Collaborating intelligent lifts in large office buildings

Q1) Description of System [198 words]

The aim of the system is to use intelligent lifts for the office building to transporting people between the floors of the office building, while dealing with problems like traffic during peak times, energy conservation, and various emergencies.

Therefore, the user of the system would be all the passengers using the lift, and the management of the office building. The environment of the given system is the status of each lift, described by its location, ongoing direction, and the total weight of current passengers. Even though this environment should be accessible and deterministic, there are situations like fire and power outages that render the entire environment dynamic as information about its status becomes inaccessible.

Also, there would be two types of agents in the system - lift agents and the control agent. The lift agents will represent the lifts and have an internal state of the lift. The control agent receives input from each floor, and distributes it to the lift agents depending on the state and utilities they possess. For example, lifts with more free space or weight will be preferred over full ones, closer lifts and lighter lift may acquire more orders from the control agent. [198]

Q2) Architecture, Data Structure, Algorithms [496 words]

Before choosing architectures for the system, we need to review the reasoning and decision making process behind our system. Thus, all lifts in the system are expected to have following goals:

1. During emergencies and maintenance, they should disconnect from the system, abandon all future activity, stop and lock themselves for safety.
2. Carry the passengers inside the lift to their destinations.
3. Collect passengers from different floors as soon as possible.

These 3 goals are ordered by their priorities. Moreover, the delivery part has a higher priority than the collection part, is because it would be considered unacceptable for passengers to be stuck in the lift and sent to the opposite direction of destination level.

Considering these goals, it is clear that purely reactive agent architecture is not suitable, since this type of architecture does not take long-term information into account. A deliberative reasoning architectures appears

to be suitable for the lift system, considering that in almost every case there exists an available solution, and the accurate status of current environment is obtainable. The problem lies in the fact that reasoning takes more time, which could cause arise especially during emergencies.

This analysis of the necessary architecture proves that a Hybrid Architecture is the best choice for the lifts in the system. Separating the reactive part - detection of overload, jammed door, power outage - from the planning part of the system. Other inputs like lift button presses are passed to the planning layers, giving capability of logical planning for the future actions.

In order to plan for future actions, the system will need to have an internal state represented in logical terms. The lift agents will have basic information like current load, door status, and last/current stopped floor. The system also needs to deliver the passenger within the lift, and store the information received from the control agent, it will have a list of floors it need to stop at. To prevent sudden change of direction, current direction will be recorded. Moreover, the lift may have different running modes (during peak hours, some of the lift may change into odd/even floor only lifts).

A lift agent may have the following internal state, adapted from concurrent MetateM:

DoorOpened;Load(560kg);Going(Up);Moving(False);Mode(Normal);
LastLevel(20);Order(26,Goal);Order(23,Up);

Meanwhile, the control agents holds a list of tasks that are not distributed yet:

Order(1,Up),Order(15,Up),Order(22,Down)

The control agent will keep trying to distribute the tasks stored towards the appropriate lift agents. Using the above example, if the someone pressed the up button while on floor 24, the control agent would generate Order(24,Up), and ask the lift agents immediately to execute it. As the lift is going up, and at floor 20, the lift will add the order to the database, and send a confirmation back to the control agent.

Also, the lift agent might re-considerate its orders, if the load is almost full. In this case, the order may be sent back to the control agent, so that another lift may come and solve the traffic. [496]

Q3) Interactions Mechanisms, Protocols [493 words]

Other than the three priorities of the lift system discussed above, we still need the lift system to be power efficient. It would be a huge waste of power, if all of the list goes to the same floor, trying to pick up the same passenger or the situation in which the heavier lift at the bottom floor would be given to order, rather than a nearby lift, to travel across the entire building to pick up one passenger.

A Contract-Net Protocol may be used to solve the problem of replicate works. When the control agents recognizes a new button press from user, the order is registered in the notation from question 2. Then the control agent announce all the lifts agent in the system about the new order; the lift agent could reject it temporarily, or inform the control agent when it could do the task.

Since the control agent needs to assign only one lift agent to the task, the system will have to decide which lift is the most suitable for the job. Due to the possibility of having multiple pending tasks that needs to be allocated, we could use decision making methods like auctions and bargains.

For example, a lift could have its utilities calculated using its speed, weight, and available load. The emergency lift may have a larger capacity, but it is heavier and slower, so a free normal lift would be preferred to pick up passengers first. The lifts also have a limited carrying capacity, we could use that to refrain the lift from taking all the orders from the control agent, so each agent needs to plan for bidding on the tasks.

Although the tasks are allocated using auctions, there is one major difference: the price for each agent for the same task is different, as it is calculated based on the distance between the lift and passenger, and the wait time of that task. Therefore, a closer lift only have to pay a small “price” to get the task, and passengers that waited for a long time may become more desirable in the system. This makes the auction process similar to the Dutch auction. The control agent continuously lowers the “price” when the user waits, preventing far-away passengers to waiting too long for the lift. This method is more efficient in time than the English auction, because the control agent does not have to wait for the agents to come up with a higher bid.

After awarding a lift agent with the task, the agent is possible to fail, or give up on the task. The agent might give up the task after the load is considered “almost full”. In this scenario, the agent might not get another

passenger and it might return the task to the control agent with an increased wait time, allowing other agents to solve the traffic instead. The agent will fail all of its task if it goes to maintenance. [493]

Q4) Work Examples [594 words]

To ensure that the rightful interaction and decision making techniques were used that would satisfy the priorities listed in question 2, here is an example of how the system reacts in practice:

Consider the case of a medium-sized office building with underground floors -2, -1, ground floor, and floors 1-20. There are three normal lift and one emergency lift. It is 8:00 in the morning and the management of the building has set two of the normal lift to odd and even floor only mode. Here are the current internal states of the agents:

Normal Lift 1, moving upwards around floor 3:
Weight(1800kg);MaxLoad(1200kg);Load(1050kg);
Going(Up);Moving(True);Mode(Normal);LastLevel(3);
Order(4,Goal);Order(6,Goal);Order(8,Up);Order(11,Goal);
Normal Lift 2, switched to odd floor mode, stopped at floor 13:
DoorOpened;Weight(1800kg);MaxLoad(1200kg);Load(350kg);
Going(Up);Moving(False);Mode(Odd);LastLevel(9);
Order(15,Goal);Order(17,Goal);Order(19,Goal);
Normal Lift 3, switched to even floor mode, moving down to ground floor,
Weight(1800kg);MaxLoad(1200kg);Load(0kg);
Going(Down);Moving(True);Mode(Even);LastLevel(1);
Order(-2,Control);
Emergency Lift, currently idle on floor 10:
DoorOpened;Weight(2500kg);MaxLoad(1800kg);Load(0kg);
Going(Stop);Moving(False);Mode(Emergency);LastLevel(10);
Control Agent:
Order(0,Up(12s));Order(-1,Up(28s);Order(-2,Up(36s));
Order(19,Down(82s));

For each lift to be able to take the passenger inside to the required destination, it need to start decelerating before it arrives at that particular floor. Take lift 1 from the example, the lift agent may have the following reasoning:

Moving(True)^Going(Up)^LastLevel(x)^Order(x+1,_)=>Do(Decelerate);

The agent may also use the internal state for other operations:

Moving(False)^LastLevel(x)^Order(x,_)=>Do(OpenDoor)
MaxLoad(x)^Load(y)^(x>y)^DoorOpened=>Do(CloseDoor)

When an lift agent has no more orders on one direction, it could change the Going state to Stop, which allows the control agent to move it towards a place to solve a type of orders. In this example, the Emergency Lift is idle, so it would be moved to either -2 or 19 floor, like lift 3, which is being moved to -2 floor to solve the Up traffic. To solve the pending Up/Down order respectively. We may calculate the utility by just adding up the wait time:

Up: 12s+28s+36s=76s<82s: Down

The control agent may just give the emergency lift an order to move to floor 19 for the Down order. This solves the problem of letting passengers wait too long for a lift, at the cost of energy, and making average waiting time higher.

When an order that can be done by multiple lift agents is added, the control agent will announce the order to available agents. Using the above example, when a user presses up at floor 11, and we assume that the emergency lift is also going up. Then, the emergency lift and lift 1 will be available for the job, both agents send their utility to attend the bidding of the task. The utility may be calculated using this formula:

100*(Max Load–Load)/Weight–10*Pending Orders

Lift 1 would have utility of -1, while the emergency lift has a very high utility of 72. However the emergency lift might be a good choice as lift 1, since it is almost full, and relatively far away from the target floor. The control agent may also use some inverse function to calculate the “price” of the task, like this formula:

10*Floor Distance/(Order Waiting Time+1) for each lift,

This makes the power consuming choices delayed to save energy. Also, with a negative utility, lift 1 might not be able to acquire any more tasks, considering it has an almost full load. It would just waste time and power to do an impossible task as it might just send orders back to the control agent when the utility is too low.

When a lift enters emergency mode, it sends all orders back to the control agent, sets both moving and going to stop, keeps the elevator door at current state, and locks itself in current shaft position for safety purposes. [594]

Q5) Evaluation [196 words]

The whole system is heavily based on reasoning, which might takes a lot of time and makes the lift move through the floor is should stop at. Also the internal state might be too complicated, some of the actions can be handled by the lifts' firmware. Giving a clearer reasoning process, instead of wasting time on irrelevant actions like open doors while moving.

In question 4, most of the utility functions are not tested, this may lead to undesired side effects. Like choosing a less energy conservative lift just because it is closer, or making people on the ground floor wait too long for the lifts, in the worst case, completely ignore some of the orders as lifts are always full around those floors. These factor may make the system to be considered broken.

As a whole, the system tends to be too centralized, since all of the task distribution depends on the control agent. The lifts agent only collaborate when the control agent interferes by re-allocating tasks, which takes quite a long time due to the bidding nature of the Contract-Net Protocol. A different strategy like coalition formation between lifts may be a better choice. [196]