Verifying a Distributed Motion Planner

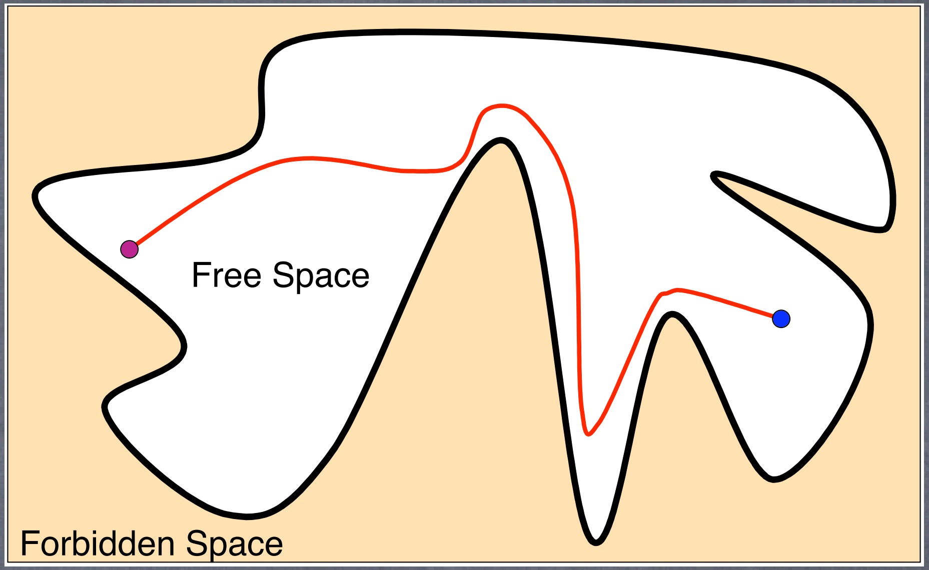
Presented by Jing Yang

Dec. 6, 2007

Agenda

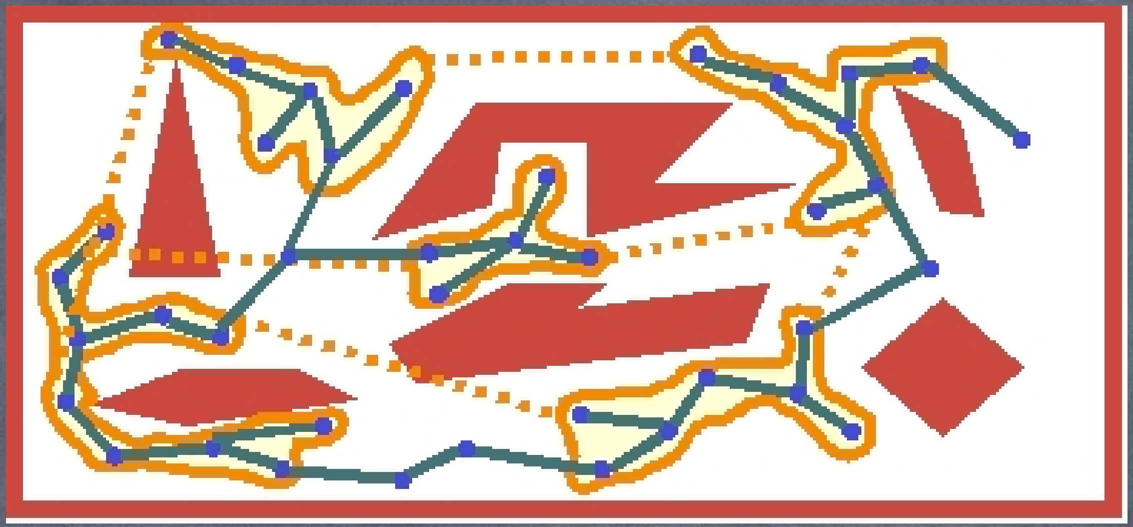
- Review the distributed motion planner (DSRT)
- Verification - experiencing JPF
    - State Search
    - Race Detection
    - Moving "synchronized" around
- Discussions

Motion Planning Basics



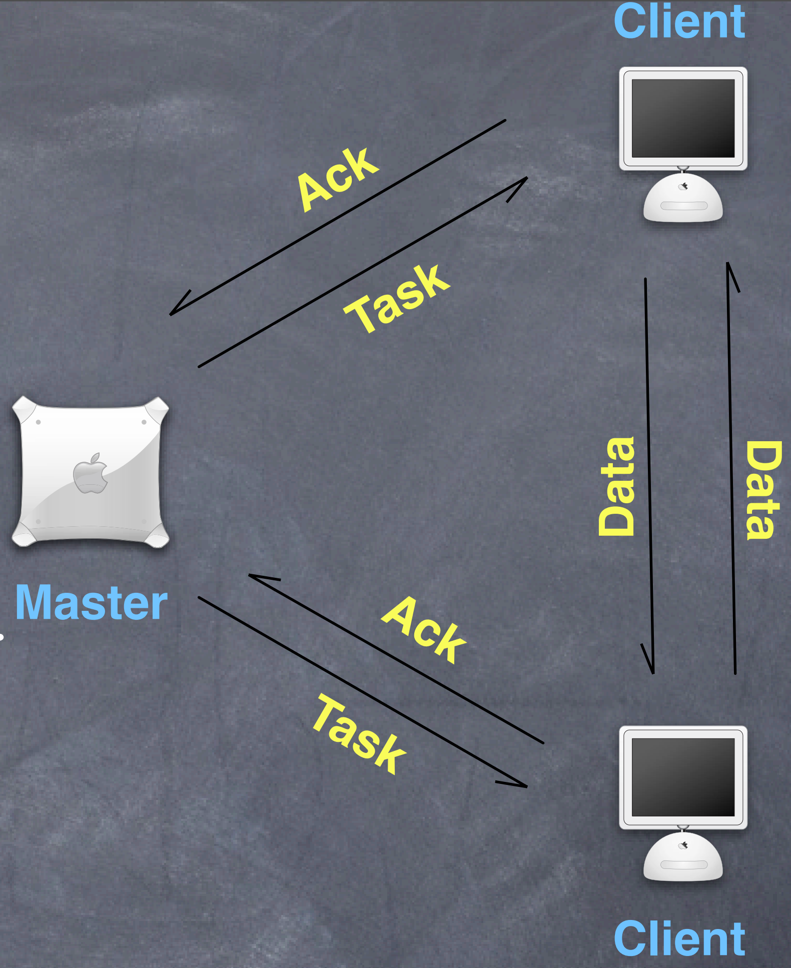
- Motion Planning -- Given two configurations of a robot, find a free path in the free Configuration space that connects them.

DSRT



- Milestone Computations
- Candidate Edge Computations
- Edge Computations - concurrency issues!

Implementing DSRT



- Communications
  - Task assignment: master -> client
  - Ask for task: client -> master
  - Data sharing: client -> client
- Asynchronous message passing
  - Messages are queued, sender doesn't block, receiver blocks when there is no queued message.
- Many-to-one channels

Experiencing JPF

- JPF reports error if there is a non-terminated thread.
- Let Master broadcast "Finish" messages to all Clients.

Experiencing JPF

- Old JPF reports “no live thread” and “deadlock”, if a thread object is created but not started.
- New version works!

Experiencing JPF - DFS/BFS

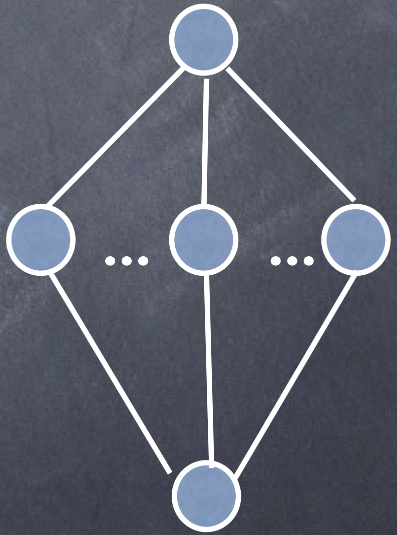
	2 threads	3 threads	4 threads
DFS	7 paths	11 paths	15 paths (4 hours on navy)
BFS	7 paths	10 paths	Out of memory

# paths doesn't increase when # Edges increases.

Experiencing JPF - paths

- Try counting paths of the DFS search
  - "A" at the end of main() after join();
  - "B" at the end of Master.run();
  - "C" the end of Client.run().

	# A	# B	# C
Print A only	7	-	-
Print A and B	7	136	-
Print A and C	7	-	39
Print A, B and C	7	141	44



Assertion

- Master: Each candidate edge is computed once
    - Put assert() at the end of the while loop
- Client: An candidate edge can be computed when its two ends are stored in the local memory
    - Put assert() inside the connect method

Experience JPF - Racing

- Add jpf.listener = gov.nasa.jpf.tools.precise.PreciseRaceDetector in jpf.properties
- 1 Master & 1 Client: no racing
    - Each channel has only one sender and one receiver
    - Sender and receiver do not change variables in the channel at the same time
- 1 Master & 2+ Clients: racing
    - Multiple threads attempt sending to a same channel

AsynchChannel: send & receive

```
public synchronized void send(Object m){
    if (m==null) throw new NullPointerException();
    messages.addElement(m);
    numMessages++;
    if (numMessages <= 0) notify(); }

public synchronized Object receive(){
    Object receivedMessage = null;
    numMessages--;
    if (numMessages < 0)
        try {wait();}
            catch (InterruptedException e) {}
    receivedMessage = messages.firstElement();
    messages.removeElementAt(0);
    return receivedMessage;}
```

Experiencing JPF - extra

```
public synchronized void send(Object m){
    if (m==null) throw new NullPointerException();
    messages.addElement(m);
    numMessages++;
    if (numMessages <= 0) notify(); //unblock the receiver}
```

- Try putting “synchronized” in different places

Experiencing JPF - extra

```
public void send(Object m){
    if (m==null) throw new NullPointerException();
    messages.addElement(m);
    numMessages++;
    synchronized(this){ if (numMessages <= 0)  notify();} }
```

- Java does not report error
- JPF has NoSuchElementException error when main thread is calling join() method
- JPF reports Deadlock after removing join in main. Both master and client are waiting.

Experiencing JPF - extra

```
public void send(Object m){
    if (m==null) throw new NullPointerException();
    messages.addElement(m);
    synchronized(this){ numMessages++;
    if (numMessages <= 0)   notify();} }
public synchronized Object receive(){
    Object received = null;
    synchronized(this){ numMessages--;
    if (numMessages < 0)
        try {wait();} catch (InterruptedException e) {} }
    received = messages.firstElement();
    messages.removeElementAt(0); return receivedMessage;}
```

- JPF reports no error for 2 clients

Summary

- Distributed motion planner: DSRT
- Experience JPF on DSRT implementation
  - Search state
  - Paths
  - Race condition

Questions?
Thank you!