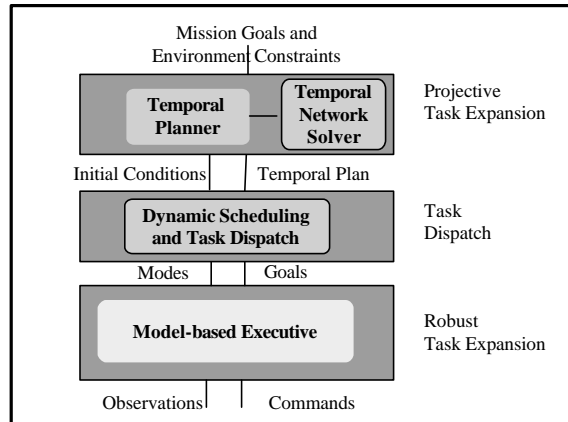


## Robust Task Execution: Procedural and Model-based

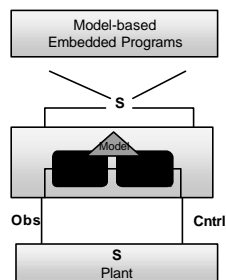
Brian C. Williams  
16.412J/6.834J  
March 14<sup>th</sup>, 2005



## Desiderata: Robust Task-level Execution

Create Languages that are:

- Suspicious
  - Monitor intentions and plans
- Self-Adaptive
  - Exploits and generates contingencies
- Anticipatory
  - Predicts, plans and verifies into future
- State Aware
  - Commanded with desired state
- Fault Aware
  - Reasons about and responds to failure



## Outline

- Safe Procedural Execution
- Model-Predictive Dispatch
- Model-based Reactive Planning

## Robust Task Execution: RAPS [Firby PhD]

- RAPS Monitors Success Against Spec

```
(define-rap (move-to thing place)
  (succeed (LOCATION thing place))
  (method
    (context (and (LOCATION thing loc)
                  (not (= loc UNKNOWN))))
    (task-net
      (t0 (goto loc) ((TRUCK -LOCATION loc) for t1))
      (t1 (pickupthing)((TRUCK -HOLDING thing) for t2)
          ((TRUCK -HOLDING thing) for t3))
      (t2 (goto place) ((TRUCK -LOCATION place) for t3))
      (t3 (putdown thing))))
    (method
      (context (LOCATION thing UNKNOWN))
      (task-net
        (t0 (goto WAREHOUSE))))))
```

## Robust Task Execution: RAPS [Firby PhD]

- RAPS Exploits contingencies by performing functionally redundant method selection

```
(define-rap (move-to thing place)
  (succeed (LOCATION thing place))
  (method
    (context (and (LOCATION thing loc)
                  (not (= loc UNKNOWN))))
    (task-net
      (t0 (goto loc) ((TRUCK -LOCATION loc) for t1))
      (t1 (pickupthing)((TRUCK -HOLDING thing) for t2)
          ((TRUCK -HOLDING thing) for t3))
      (t2 (goto place) ((TRUCK -LOCATION place) for t3))
      (t3 (putdown thing))))
    (method
      (context (LOCATION thing UNKNOWN))
      (task-net
        (t0 (goto WAREHOUSE))))))
```

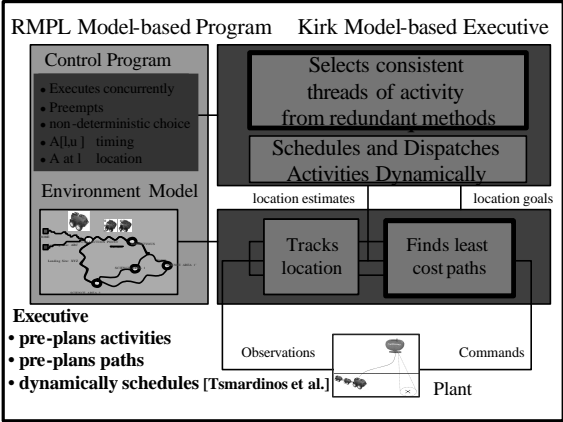
### Robust Task Execution: RAPS [Firby PhD]

---

- RAPS Exploits contingencies by performing functionally redundant method selection
  - Methods are chosen based on the current situation.
  - If a method fails, another is tried instead.
  - Tasks do not complete until satisfied.
  - Methods can include monitoring subtasks to deal with contingencies and opportunities.

➤ **Methods selected reactively**  
 ⇨ Model-predictive dispatch

➤ **Goals explicitly observable and controllable**  
 ⇨ Model-based execution



### Outline

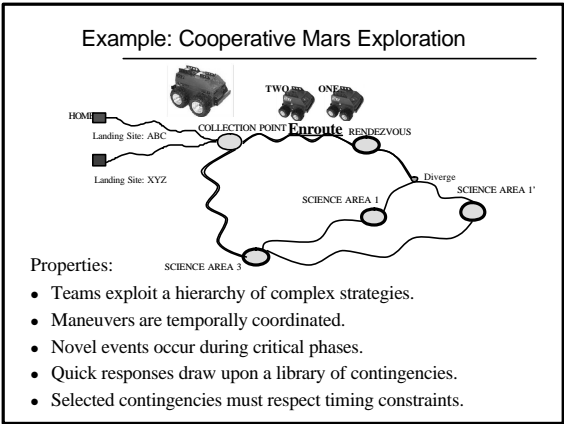
---

- Safe Procedural Execution
- Model-Predictive Dispatch
  - Model-based Programming
  - Temporal Plan Networks (TPN)
  - Activity Planning (Kirk)
  - Unifying Activity and Path Planning
- Model-based Reactive Planning

### Example: Cooperative Mars Exploration

---

How do we coordinate heterogeneous teams of orbiters, rovers and air vehicles to perform globally optimal science exploration?



### Reactive Model-based Programming

---

Idea: Describe team behaviors by starting with a rich concurrent, embedded programming language (RMPL, TCC, Esterel):

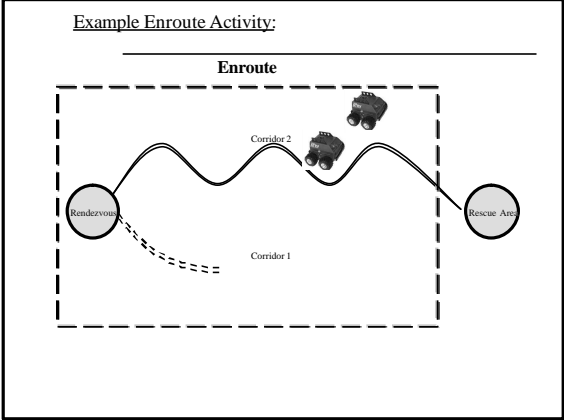
- c
- If c next A
- Unless c next A
- A, B
- Always A
- Sensing/actuation activities
- Conditional execution
- Preemption
- Full concurrency
- Iteration

Add temporal constraints:

- A [L,u]
- Timing

Add choice (non-deterministic or decision-theoretic):

- Choose {A, B}
- Contingency



**RMPL for Group-Enroute**

```

Group-Enroute() [1,u] = {
  choose {
    do {
      Group-Traverse-Path(PATH1_1,PATH1_2,PATH1_3,RE_POS)[1*90%,u*90%];
    } maintaining PATH1_OK,
    do {
      Group-Traverse-Path(PATH2_1,PATH2_2,PATH2_3,RE_POS)[1*90%,u*90%];
    } maintaining PATH2_OK
  };
  {
    Group-Transmit(OPS,ARRIVED)[0,2],
    do {
      Group-Wait (HOLD1,HOLD2)[0,u*10%]
    } watching PROCEED
  }
}

```

**RMPL for Group-Enroute**

**Activities:**

```

Group-Enroute() [1,u] = {
  choose {
    do {
      Group-Traverse-Path(PATH1_1,PATH1_2,PATH1_3,RE_POS)[1*90%,u*90%];
    } maintaining PATH1_OK,
    do {
      Group-Traverse-Path(PATH2_1,PATH2_2,PATH2_3,RE_POS)[1*90%,u*90%];
    } maintaining PATH2_OK
  };
  {
    Group-Transmit(OPS,ARRIVED)[0,2],
    do {
      Group-Wait (HOLD1,HOLD2)[0,u*10%]
    } watching PROCEED
  }
}

```

**RMPL for Group-Enroute**

**Conditionality and Preemption:**

```

Group-Enroute() [1,u] = {
  choose {
    do {
      Group-Traverse-Path(PATH1_1,PATH1_2,PATH1_3,RE_POS)[1*90%,u*90%];
    } maintaining PATH1_OK,
    do {
      Group-Traverse-Path(PATH2_1,PATH2_2,PATH2_3,RE_POS)[1*90%,u*90%];
    } maintaining PATH2_OK
  };
  {
    Group-Transmit(OPS,ARRIVED)[0,2],
    do {
      Group-Wait (HOLD1,HOLD2)[0,u*10%]
    } watching PROCEED
  }
}

```

**RMPL for Group-Enroute**

**Sequentiality:  
Concurrency:**

```

Group-Enroute() [1,u] = {
  choose {
    do {
      Group-Traverse-Path(PATH1_1,PATH1_2,PATH1_3,RE_POS)[1*90%,u*90%];
    } maintaining PATH1_OK,
    do {
      Group-Traverse-Path(PATH2_1,PATH2_2,PATH2_3,RE_POS)[1*90%,u*90%];
    } maintaining PATH2_OK
  };
  {
    Group-Transmit(OPS,ARRIVED)[0,2],
    do {
      Group-Wait (HOLD1,HOLD2)[0,u*10%]
    } watching PROCEED
  }
}

```

**RMPL for Group-Enroute**

**Temporal Constraints:**

```

Group-Enroute() [1,u] = {
  choose {
    do {
      Group-Fly-Path(PATH1_1,PATH1_2,PATH1_3,RE_POS)[1*90%,u*90%];
    } maintaining PATH1_OK,
    do {
      Group-Fly-Path(PATH2_1,PATH2_2,PATH2_3,RE_POS)[1*90%,u*90%];
    } maintaining PATH2_OK
  };
  {
    Group-Transmit(OPS,ARRIVED)[0,2],
    do {
      Group-Wait (HOLD1,HOLD2)[0,u*10%]
    } watching PROCEED
  }
}

```

### RMPL for Group-Enroute

```

Group-Enroute([1,u] = {
  choose {
    do {
      Group-Traverse-Path(PATH1_1,PATH1_2,PATH1_3,RE_POS)[1*90%,u*90%];
    } maintaining PATH1_OK,
    do {
      Group-Traverse-Path(PATH2_1,PATH2_2,PATH2_3,RE_POS)[1*90%,u*90%];
    } maintaining PATH2_OK
  };
  {
    Group-Transmit(OPS,ARRIVED)[0,21,
  do {
    Group-Wait (HOLD1,HOLD2)[0,u*10%]
  } watching PROCEED
  }
}

```

Non-deterministic choice:

### Model-Predictive Dispatch for RMPL

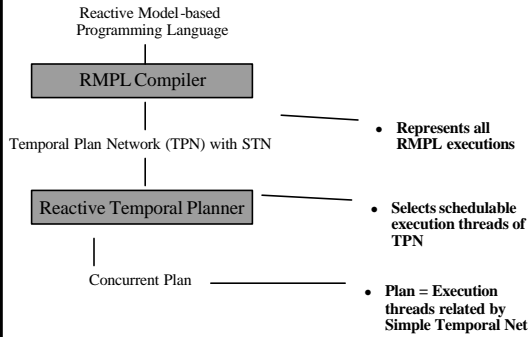
How do we provide fast, temporally flexible planning for contingent method selection?

- Graph-based planners support fast planning.
- ... but plans are totally order.
- Desire flexible plans based on simple temporal networks (e.g., Constrain-based Interval Planning).

How do we create temporally flexible plan graphs?

- Augment simple temporal networks with activities & choice.
- ⇒ temporal plan network TPN).

### Model-Predictive Dispatch for RMPL

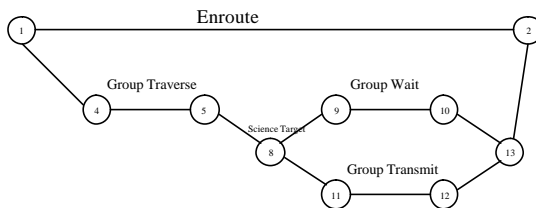


### Outline

- Safe Procedural Execution
- Model-Predictive Dispatch
  - Model-based Programming
  - Temporal Plan Networks (TPN)
  - Activity Planning (Kirk)
  - Unifying Activity and Path Planning
- Model-based Reactive Planning

### Enroute Activity:

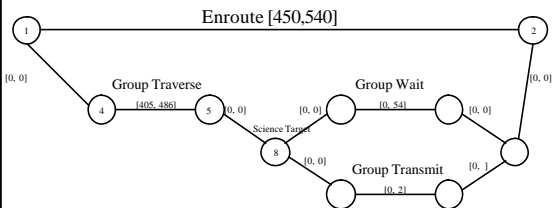
- Start with flexible plan representation



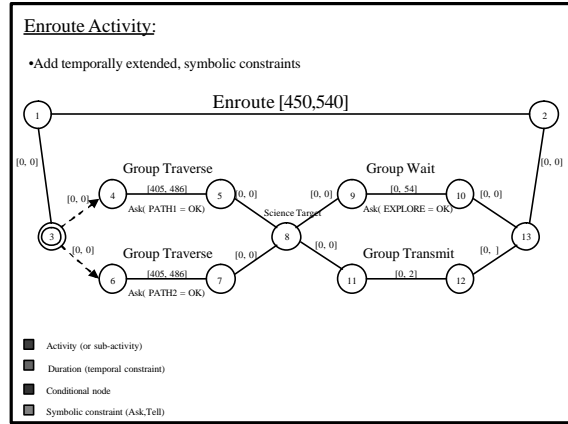
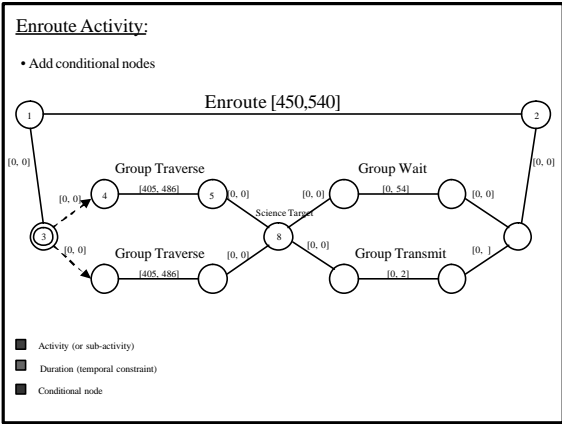
■ Activity (or sub-activity)

### Enroute Activity:

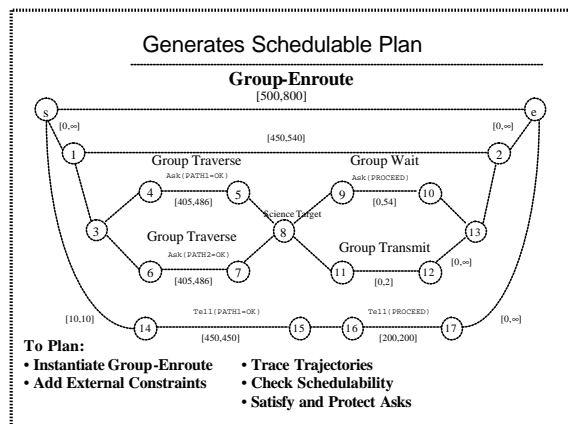
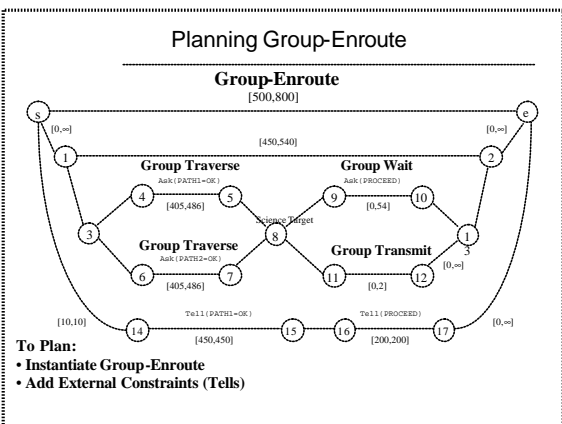
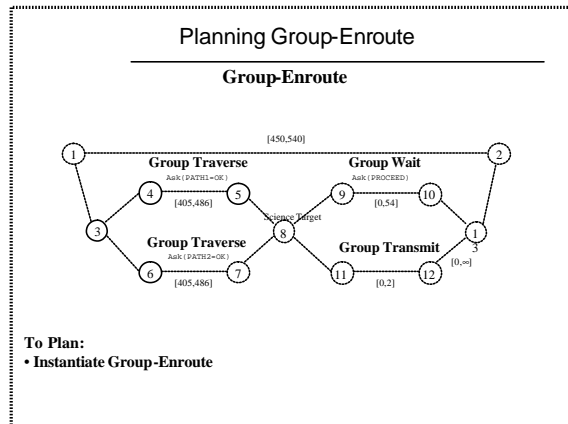
- Start with flexible plan representation

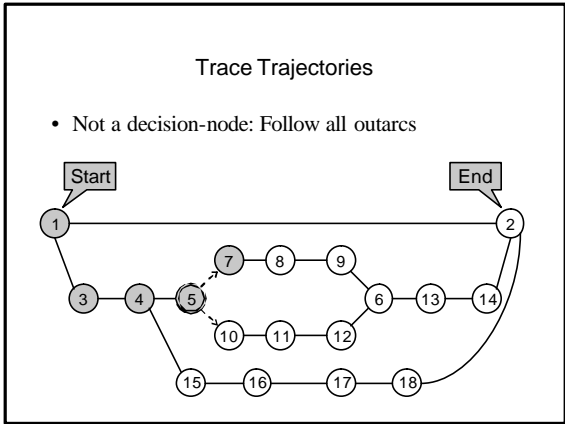
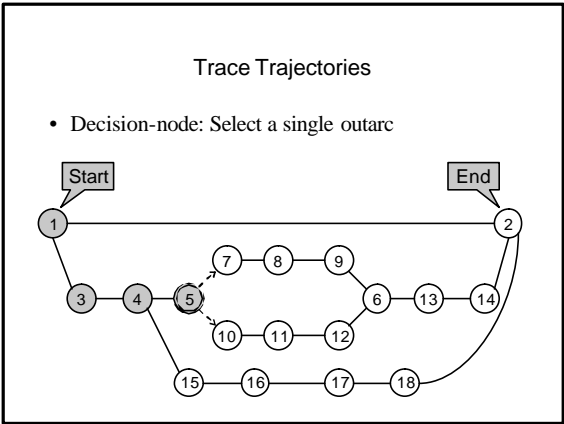
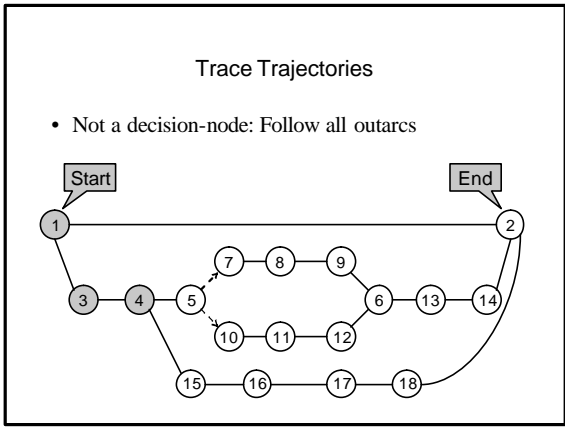
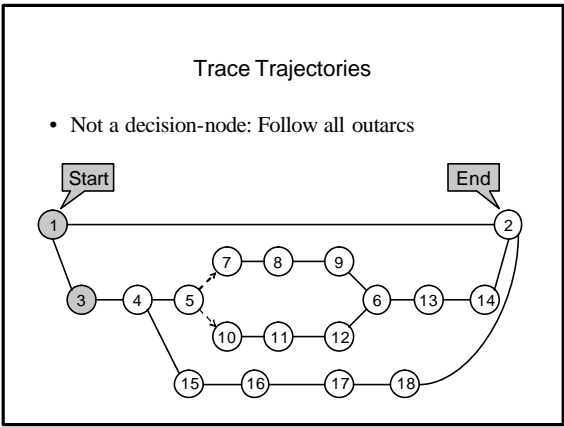
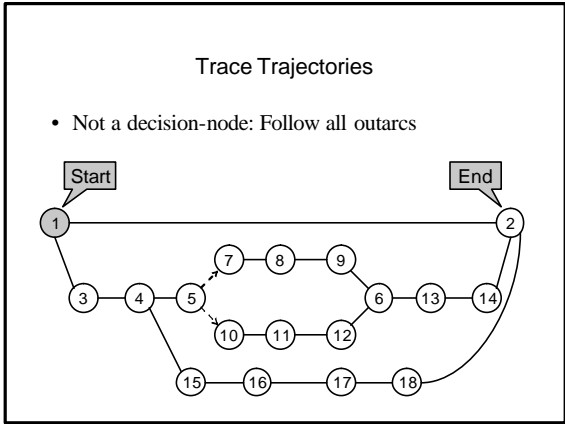
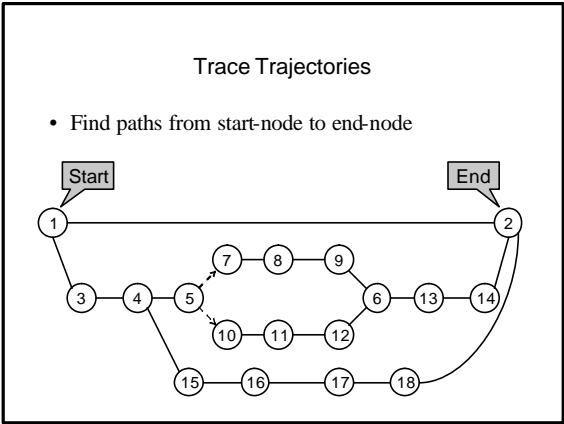


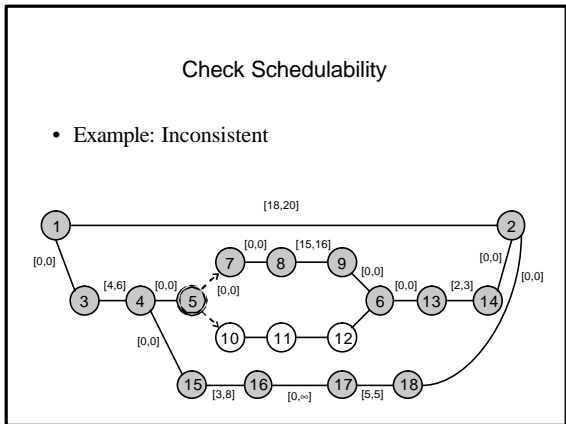
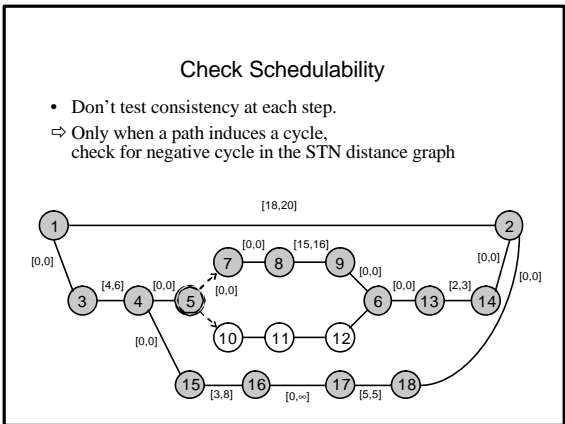
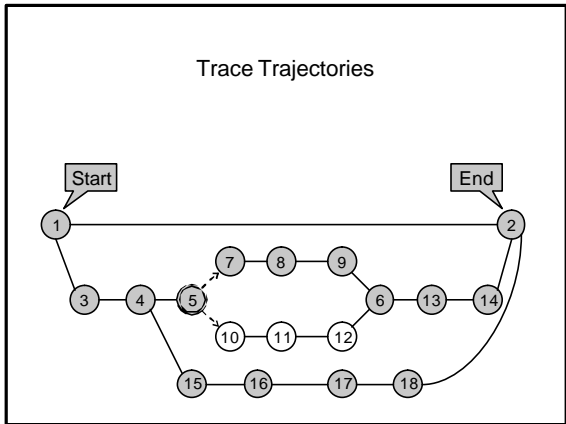
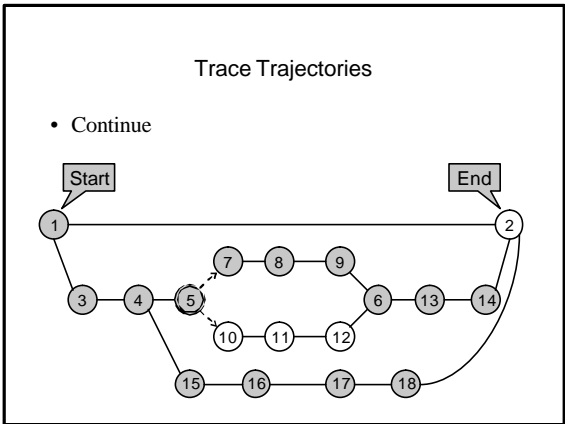
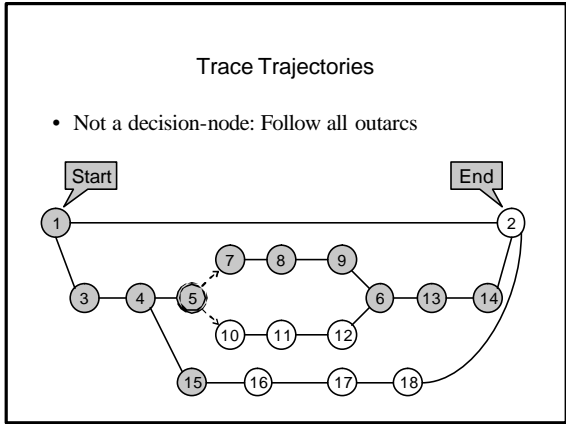
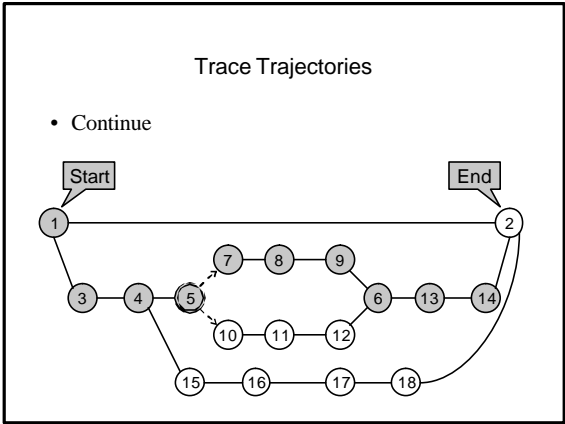
■ Activity (or sub-activity)  
■ Duration (temporal constraints)



- Outline**
- Safe Procedural Execution
  - Model-Predictive Dispatch
    - Model-based Programming
    - Temporal Plan Networks (TPN)
    - Activity Planning (Kirk)
    - Unifying Activity and Path Planning
  - Model-based Reactive Planning

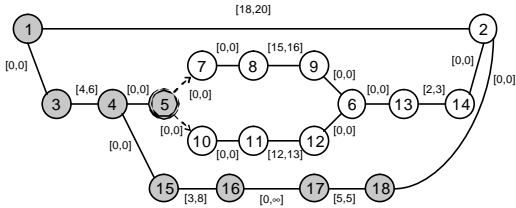






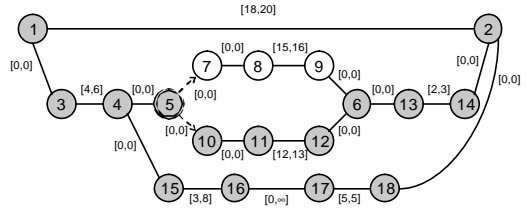
### Trace Alternative Trajectories

- Backtrack to choice



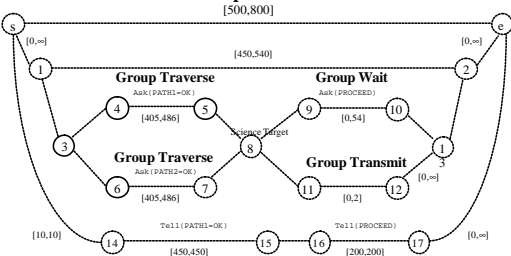
### Trace Alternative Trajectories

- Complete paths



### How Do We Handle Asks?

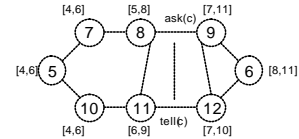
#### Group-Enroute



- Unconditional planning approach:
- Guarantee satisfaction of asks at compile time.
  - Treatment similar to causal-link planning

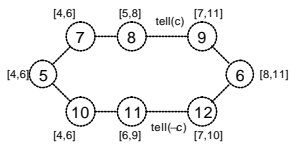
### Satisfying Asks

- Compute bounds on activities.
- Link ask to equivalent, overlapping tell.
- Constrain tell to contain ask.



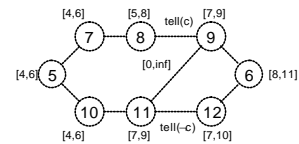
### Avoiding Threats

- Identify overlapping Inconsistent activities.



### Symbolic Constraint Consistency

- Promote or demote





## How do we optimally select activities and paths?

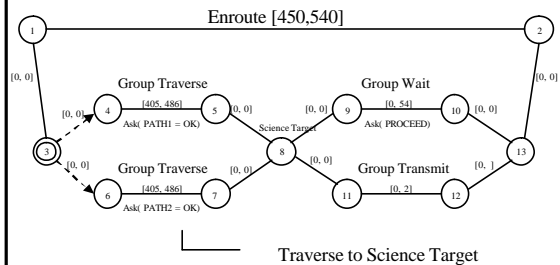
Background: Can perform global path planning using Rapidly-exploring Random Trees (RRTs) (la Valle).

Approach:

1. Search for globally optimal activity and path plan by
  - unifying TPN & RRT graphs, and
  - by searching hybrid graph best first.
2. Refine plan using receding horizon control.

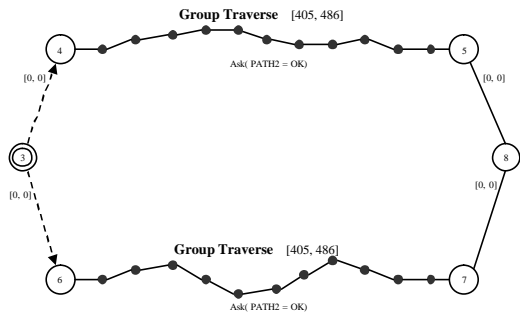
### Enroute Activity:

• Closer look at Group Traverse sub-activity



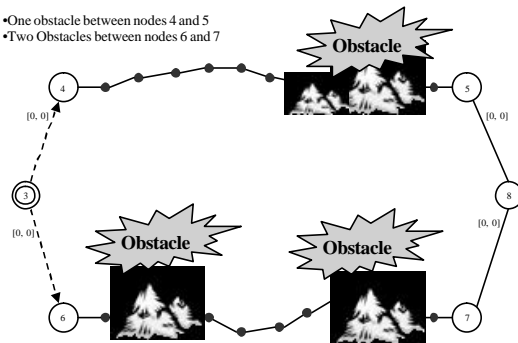
### Group Traverse sub-activity:

• Traverse through way points to science target



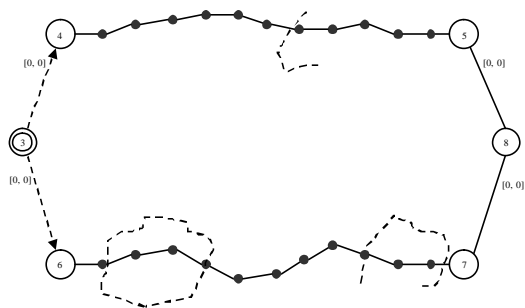
### Group Traverse sub-activity:

• One obstacle between nodes 4 and 5  
• Two Obstacles between nodes 6 and 7

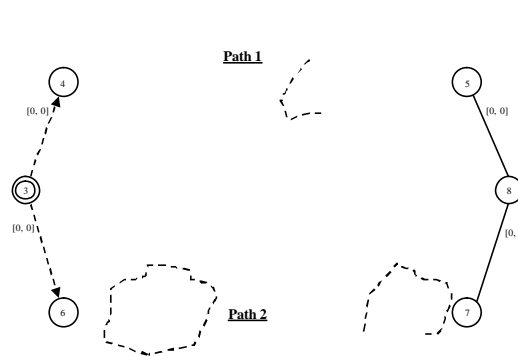


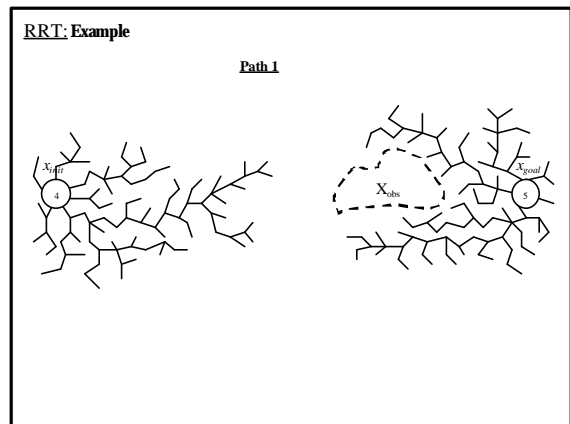
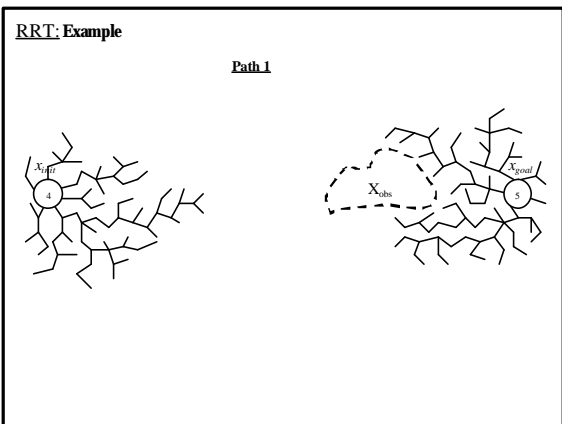
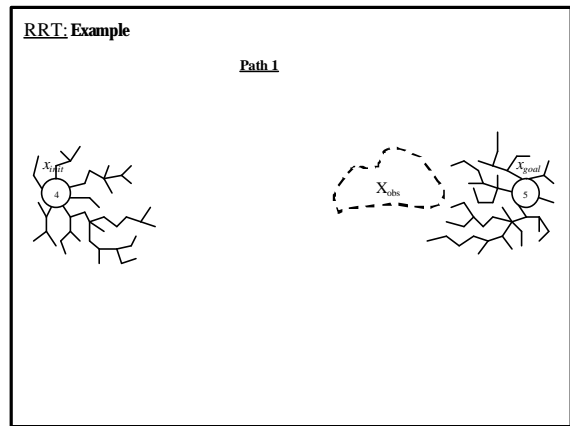
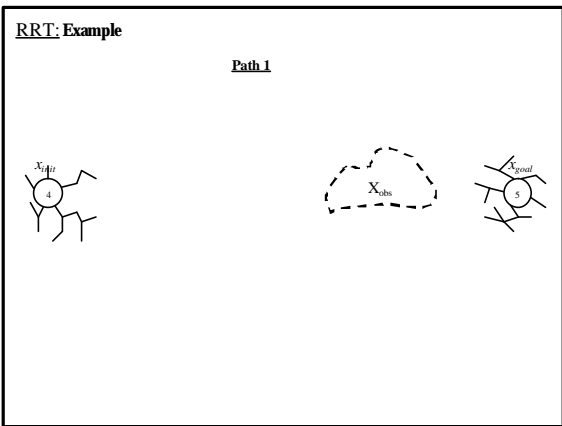
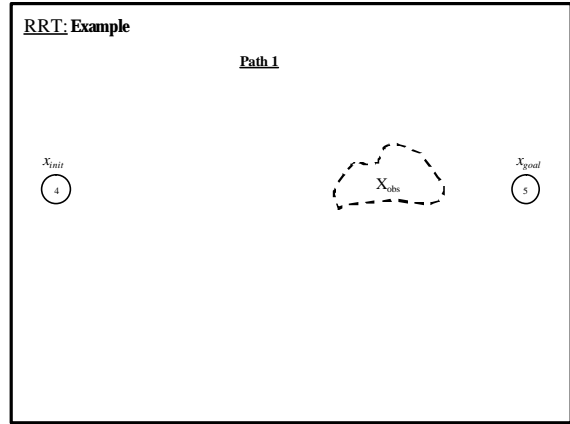
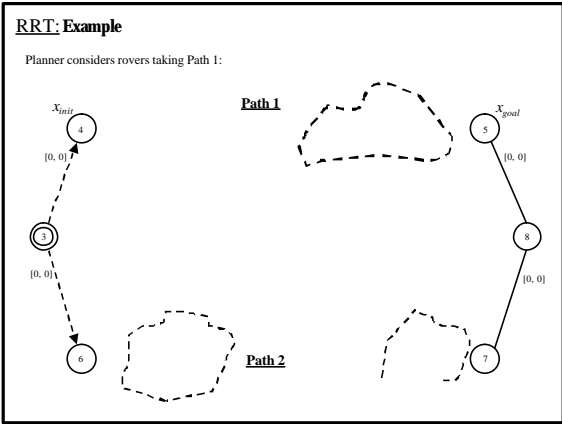
### Group Traverse sub-activity:

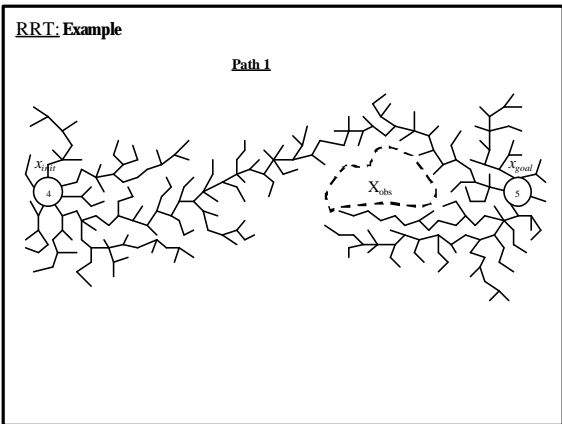
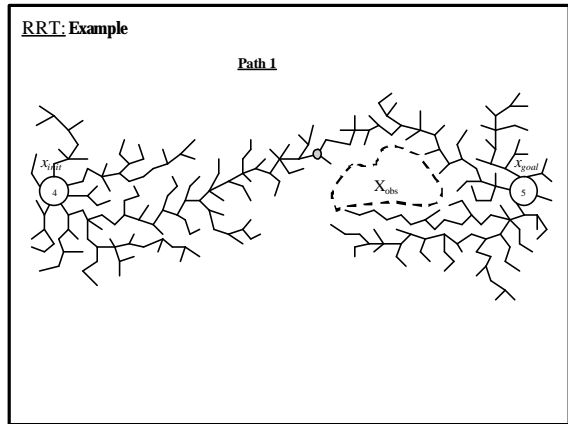
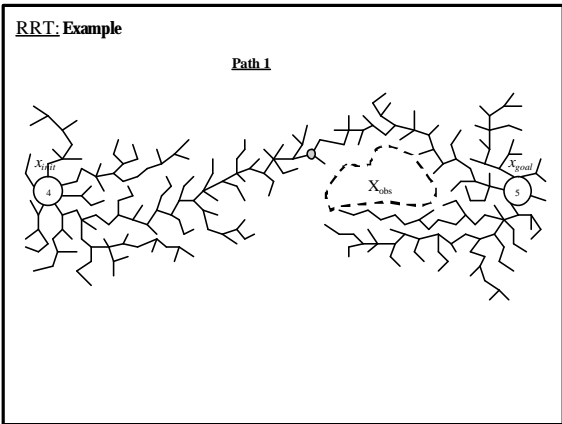
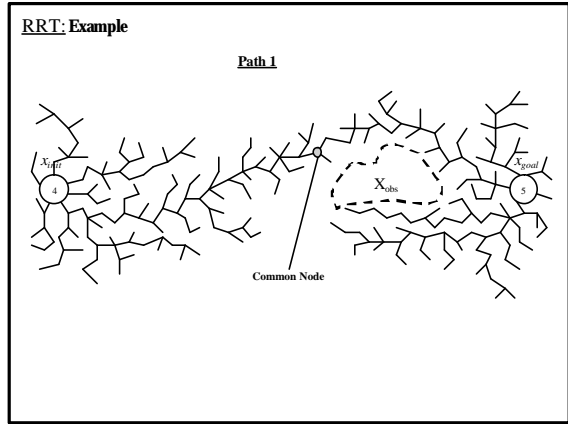
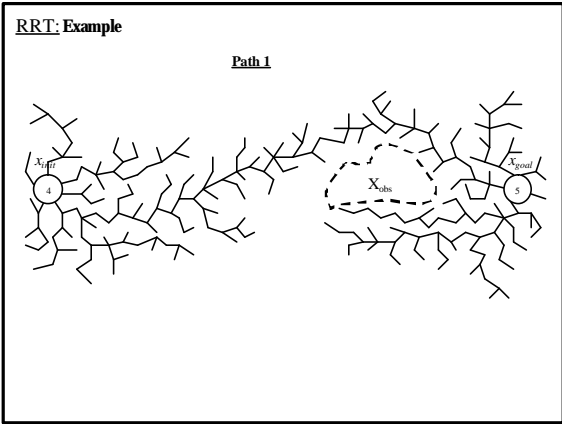
• Non-explicit representations of obstacles obtained from an incremental collision detection algorithm



### RRT: Example







**Model-Predictive Dispatch**

---

Goal: Fast, robust, temporal execution with contingencies, in an uncertain environments.

Solution: Model-predictive Dispatch, a middle ground between non-deterministic programming and temporal planning.

- Rich embedded language, RMPL, for describing complex concurrent team strategies extended to time and contingency.
- Kirk Interpreter "looks" for schedulable threads of execution before "leaping" to execution.
- Temporal Plan Network provides a flexible, temporal, graph-based planning paradigm built upon Simple Temporal Nets.
- Global optimality achieved by unifying activity planning and global kino-dynamic path planning.