

CS 249A
Assignment 2 Review Session

Aditya Ramesh

Overview

- Design and implement a shipping network
 - Create various entities
 - Query network for information
- Teams of 2
- Related to Assignment 3
- Create class library that can be called by client code
- Use attribute-oriented interface, notification model and entities/value types

Shipping Network

- Graph with locations as nodes and segments as edges
- Segment
 - Truck, boat, plane
 - Contains only source
 - Return segment contains destination
 - Directed
 - Return segment must be of same type

Location

- Customer
 - Connects to any segment
 - Shipment must either start or end here
- Port
 - Connects to any segment
- Terminal (Plane, truck, boat)
 - Connects to only particular type of segment

Overall Design

- 3 Layers
- Client
 - Sets up and queries network
 - Issues commands to rep layer
 - Eg. `fleet->attribute('Boat, speed');`
- Rep
 - String-valued service
 - Interface is fixed (Instance.h)
 - Issues commands to engine layer

Engine Layer

- Maintains actual network
- Cannot be directly accessed by client
- Can design own interface
- Place where required entities should exist
- Need to implement both rep and engine layer

Required Entities

- Location
- Segment
- Stats
 - Keeps track of state of network
 - Maintained via reactors
- Conn
 - Support queries on network
- Fleet
 - Information about transportation options

Queries

- Constraints include time, distance, cost and expedited shipment
- Connect
 - All paths between locations matching constraints
- Explore
 - All paths starting from location matching constraint
- Follow output format exactly as specified

Miscellaneous

- Output errors via stderr and ignore client request
- Stats, Conn, Fleet can have only one instance
- Milestone due on Oct 28, 11:59 PM via email
 - Pass/Fail marking
 - Show interface design (.h files enough)

Smart Pointers

- Recommended but not necessary

```
Class Location : public Fwk::PtrInterface<Location>
{
    typedef Fwk::Ptr<Location const> PtrConst;
    typedef Fwk::Ptr<Location> Ptr;
}
```

Value Types

- Replacement for primitive types
- Ordinal is for ordered types (e.g Distance)
- Nominal is for unordered types (e.g PortId)

```
class PortId : public Nominal<PortId, int> {
    public:
        PortId(int num) : Nominal<PortId,int>(num) {
            if (num < 0)
                //do error handling logic
        }
}
```