1. Determistic Greedy Scheduler (Enforce required tightness at node)

```
last_occupied_time = {}

func schedule(state) {
    sort_by_departure_time()
    for aircraft in state {
        # Assign target_ndoes per aircraft from its location to target location,
        # the time difference is calculated by a fixed speed.
        # *Separation at node is enforced.*
        target_nodes = get_new_target_nodes(aircraft.location, aircraft.destnation,
                                            aircraft.speed, last_occupied_time)
        mark(last_occupied_time, target_nodes)
        aircraft.assign(target_nodes)
    }
}
```

Problem: we can't just separate at node; otherwise, all aircrafts will move slowly on links.

Conflict

So, we define conflicts on two aircrafts.

```
func state.get_all_conflicts() {

    all_conflicts = []

    for a1 in all_aircrafts:
        for a2 in all_aircrafts:
            if get_distance(a1, a2) < SEPARATION_DISTANCE:
                all_conflicts.add(Conflict(a1, a2, time.now))

    retrun all_conflicts

}
```

2. Determistic Greedy Scheduler (Enforce required separation between aircrafts)

This is modified from Ritwik's implementation.

```
func schedule(state) {

    sort_by_departure_time()

    for aircraft in state {
        # Ignore any constraint now
        target_nodes = get_new_target_nodes(aircraft.location, aircraft.destnation,
                                            aircraft.speed, None)
        aircraft.assign(target_nodes)
    }

    # Resolve conflicts (in each time unit from now)
    for t in range(reschedule_time) {
        while(conflicts =
                simulator.predict(NO_UNCERTAINTY, t_later_from_now = t)
```

```
                        .get_all_conflicts() is not NONE) {
            for conflict in conflicts {
                a = get_less_priority_aircraft(conflict.a1, conflict.a2)
                a.target_nodes.add_delay_at(conflict.location)
            }
        }
    }

}
```

3. Greedy Scheduler under Uncertainty

We define the probability of an aircraft moving to the next node as expected is P_p. This is the value used by the scheduler only, not the real world simulator.

Approach: This is similar to "2. Determistic greedy Scheduler", but we ignore the conflicts that happens with probability lower than $P_threshold$.

Problem: We always have two aircrafts involve in one conflict, do this means: if $P_p \times P_p < P_threshold$, we don't solve any conflict at all. Otherwise, we solve all conflicts like we did in "2. Determistic Greedy Scheduler."