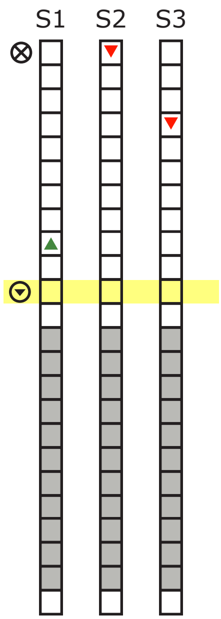
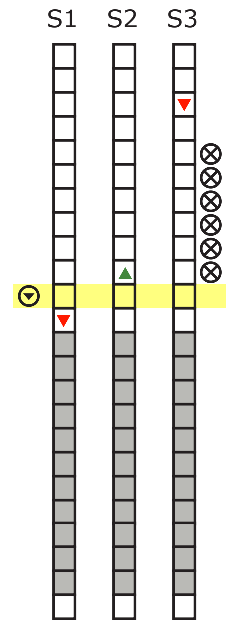
Possible Bank Select Strategies

When a Bank Level requests service up or down, a single Shaft within the Bank must be selected to handle the call. A simple strategy would be to randomly select one of the available (in-service) Shafts. This strategy would frequently make passengers wait too long for a Cabin to arrive.



Another strategy might be to choose the closest Cabin to the calling Bank Level. As we can see from the scenario at left, this is not always a good plan. S1 is the closest Cabin, but the Cabin is going the wrong way with a stop pending at the top floor! S3 on the other hand is pointed in the correct direction, so it has the shortest distance to travel. Clearly, we need to take direction into consideration.

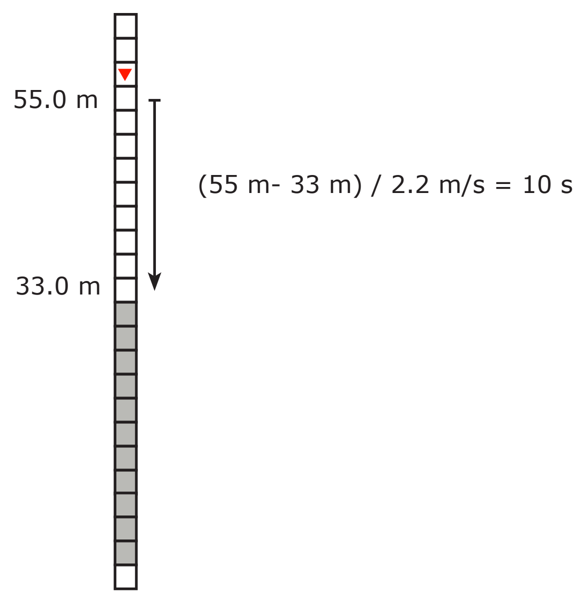


So we could select a Cabin that is both approaching the calling floor and moving in the same direction as the call. In the scenario to the left that would be S3. But S3 is not the best choice because of all the stops. S1, on the other hand, is likely to hit bottom and come around before S3 is finished. This example demonstrates that we should assess each Cabin using a handful of weighted factors and calculate the best score.

Average time without stopping

Clearly distance from the calling floor is an important factor. A moving cabin travels at variable speed. For our consideration, the average travel speed should be sufficient. We could say that a Cabin travels anywhere from 3 to 16 kph, spending most of its time toward the lower speed. We could guess that reasonable average transit speed is 8 kph or about 2.2 m/s.

So, for any given stretch, we just measure the distance between Shaft Levels and divide by the average transit speed to estimate the transit duration.



Stop duration

The time consumed at each is also a key factor. By adding the door open time, door close time, open/close transitions, and allowing for an average of one door re-close due to obstruction, we could say that a typical stop represents a 10 second delay. In reality the time will vary, but 10 seconds might be a good average.

Hmm, this brings up a problem. At the time we select a Cabin do we consider only the currently registered stops on the way to the destination or do we consider all potential stops? The longer the distance, the greater the probability that some kids will step into the Cabin and push a handful of buttons! Whenever you step into a Cabin to go a long way in a hurry, you hope for the best case scenario and fear the worst. It's probably wise to, again, just take the average for our calculations.

Route

Time lost due to distance, pending and potential stops can't be computed until we know the exact route from the Cabin to the calling floor. That may seem strange since the Cabin lives in a one dimensional world. But remember that the travel algorithm requires that the Cabin move as much as it can in one direction before reversing. So we must consider the number of reversals and the distance to each before we can measure distance and count stops.

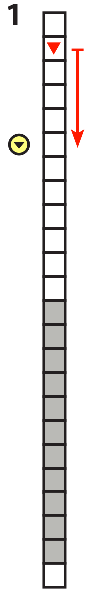
Route from Cabin to Calling Floor

We do not know what future stop and call requests will be placed before a Cabin arrives at the calling floor, servicing in the correct direction. But with some probability averaging, we can reasonably estimate the likely delay before a Cabin arrives at the calling floor.

Two factors influence the travel path of the Cabin. 1) Do the Cabin and floor call directions match? 2) Is the Cabin behind or ahead of the calling direction? We construct a table to show all possibilities and then illustrate a scenario for each.

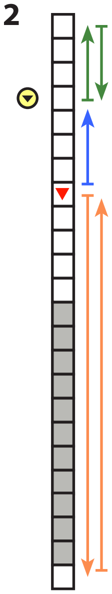
Service direction	Relative position: behind	Relative position: ahead
match	1	2
opposite	3	4

1



Here, the Cabin's travel direction matches the call direction and the Cabin is behind the calling floor. In this case there is only the approach to consider. If there aren't too many stops in between, this is probably an ideal choice.

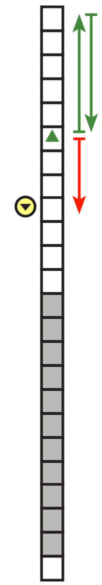
2



The Cabin's travel direction matches the calling direction, but it has already moved beyond the calling floor. The first leg takes us some distance away from the calling floor since the Cabin starts with a downward travel direction. The return leg has three components.

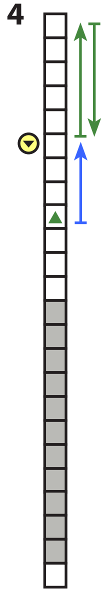
First there is the retrace back to the Cabin's original location. Then there is a path back and potentially overshooting the calling floor. Finally there is the return path which will intercept our called floor. In most cases this will be an undesirable choice.

3



The Cabin is behind the calling floor with an opposite travel direction. So the Cabin need only finish its current service direction and then it's a straight shot to the calling floor.

4



The cabin is past the calling floor with a travel direction opposite the calling direction. This means that there may be some overshoot and return.

Bank Select Algorithm

Leon Starr

Project: Elevator 2.0
Domain: Application
Subsystem: Same
Doc_number: tn.6 Version: 1.0

The algorithm

We should have enough factors now to make a reasonable stab at a credible bank selection algorithm. In short, we will estimate a travel delay for each Cabin in our Bank. Then we will choose the Cabin with the shortest delay. The trick, of course, is computing this delay.

We will need the following Bank attributes:

Average_cabin_speed
Average_stop_duration

We should have enough factors now to make a reasonable stab at a credible bank selection algorithm. In short, we will estimate a travel delay for each Cabin in our Bank. Then we will choose the Cabin with the shortest delay. The trick, of course, is computing this delay.

Determine the Cabin's position with respect to the call. Is it ahead (pointed to by the call arrow) or behind (facing the tail of the call arrow)?

Determine whether or not the Cabin direction matches the calling direction.

Based on these two determinations, choose route 1-4 as shown on the previous page.

Count stops and measure distance for each leg of the route. Add the leg distances together and the total stops.

transit_delay = distance / my_Bank.Average_cabin_speed
stop_delay = stops * my_Bank.Average_stop_duration
total_delay = transit_delay + stop_delay

Now this algorithm accounts for potential stops only indirectly. A longer distance will contribute a penalty. You could argue, though, that the portion of a longer distance through serviceable bank levels with no requests pending should contribute a penalty for potential stops. You could, for example, assume one potential stop for each five non-requested serviceable floors passed, but there's probably no need to get so complicated. Still - nice exercise for the student...