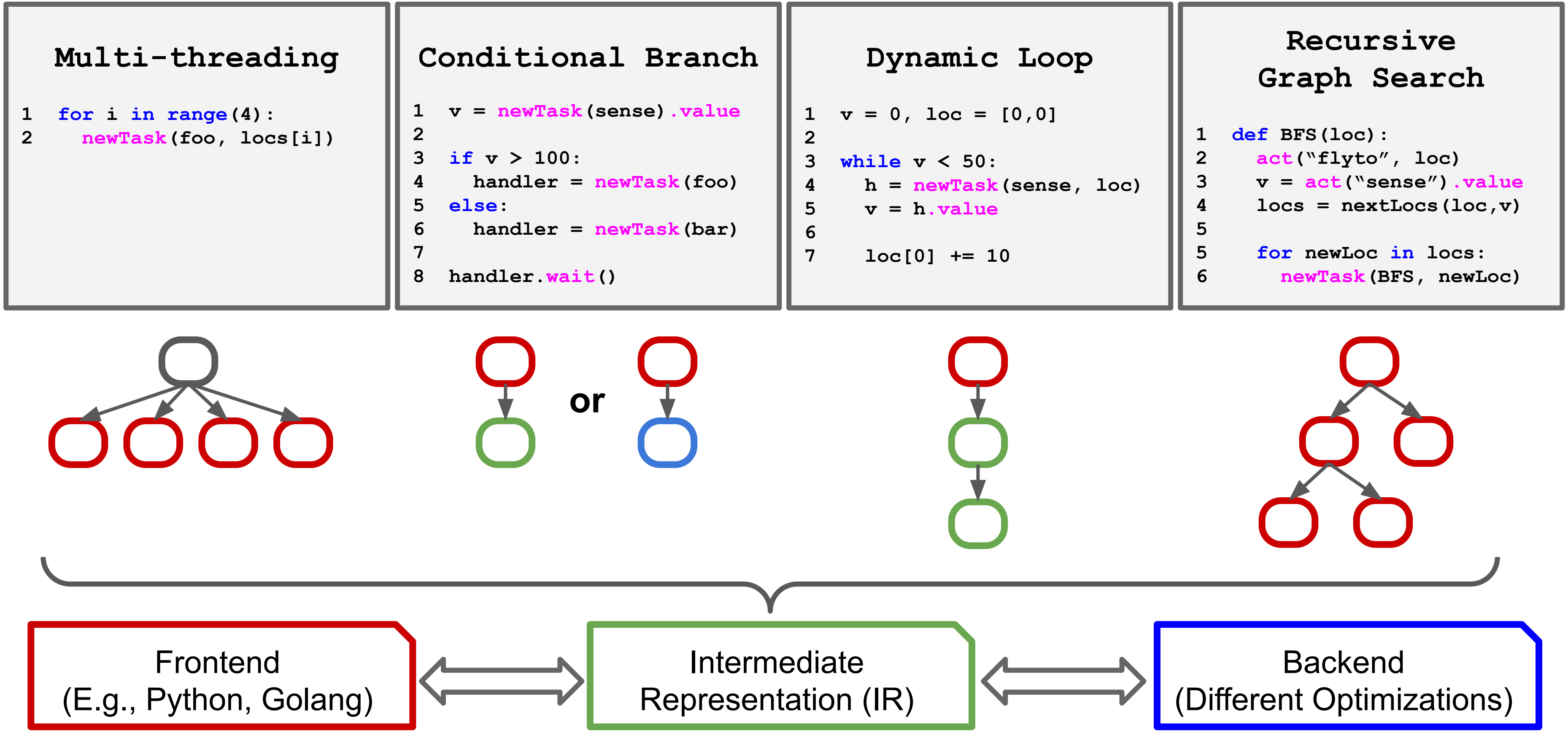
```
1   def sense(loc):
2     act("flyto",loc)
3     handle = act("measure_WiFi_signal")
4     return handle.value # blocking
5
6   if __name__ == "__main__":
7     loc = initial_loc # e.g., (0,0)
8     for i in range(10):
9       locs = four_corners(loc)
10
11      task0 = newTask(sense, locs[0])
12      task1 = newTask(sense, locs[1])
13      task2 = newTask(sense, locs[2])
14      task3 = newTask(sense, locs[3])
15
16      values = [ task0.value, task1.value,
                   task2.value, task2.value ]
        # blocking

17      loc = GradientDescent(locs, values)
```



Example: Locate WiFi Hotspot Using Gradient Descent

```
1   def sense(loc):
2     act("flyto",loc)
3     handle = act("measure_WiFi_signal")
4     return handle.value # blocking
5
6   if __name__ == "__main__":
7     loc = initial_loc # e.g., (0,0)
8     for i in range(10):
9       locs = four_corners(loc)
10
11      task0 = newTask(sense, locs[0])
12      task1 = newTask(sense, locs[1])
13      task2 = newTask(sense, locs[2])
14      task3 = newTask(sense, locs[3])
15
16      values = [ task0.value, task1.value,
                   task2.value, task2.value ]
        # blocking

17      loc = GradientDescent(locs, values)
```

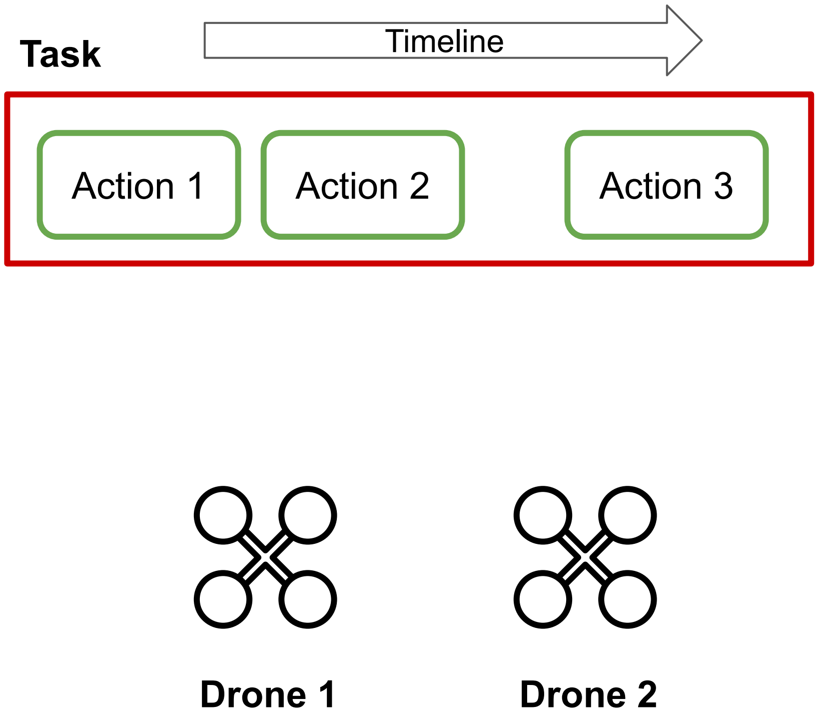



Multi-threading
for i in range(4):
newTask(foo, locs[i])
Conditional Branch
v = newTask(sense).value
if v > 100:
handler = newTask(foo)
else:
handler = newTask(bar)
handler.wait()
Dynamic Loop
v = 0, loc = [0,0]
while v < 50:
h = newTask(sense, loc)
v = h.value
loc[0] += 10
Recursive Graph Search
def BFS(loc):
act("flyto", loc)
v = act("sense").value
locs = nextLocs(loc,v)
for newLoc in locs:
newTask(BFS, newLoc)
or
Frontend
(E.g., Python, Golang)
Intermediate
Representation (IR)
Backend
(Different Optimizations)

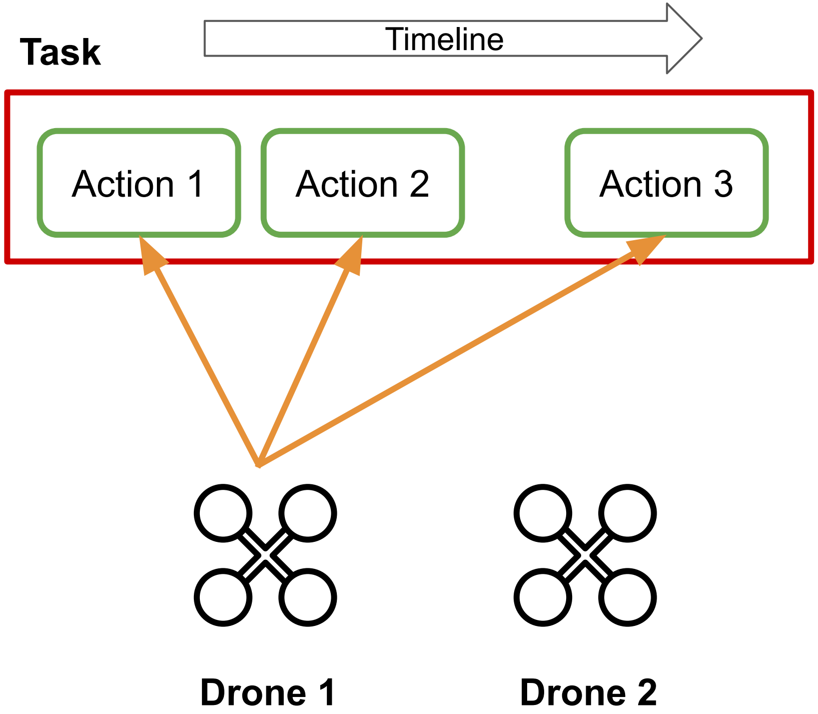
Outline

- Introduction
- Motivation
- Design of BeeCluster
  - Programming Interface
    - Programming model (dynamic tasking)
    - **Task-to-drone mapping options**
  - Predictive Optimization
- Implementation
- Case Studies

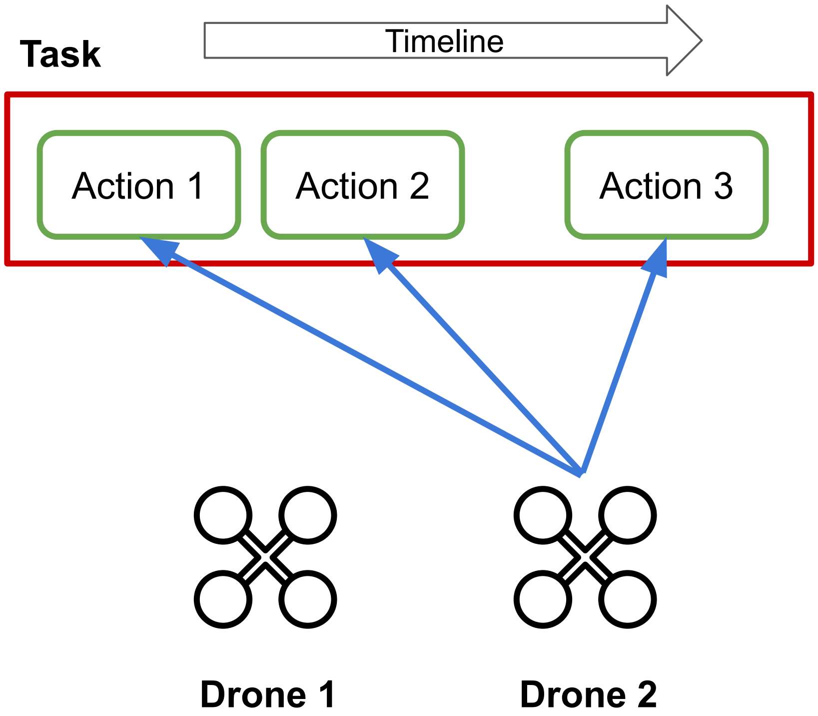
Task-to-Drone Mapping Options



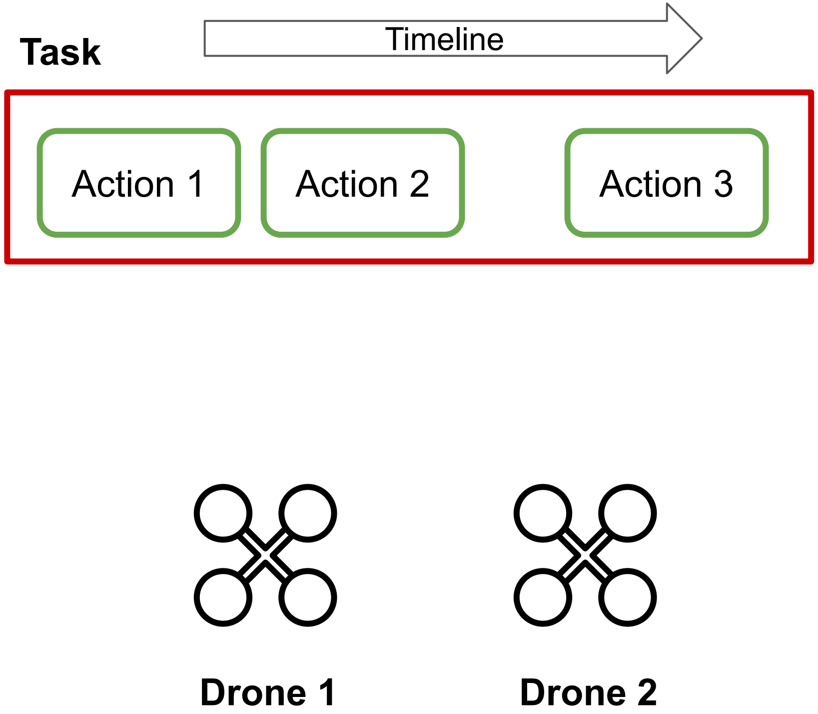
Task-to-Drone Mapping Options



Task-to-Drone Mapping Options

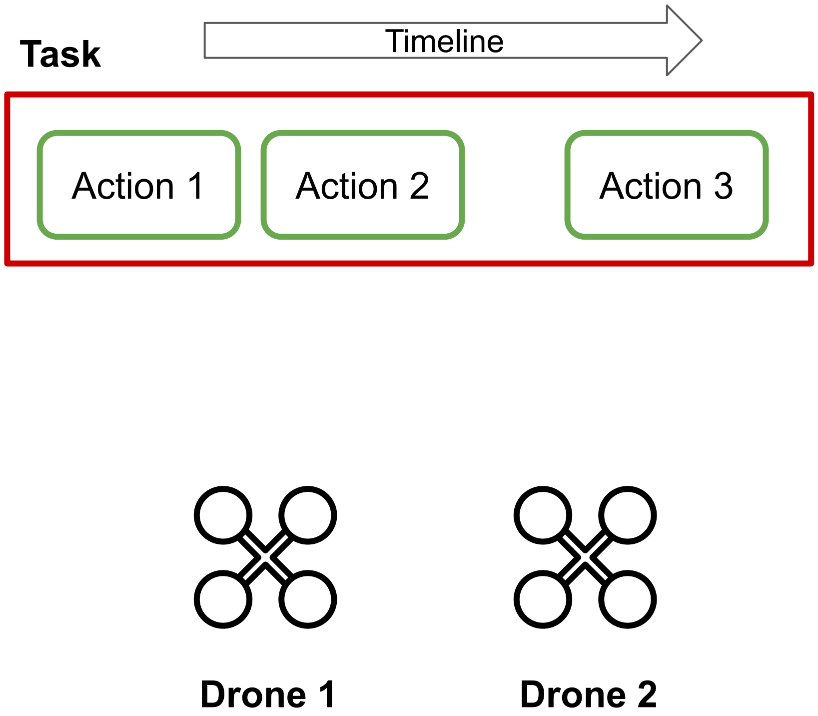


Task-to-Drone Mapping Options



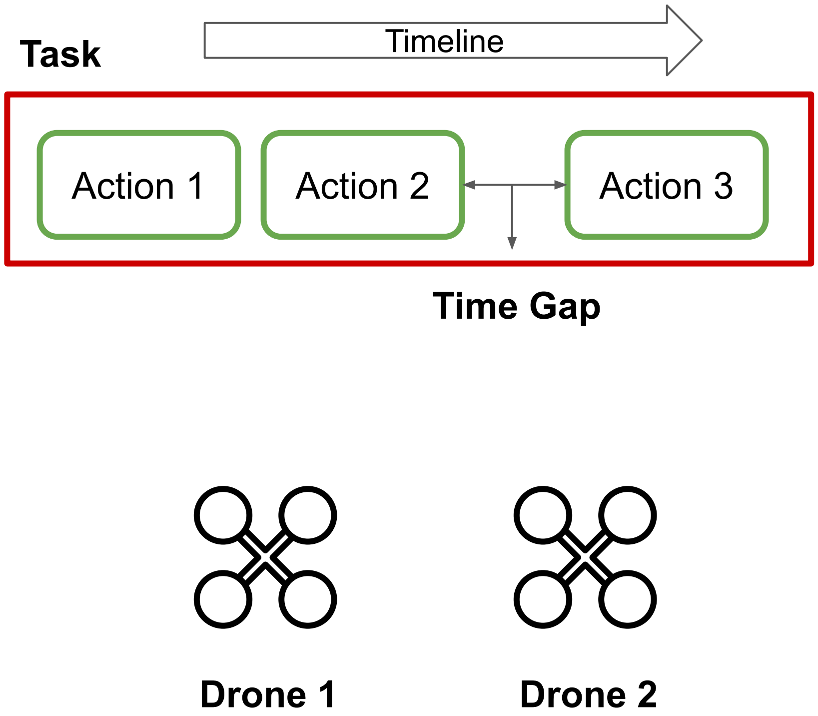
Task-to-Drone Mapping Options

- Same-Drone Flag
    - When this flag is True, all actions within the task have to be mapped to the same physical drone.
- Interruptible Flag
    - When this flag is True, the scheduler tries to minimize the time gap between two consecutive actions (best effort).



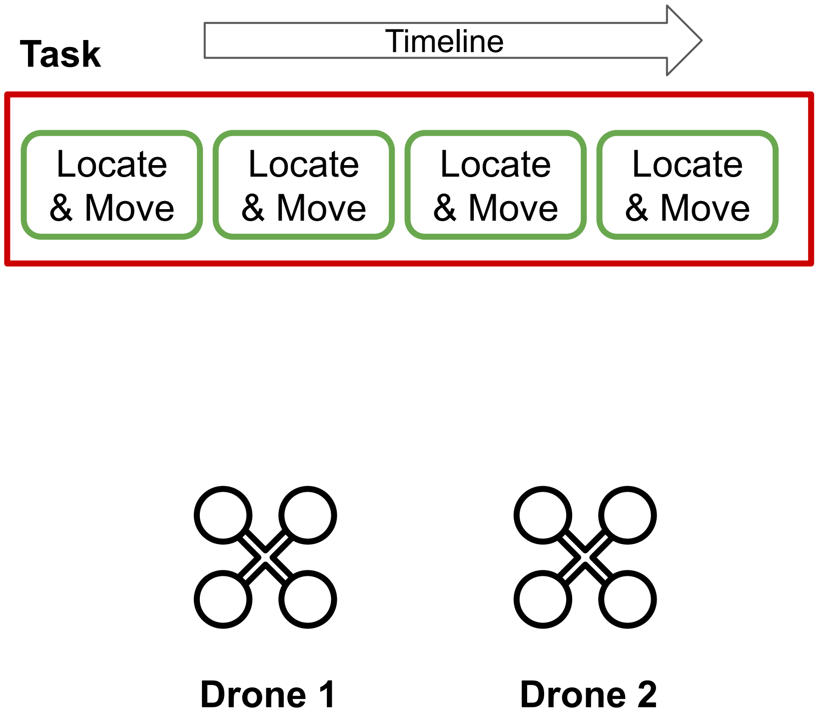
Task-to-Drone Mapping Options

- Same-Drone Flag
  - When this flag is True, all actions within the task have to be mapped to the same physical drone.
- Interruptible Flag
  - When this flag is True, the scheduler tries to minimize the time gap between two consecutive actions (best effort).



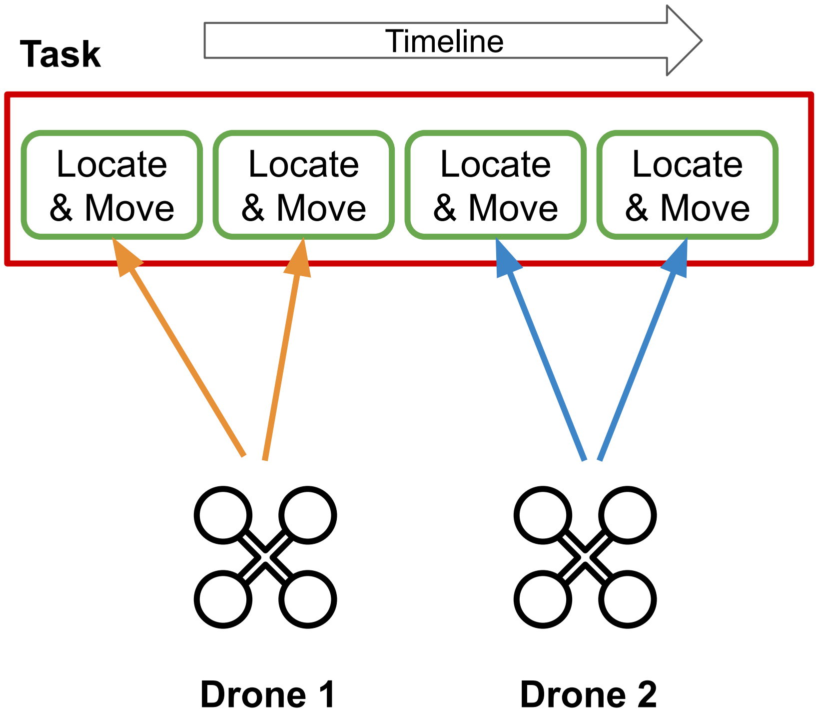
Task-to-Drone Mapping Options

- Same-Drone Flag **= False**
  - When this flag is True, all actions within the task have to be mapped to the same physical drone.
- Interruptible Flag **= False**
  - When this flag is True, the scheduler tries to minimize the time gap between two consecutive actions (best effort).



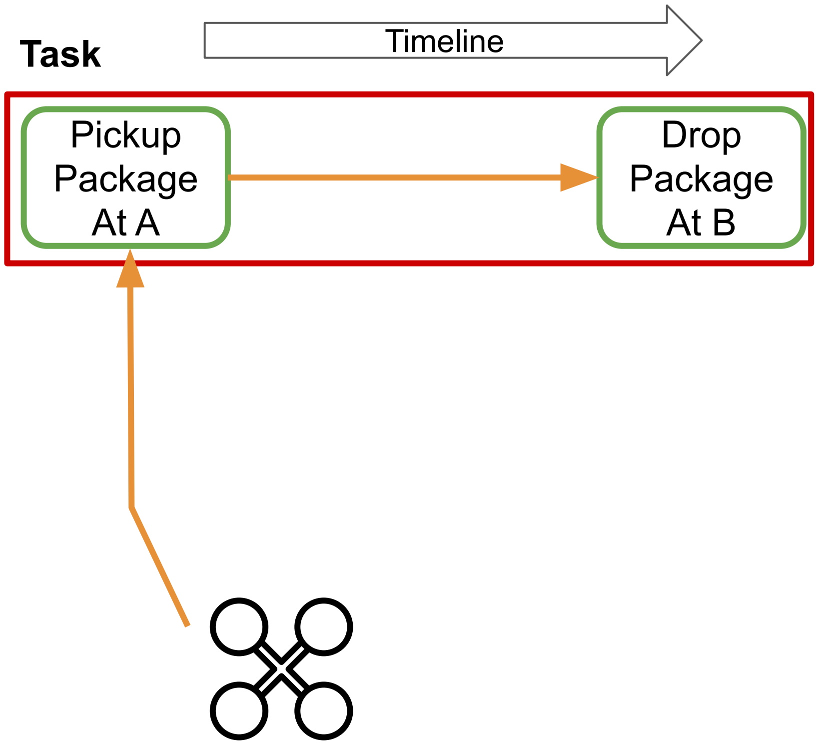
Task-to-Drone Mapping Options

- Same-Drone Flag **= False**
  - When this flag is True, all actions within the task have to be mapped to the same physical drone.
- Interruptible Flag **= False**
  - When this flag is True, the scheduler tries to minimize the time gap between two consecutive actions (best effort).



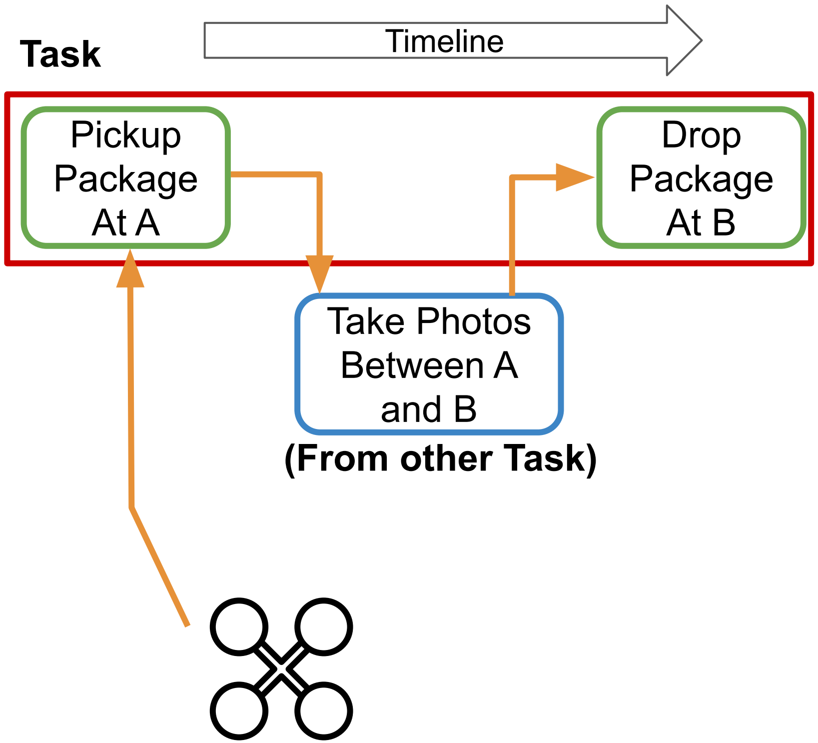
Task-to-Drone Mapping Options

- Same-Drone Flag **= True**
  - When this flag is True, all actions within the task have to be mapped to the same physical drone.
- Interruptible Flag
  - When this flag is True, the scheduler tries to minimize the time gap between two consecutive actions (best effort).



Task-to-Drone Mapping Options

- Same-Drone Flag **= True**
  - When this flag is True, all actions within the task have to be mapped to the same physical drone.
- Interruptible Flag **= True**
  - When this flag is True, the scheduler tries to minimize the time gap between two consecutive actions (best effort).



Four Task-to-Drone Mapping Options

Same-Drone Flag = **True**
Interruptible Flag = **True**

Package Delivery

1. Pickup Package at Location A
2. Drop Package at Location B

Same-Drone Flag = **True**
Interruptible Flag = **False**

Recording Videos

1. Fly to Location A
2. Start Video Recording
3. Fly to Location B
4. Stop Video Recording

Same-Drone Flag=**False**
Interruptible Flag = **True**

Repeated Measures

1. Fly to Location A
2. Measure WiFi Signal
3. Measure WiFi Signal
4. Measure WiFi Signal

Same-Drone Flag=**False**
Interruptible Flag= **False**

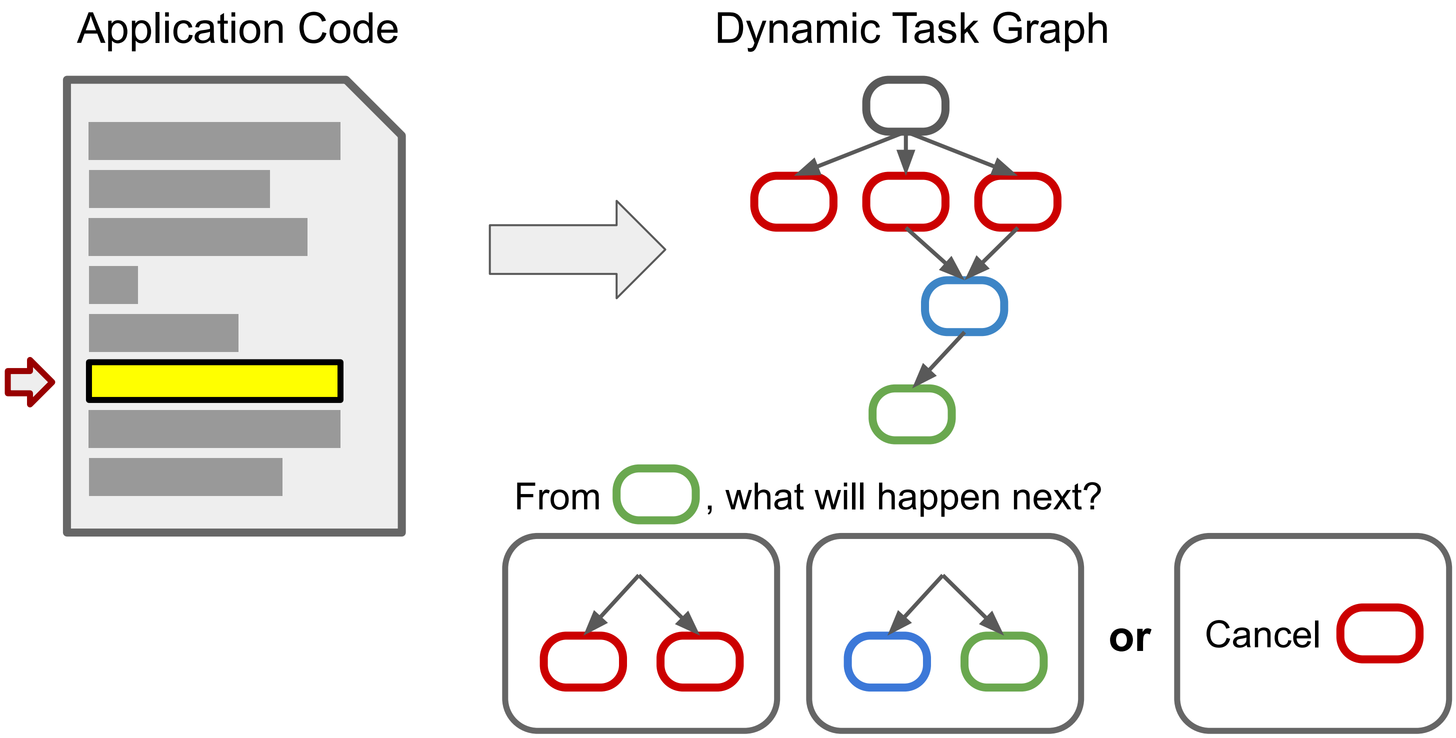
Object Tracking

1. Locate and Move
2. Locate and Move
3. Locate and Move
4. Locate and Move

Outline

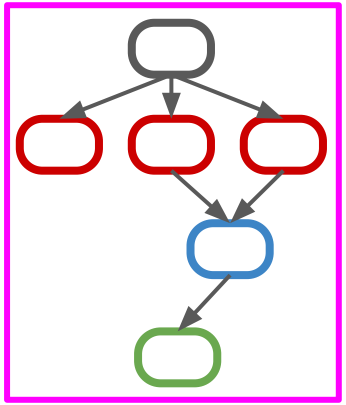
- Introduction
- Motivation
- Design of BeeCluster
    - Programming Interface
        - Programming model (dynamic tasking)
        - Task-to-drone mapping options
    - **Predictive Optimization**
- Implementation
- Case Studies

Predictive Optimization - Application Prediction

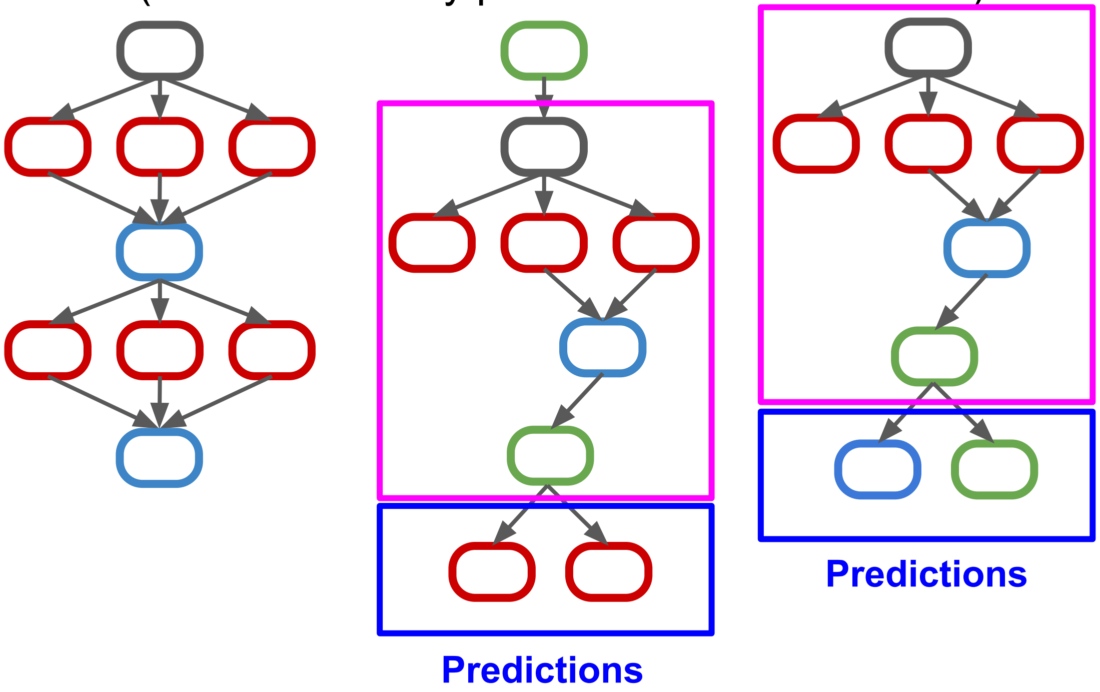


Predictive Optimization - Application Prediction

Current Dynamic Task Graph



Historical Dynamic Task Graphs
(Include the early portion of the current DTG)



Predictive Optimization - Scheduling

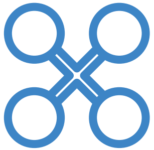
Scheduling Model

- Instantaneous assignment model

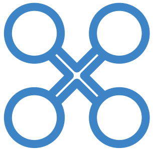
Predictive Optimization - Scheduling

Scheduling Model

- Instantaneous assignment model



Drone

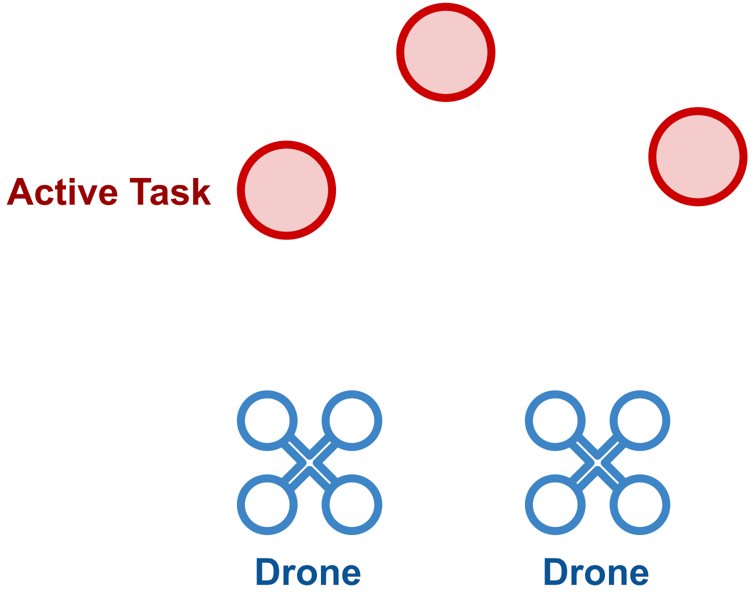


Drone

Predictive Optimization - Scheduling

Scheduling Model

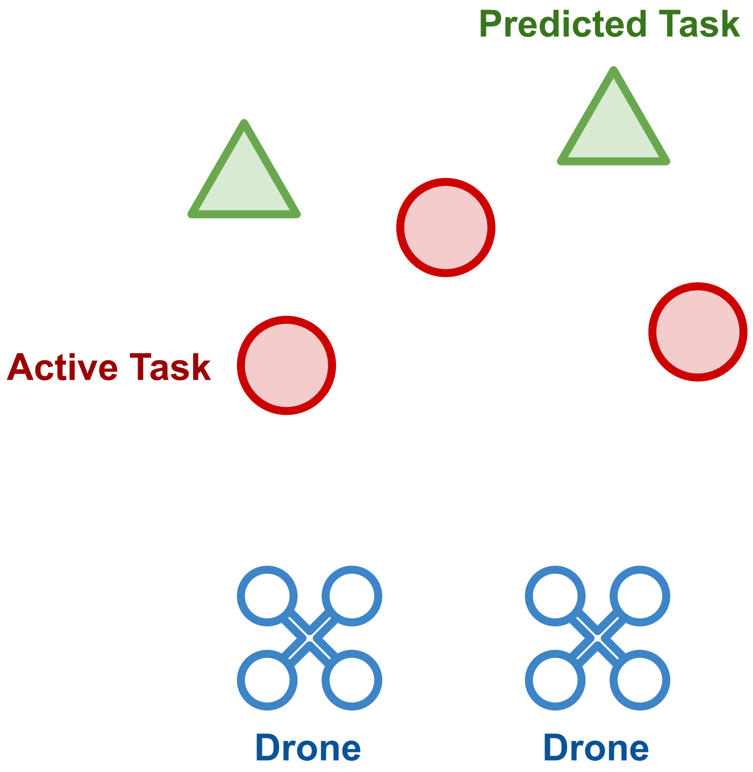
- Instantaneous assignment model



Predictive Optimization - Scheduling

Scheduling Model

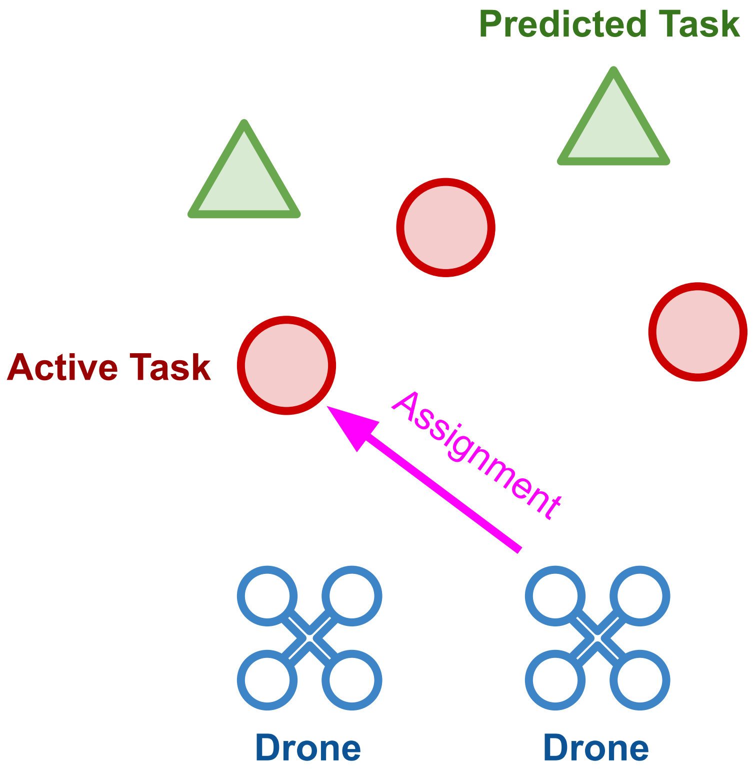
- Instantaneous assignment model



Predictive Optimization - Scheduling

Scheduling Model

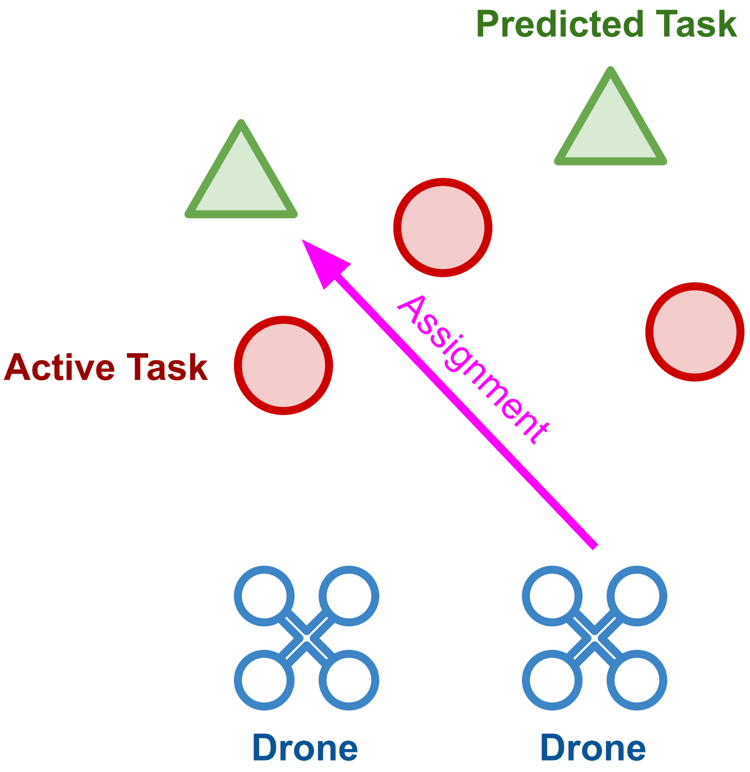
- Instantaneous assignment model



Predictive Optimization - Scheduling

Scheduling Model

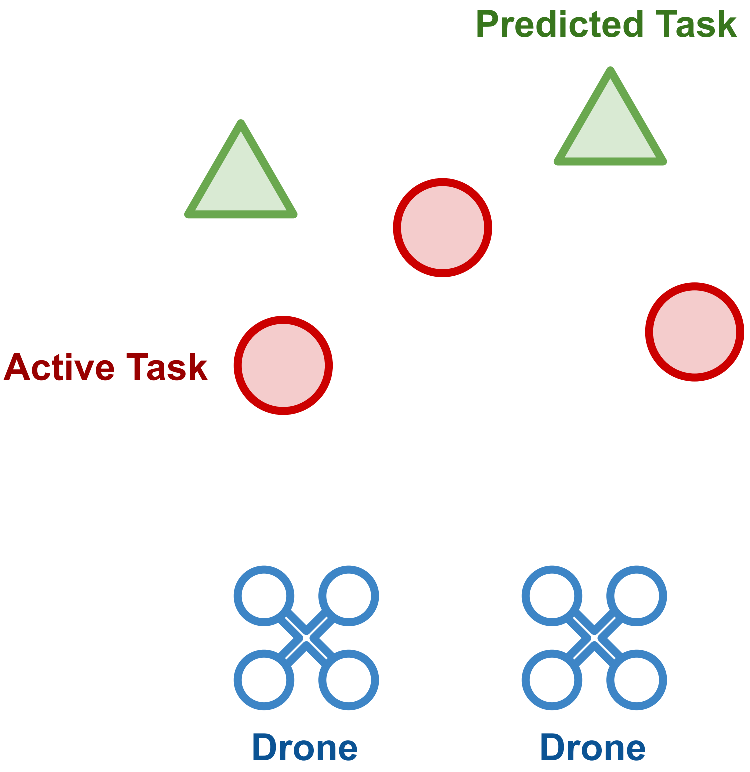
- Instantaneous assignment model



Predictive Optimization - Scheduling

Scheduling Model

- Instantaneous assignment model

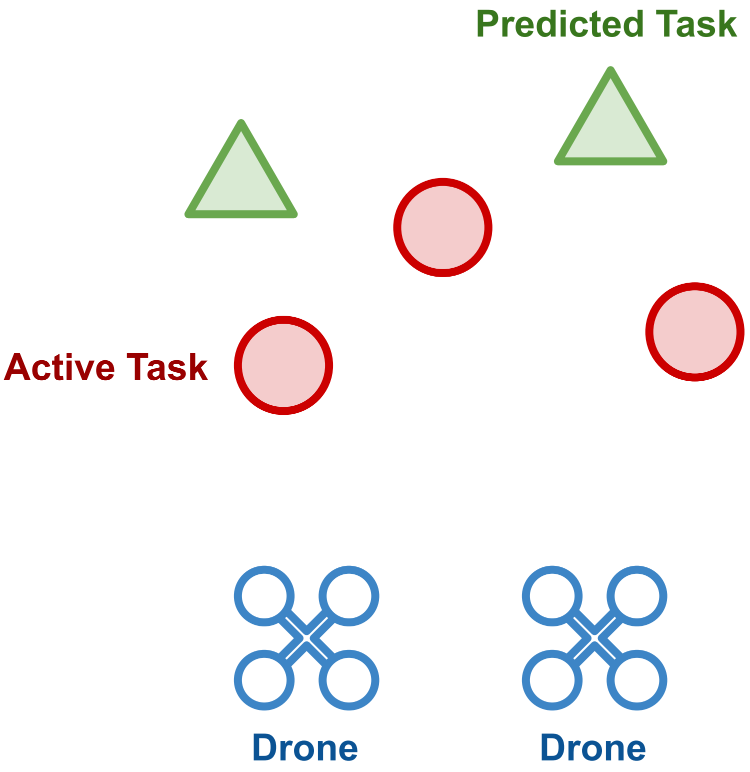


Predictive Optimization - Scheduling

Scheduling Model

- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm

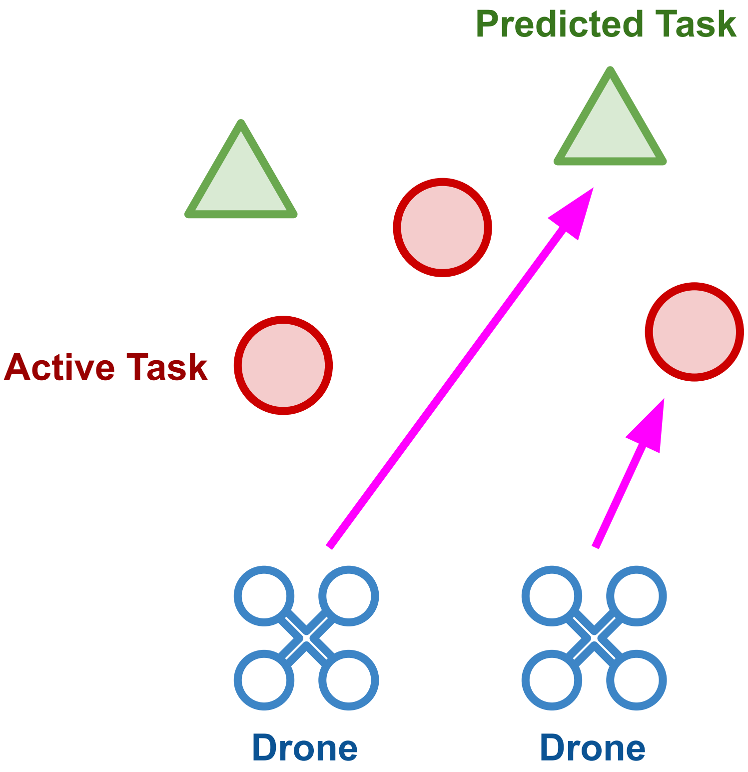


Predictive Optimization - Scheduling

Scheduling Model

- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm

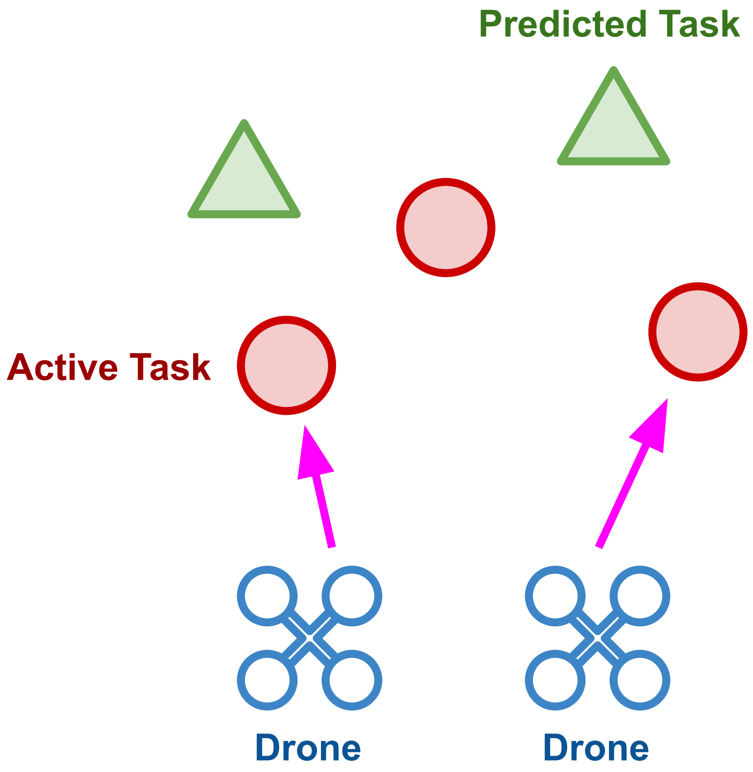


Predictive Optimization - Scheduling

Scheduling Model

- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm

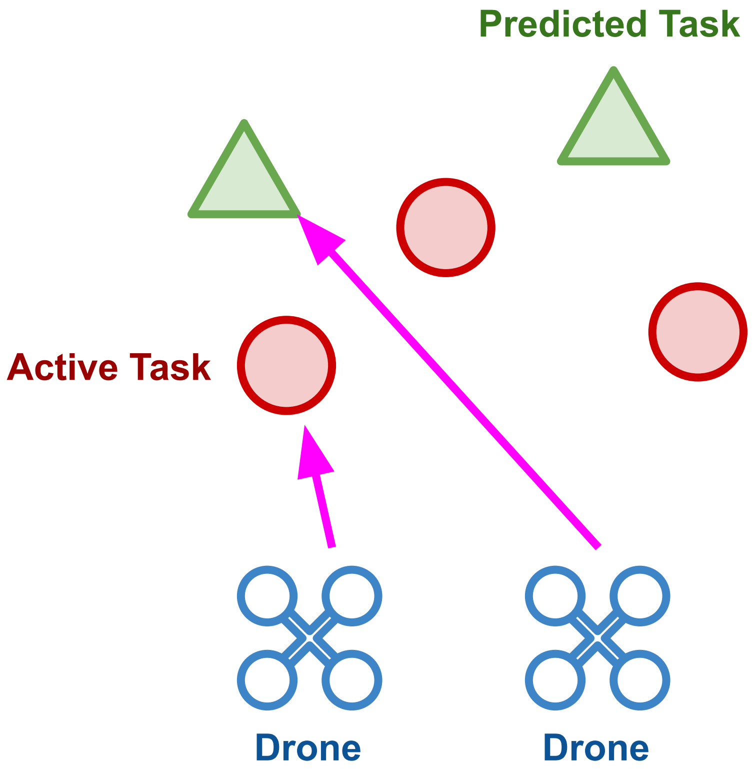


Predictive Optimization - Scheduling

Scheduling Model

- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm

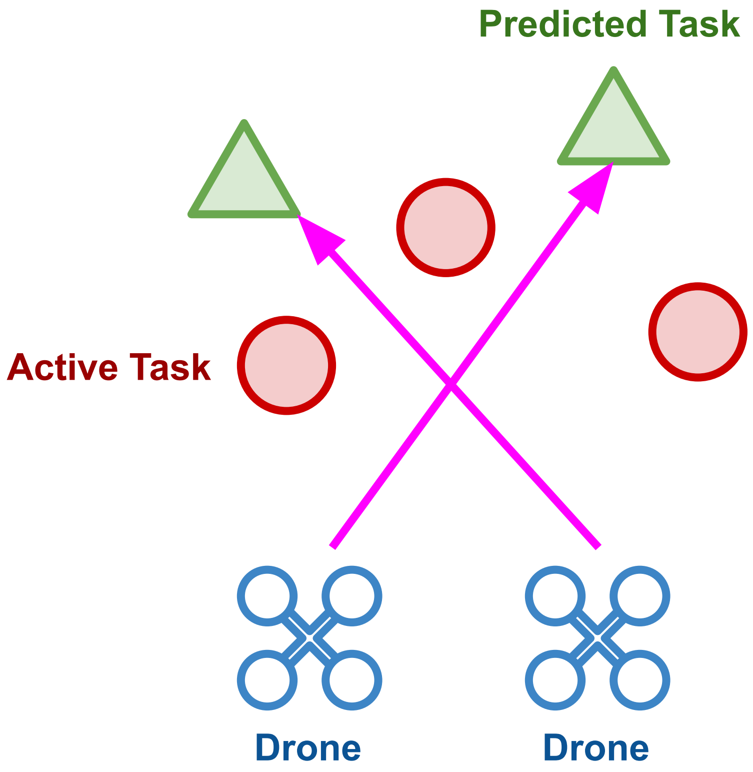


Predictive Optimization - Scheduling

Scheduling Model

- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm

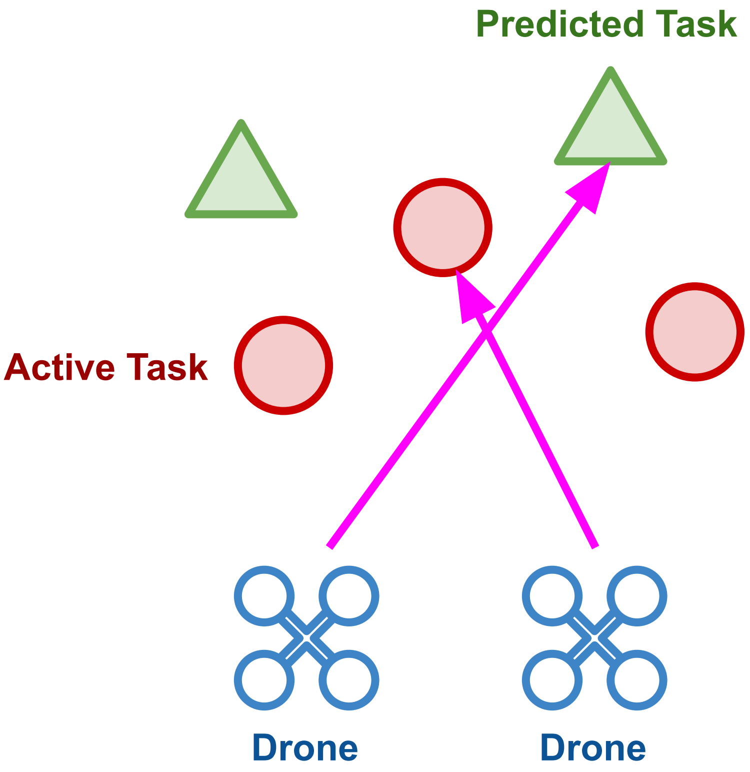


Predictive Optimization - Scheduling

Scheduling Model

- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm

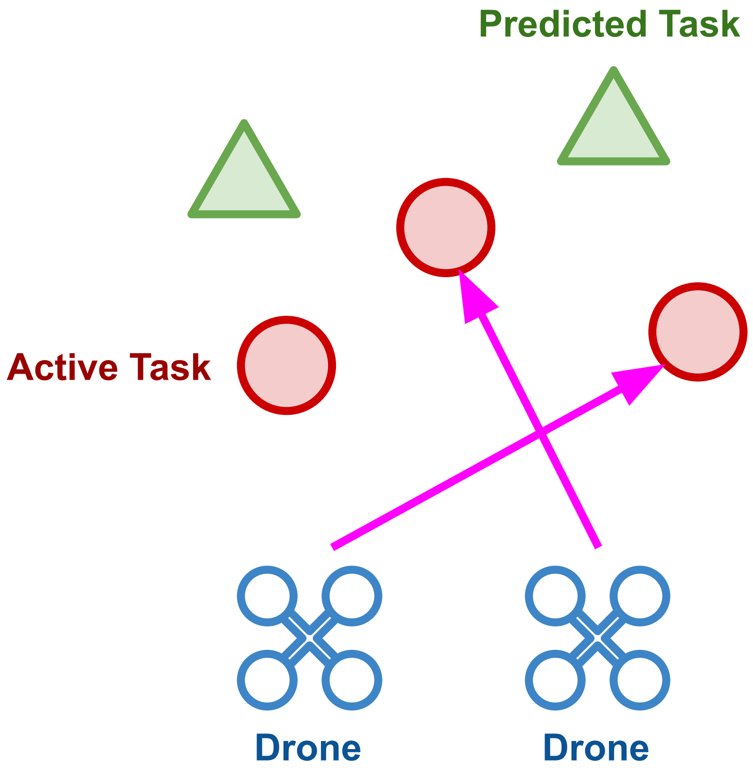


Predictive Optimization - Scheduling

Scheduling Model

- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm

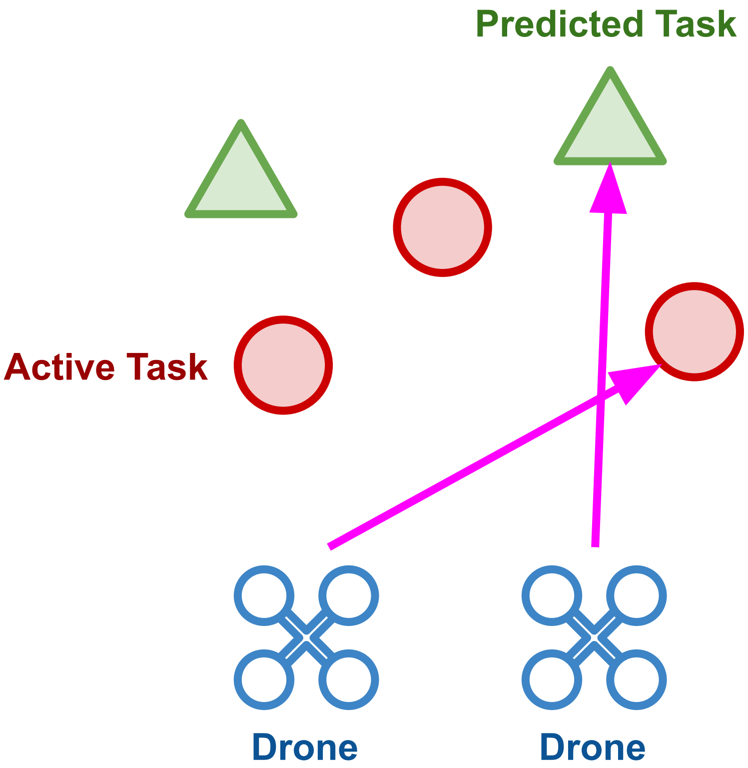


Predictive Optimization - Scheduling

Scheduling Model

- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm

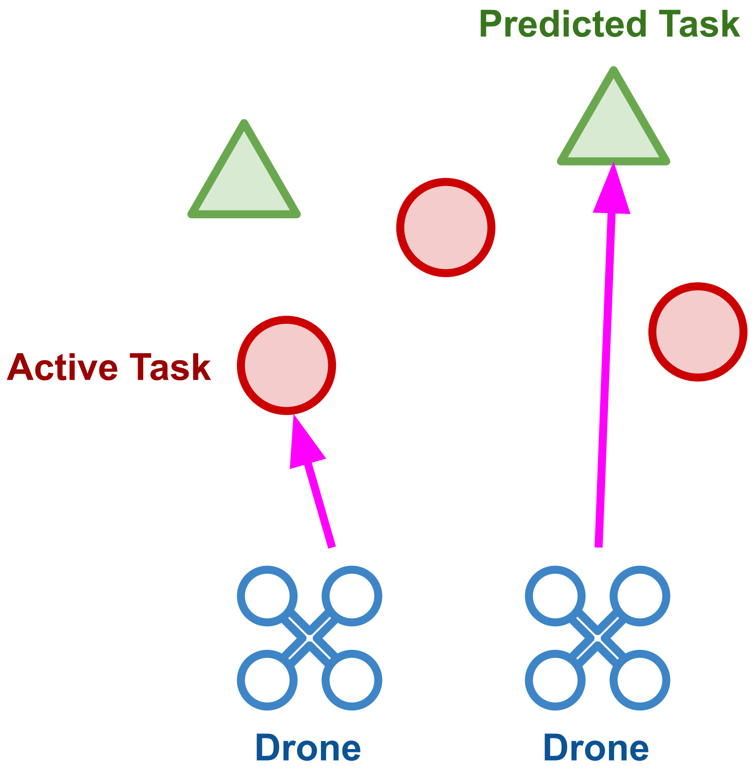


Predictive Optimization - Scheduling

Scheduling Model

- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm

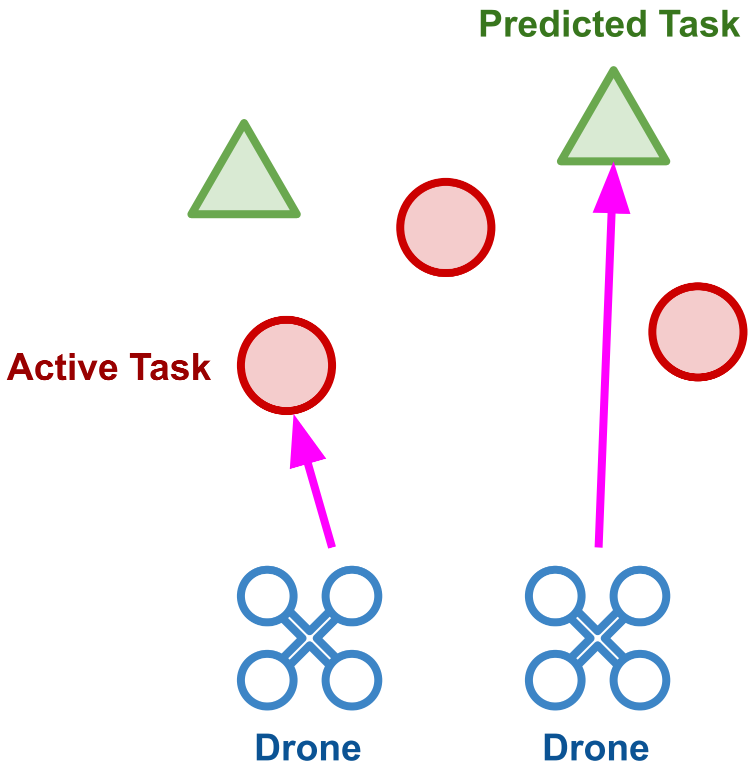


Predictive Optimization - Scheduling

Scheduling Model

- Instantaneous assignment model

Searching the best assignment using a simulated-annealing algorithm

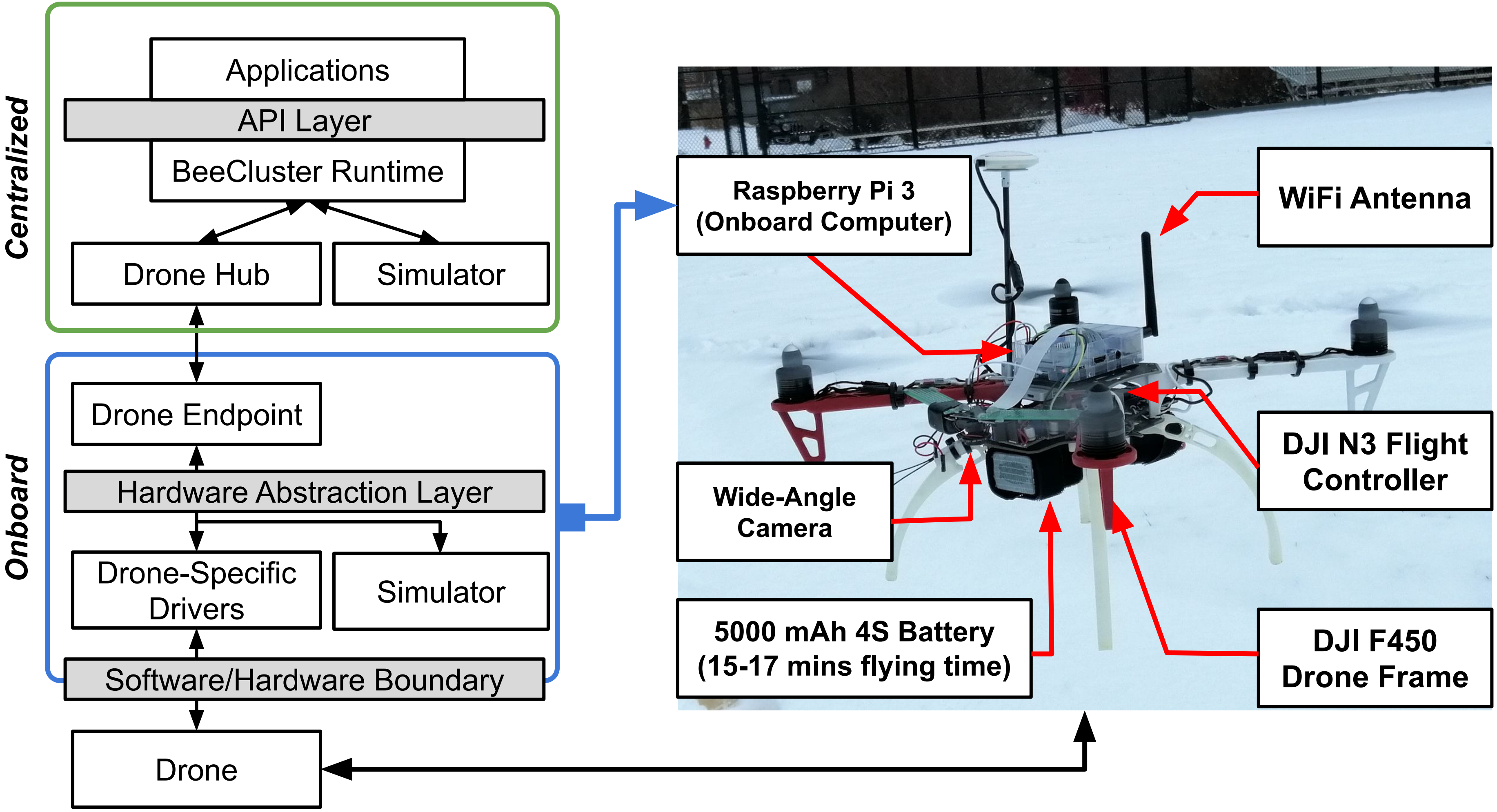


Outline

- Introduction
- Motivation
- Design of BeeCluster
  - Programming Interface
  - Predictive Optimization
- **Implementation**
- Case Studies

BeeCluster Implementation

- Software
  - 15K lines of code.
    - 80.8% in Golang for the BeeCluster core system.
    - 5.3% in python for the python wrapper of BeeCluster API.
    - 13.9% in C++ for the drone driver.
- Hardware



Centralized
Applications
API Layer
BeeCluster Runtime
Drone Hub
Simulator
Onboard
Drone Endpoint
Hardware Abstraction Layer
Drone-Specific Drivers
Simulator
Software/Hardware Boundary
Drone
Raspberry Pi 3 (Onboard Computer)
WiFi Antenna
DJI N3 Flight Controller
Wide-Angle Camera
5000 mAh 4S Battery (15-17 mins flying time)
DJI F450 Drone Frame

Case Study 1 - WiFi Hotspot Localization

Case Study 1 - WiFi Hotspot Localization

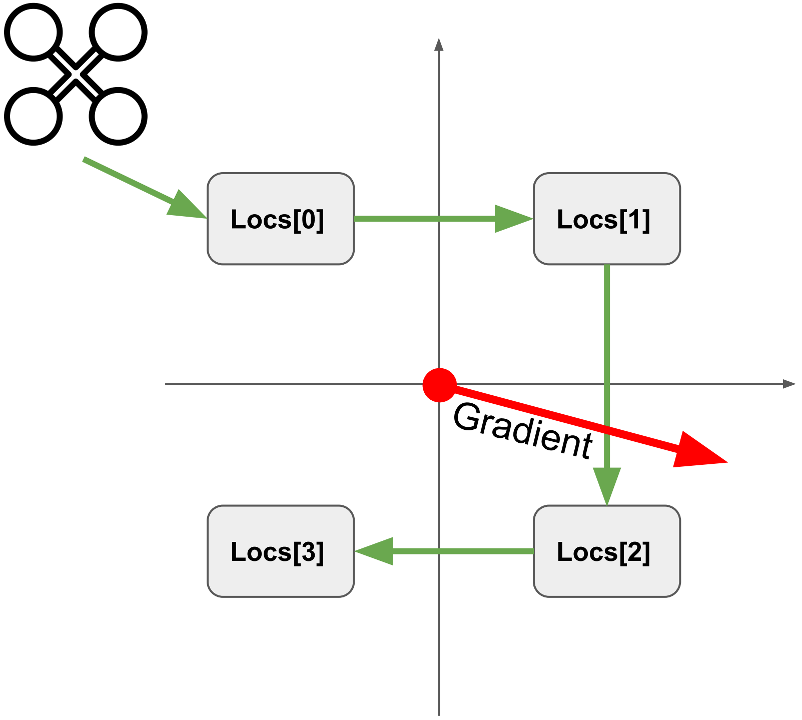
```
def sense(loc):
  act(“flyto”,loc)
  handle = act(“measure_WiFi_signal”)
  return handle.value # blocking

if __name__ == “__main__”:
  loc = initial_loc # e.g., (0,0)
  for i in range(10):
    locs = four_corners(loc, “10 meters”)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```

Case Study 1 - WiFi Hotspot Localization

```
def sense(loc):
  act(“flyto”,loc)
  handle = act(“measure_WiFi_signal”)
  return handle.value # blocking

if __name__ == “__main__”:
  loc = initial_loc # e.g., (0,0)
  for i in range(10):
    locs = four_corners(loc, “10 meters”)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```

Case Study 1 - WiFi Hotspot Localization

```
def sense(loc):
  act(“flyto”,loc)
  handle = act(“measure_WiFi_signal”)
  return handle.value # blocking

if __name__ == “__main__”:
  loc = initial_loc # e.g., (0,0)
  for i in range(10):
    locs = four_corners(loc, “10 meters”)

    task0 = newTask(sense, locs[0])
    task1 = newTask(sense, locs[1])
    task2 = newTask(sense, locs[2])
    task3 = newTask(sense, locs[3])

    values = [ task0.value, task1.value,
               task2.value, task2.value ]
    # blocking

    loc = GradientDescent(locs, values)
```



Case Study 1 - WiFi Hotspot Localization

Case Study 1 - WiFi Hotspot Localization

Local optimal
but not global optimal

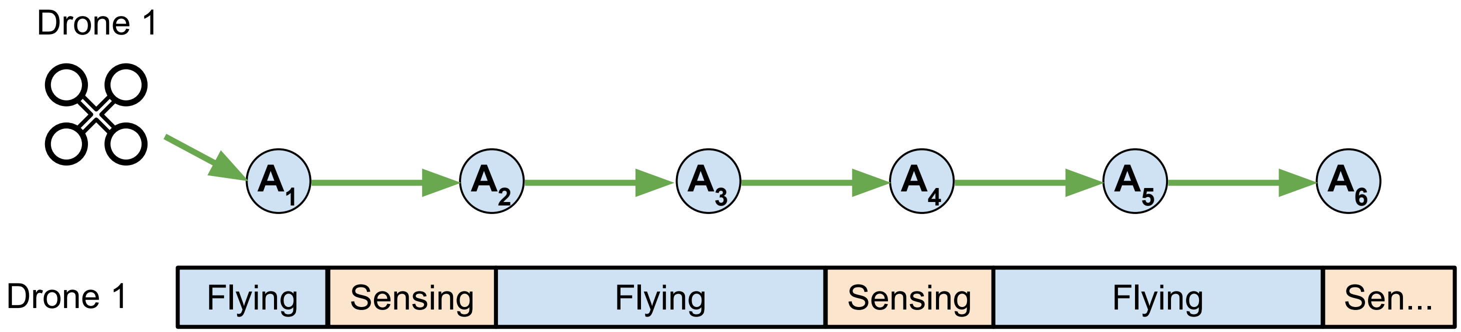
Case Study 1 - WiFi Hotspot Localization

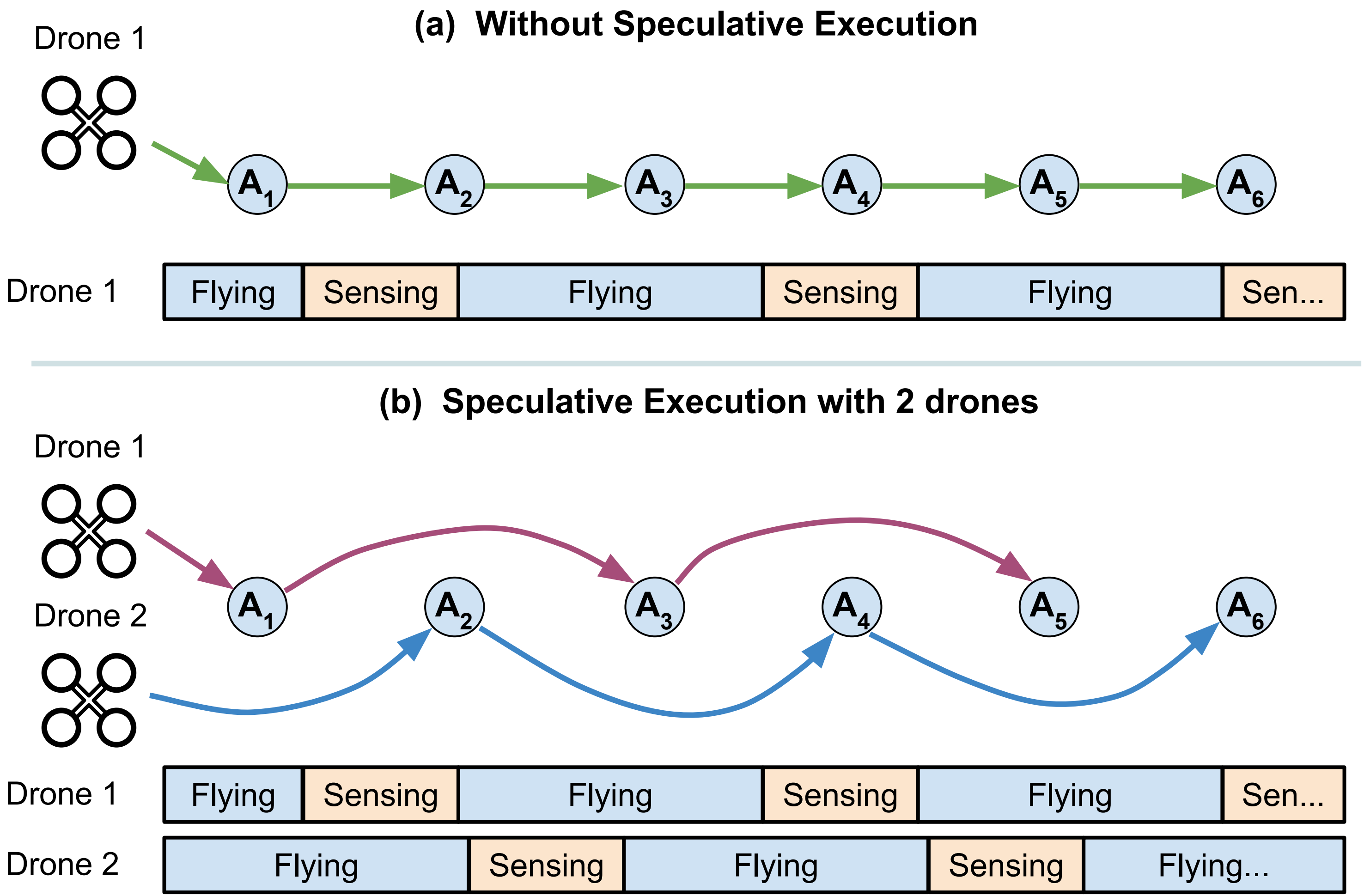
Local optimal
but not global optimal

Global optimal
with predictive optimization

Case Study 2 - Mapping Wifi Coverage

Case Study 3 - Mapping New Roads


Drone 1
A_1
A_2
A_3
A_4
A_5
A_6
Drone 1
Flying
Sensing
Flying
Sensing
Flying
Sen...



(a) Without Speculative Execution
Drone 1
A1
A2
A3
A4
A5
A6
Drone 1
Flying
Sensing
Flying
Sensing
Flying
Sen...
(b) Speculative Execution with 2 drones
Drone 1
Drone 2
A1
A2
A3
A4
A5
A6
Drone 1
Flying
Sensing
Flying
Sensing
Flying
Sen...
Drone 2
Flying
Sensing
Flying
Sensing
Flying...

BeeCluster Drone Orchestration Platform

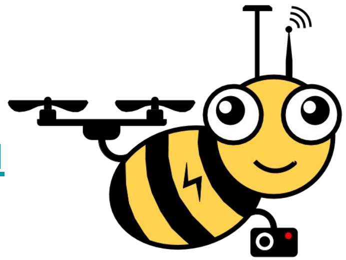
Goal:

- Make it easy to build cross-platform and optimized drone applications.

Key Features:

- Dynamic Tasking
- Rich Task-to-Drone Mappings
- Predictive Optimization

Open-Source Project



Website:
beecluster.csail.mit.edu

GitHub:
github.com/songtaohe/BeeCluster

Website

GitHub