

14

Advanced Elevator Behaviors

Distributed Embedded Systems

Philip Koopman

October 12, 2011

**Carnegie
Mellon**

Overview

- ◆ **Where we've been:**
 - End-to-end scheduling
- ◆ **Today:**
 - Revisit time-triggered design
 - Advanced elevator topics
- ◆ **Where we're going:**
 - Verification, validation, and certification

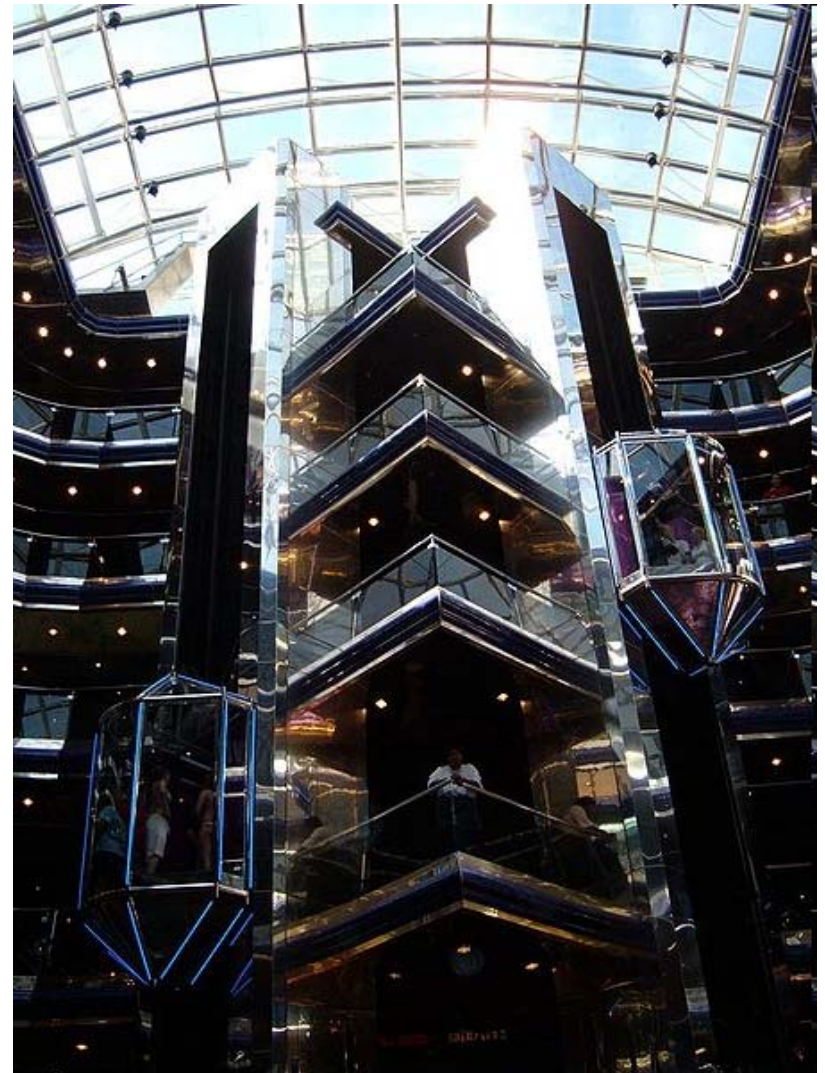


Photo Credits: http://commons.wikimedia.org/wiki/File:240_Sparks_Elevators.jpg

http://commons.wikimedia.org/wiki/File:Carnival_Sensation_Grand_Atrium_elevators.JPG 2

Time-Triggered vs. Event-Triggered

◆ Event-triggered

- Actions occur in response to an event in the system
- ET is good for systems that are naturally transactional
 - Financial systems
 - Systems backed by databases
- Record events in order to maintain state

◆ Time-triggered

- Actions occur based on the progression of time
- Decisions based on state variables
- TT is good for systems that are "state oriented"
 - Systems with physical state that needs to be represented in the system
- Infer events from changes in state variables
 - Have to sample state variables fast enough
 - Might miss events that occur faster than the time-triggered period

Entry Actions on States

- ◆ **This is not a time-triggered behavior**
- ◆ **Best practice:**
 - Eliminate entry actions on all states
 - This may require re-thinking your design...
... but when you are done your design is almost always a better more robust design
- ◆ **Do NOT add a “this is the first time” flag variable to work around this!**
- ◆ **Alternative (not the best practice):**
 - Add an additional state between the two states
 - This adds complexity
 - This adds latency (pay attention to safety constraints)

How do I know if my design is time-triggered?

◆ Pure Time-triggered designs DO:

- Perform actions (computations) and write outputs every period
- Make decisions based only on the current state of the system

◆ Pure time-triggered designs DO NOT:

- Have actions on arcs
- Make decisions based on how the current state was reached
- Have entry actions (* annotation)

◆ Your project **MUST** be time-triggered!

Advanced Elevator Topics

- ◆ **Commit point**
- ◆ **Dispatcher algorithms and the desired floor message**
- ◆ **Race conditions for distributed systems**
- ◆ **Startup conditions**
- ◆ **Passenger interactions**
- ◆ **Real-world elevator design complexities**

Commit Point Calculation

◆ Stopping distance:

The distance between where you are now and where you can stop

- Assumes following normal (non-emergency) acceleration profile
- At high speeds, this might be multiple floors away

◆ Commit Point:

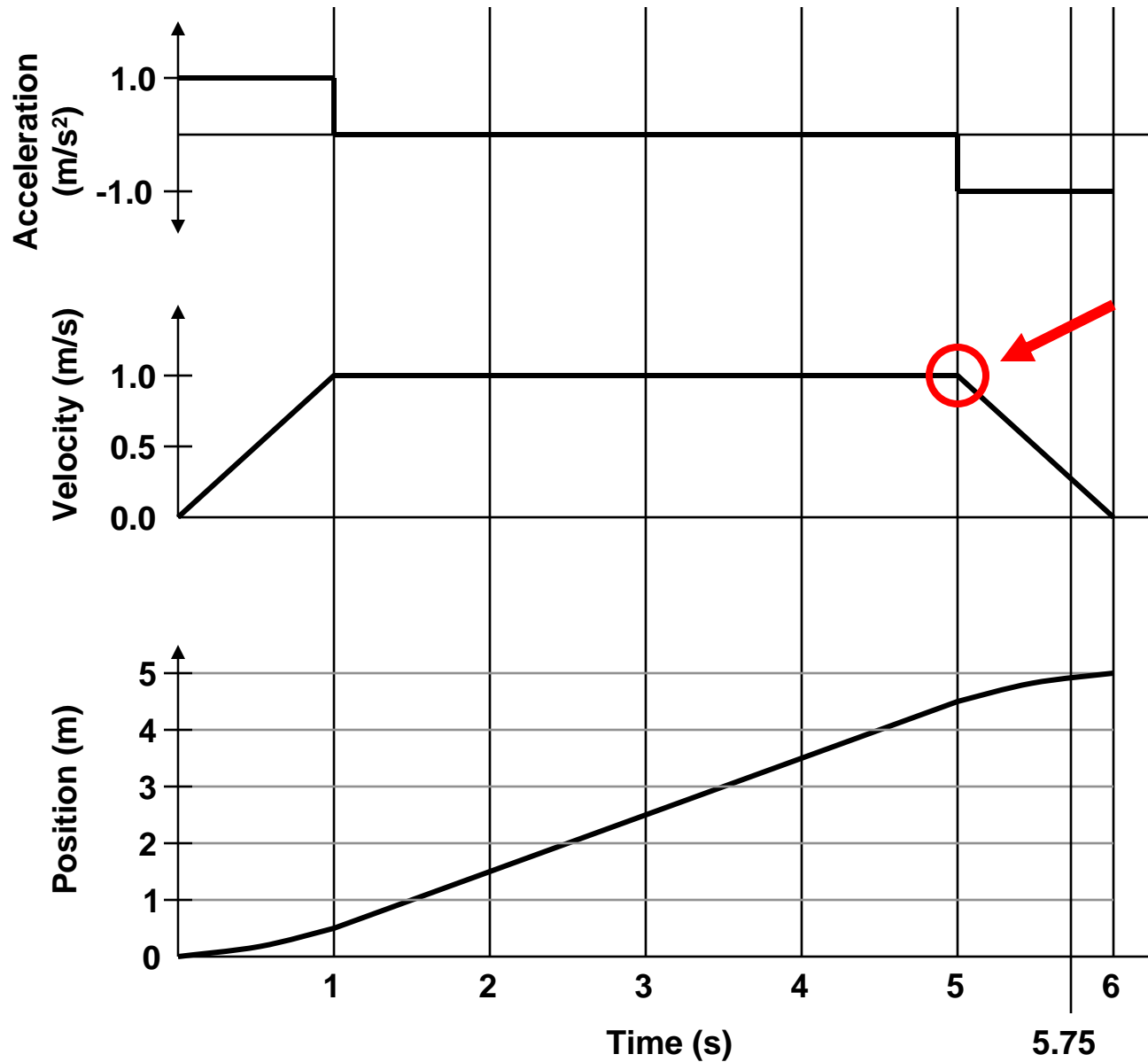
The elevator position at which you must decide whether to stop at particular floor

- Occurs when elevator reaches the stopping distance from that floor location
- This is a function of elevator speed!

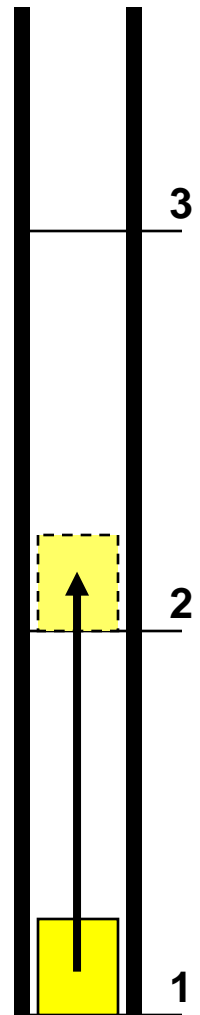
◆ When you pass the commit point, you lose the option of stopping

- Even if you still haven't reached the floor yet
- Think of the “point of no return” for that floor

Elevator Motion Profiles



**Commit
Point
Reached**



Using The Commit Point

- ◆ **The commit point gets further away as speed increases**
 - When stopped, the commit point is exactly at the floor (stopping distance = 0)
 - At high speeds, commit point is several floors away from elevator position
- ◆ **The course project elevator can stop “instantly” from slow speed**
 - But, from fast speed you need to ramp down
- ◆ **Computation must be conservative, take into account**
 - Granularity of sensor inputs
 - Worst case network delays
- ◆ **Controllers need to have a consistent notion of commit point**
 - Not too difficult in the fault-free case, or with dropped message faults
- ◆ **In real elevators, how does commit point affect car floor indicator?**
 - Why?

Dispatcher Messages – What do they mean?

◆ DesiredFloor

- Floor – the floor we intend to go to next
- Direction – the direction we intend to go **after** we reach the desired Floor
- Hallway – which doors should open

◆ DesiredDwell

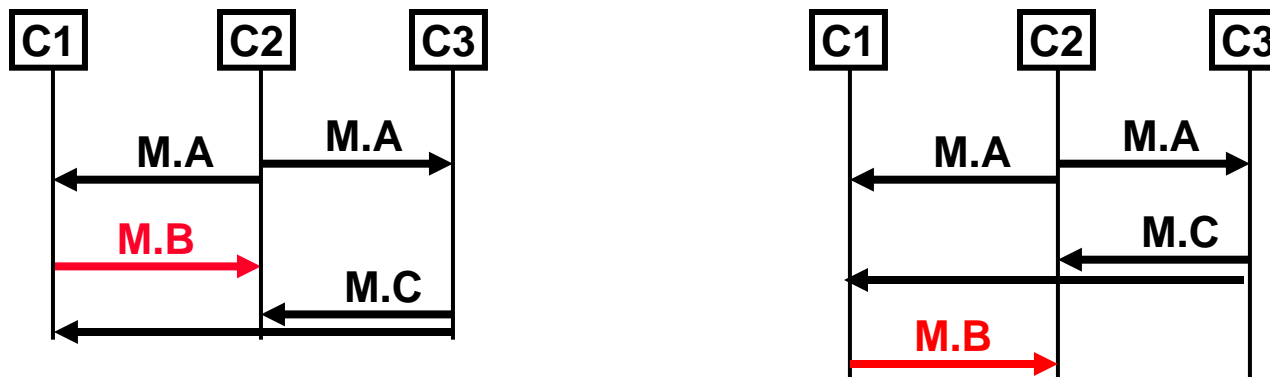
- How long to hold the doors open
- Might vary depending on whether your answering a hall call or a car call
- Might vary depending on mode (up peak, down peak)

◆ Starting in Project 8: Follow the above usage. For example:

- If the elevator is stopped and opening its doors
AND there is no pending call at the current floor
AND there is a pending call at another floor
THEN:
 - DesiredFloor.Floor must NOT BE current floor by the time the doors are fully open
 - DesiredFloor.Direction must correspond to illuminated lantern direction

Race Conditions

- ◆ **Definition:** the correct operation depends on the timing or arrival order of several events
- ◆ **Distributed nodes run asynchronously**
 - Race conditions can occur due to jitter or offset in periodic execution



- ◆ **Which part of your design is the correct place to resolve a race condition?**

Elevator Race Condition Scenario

◆ Scenario:

1. Doors close completely
2. A passenger arrives and makes a hall call just as the doors close, just as the car is about to start moving
3. **What should the elevator do here???**

◆ Two options:

- Open the door OR
- Move on to the next floor and get the passenger later

◆ Either choice is acceptable, as long as design is consistent

◆ Make sure that your drive control and door control agree

- You may need to add interlock states to one or both controllers.
- **What channels do you have to facilitate this communication?**

◆ Other race conditions are probably a result of design faults.

Elevator Startup Conditions

- ◆ **Course simulator initializes the system to a known state**
 - So you don't have to worry about this in the course project
- ◆ **Real elevators have physical state that persists**
- ◆ **Do not assume that the elevator was shut down / parked properly**
 - Immediately set outputs to something safe
 - Collect information about the system
 - Make the software state match the physical state, not vice-versa
- ◆ **Reading the current state might not be trivial**
- ◆ **Example: car position**
 - What if you are between level sensors in the hoistway?
 - What if the level position sensor is inertial?

Passenger Interaction

- ◆ **More “fun” ideas from the Internet (don’t do these in real life):**
 - Sing, "Mary Had a Little Lamb," while continually pushing buttons.
 - On a long ride, crash from side to side as if you're on rough seas.
 - Leave a box between the doors.
 - Lean against the button panel.
 - Say, "I wonder what all these do," and push the red buttons.
- ◆ **The point is, people do unexpected things**
- ◆ **Even when acting “normal,” can’t always determine passenger intent**
 - Infer some things from calls, weight sensors, door sensors
 - **How many passengers are represented by two button presses?**
- ◆ **Design your system to respond to sensor inputs**
 - Do something reasonable in corner and extreme cases
 - Avoid designs that let the passengers “hack” the elevator

Passenger Interaction

- ◆ **After safety and functional requirements, comfort is the key!**
- ◆ **What makes an elevator comfortable?**
 - Predictable behavior
 - Responsiveness
 - Smooth acceleration/deceleration
- ◆ **People may be unwilling to ride if the elevator acts in unexpected ways**
 - Elevator lanterns indicate one way but car goes the other way
 - Call lights don't respond interactively
 - Call lights go on or off for no apparent reason
 - Car takes a long time to level at a floor
- ◆ **If passenger does the “wrong” thing, ok to make them uncomfortable**
 - Excessive door reversals → ‘nudging’
 - Overweight buzzer
 - Don't do anything that violates safety criteria

Real-world Elevator Complexity

- ◆ **Course elevator design just scratches the surface**
- ◆ **Real elevators are a lot more complicated**
 - Additional inputs
 - Installation difference and building codes
 - Economics of installation
 - Modal behavior
- ◆ **Think about how complexity might affect your design or design process**
- ◆ **Don't worry about implementing complex behaviors**
 - Except maybe simple modes in the dispatcher

Additional User Interfaces

◆ In the car:

- Emergency stop switch
- Door open / Door close
- Door hold open (freight)
- Call cancel
- Card readers
- Key controls
- Passing and direction tones (accessibility)
- Inputs for emergency modes (vary by mode and local building code req's)
- RFID reader

◆ Outside the car:

- External control panels
- Special calls (e.g. penthouse)
- Maintenance controls in the hoistway or on top of the car

Installation Differences

- ◆ **Different buildings require different elevators**
 - Number of floors
 - Floor locations (might not be uniform)
 - Car size / weight / capacity
 - Acceleration profiles
- ◆ **Building codes may impose different safety and installation requirements**
- ◆ **Can parameterize some of these differences, but there are tradeoffs**
- ◆ **Two ends of the spectrum**
 - Take every possible situation into account
 - Complex designs are hard to get right
 - Can amortize design costs over many installations
 - Design a new controller for each building
 - Simple designs are easier
 - May be cost prohibitive
- ◆ **Some middle ground is probably the right answer**

Economics of Elevator Installations

- ◆ **The course elevator is completely distributed (for teaching purposes)**
- ◆ **How distributed should a real-world elevator be?**
 - The answer is driven by economics and system constraints
- ◆ **Example: hall call controllers**
 - A single MCU to control both buttons might be cheaper
 - but not if the buttons are cheap because they are modular
 - In a 2 story, 1 hallway installation, a wiring harness from the first to second floor might be cheaper than a second microcontroller
 - In a 100 story, multi-hallway installation, you certainly don't want to run a wire from the control cabinet for each button.
 - BUT what about the network load of 100+ hallway controllers sending their own mHallCall and mHallLight messages? (remember longer CAN networks are slower)
- ◆ **Optimization problem: find the lowest cost solution**
 - Remember that complexity also costs money ...
 - ... but SW cost might be difficult to measure compared to hardware costs

Elevator Modes

- ◆ **Difficult to define; One possible definition:**
 - “a distinct set of behaviors exhibited by a system”
- ◆ **It gets more complicated with distributed systems**
 - Some modes only affect some parts of the system
 - How many controllers have to change their behavior before you call it a mode?
- ◆ **Example: “up peak” and “down peak”**
 - Dispatcher behavior is changed
 - Buttons, doors, etc. remain the same
 - Does the rest of the system need to know if the dispatcher is doing up-peak?
- ◆ **Something that appears to be a mode to the customer might not be a mode with respect to the design.**



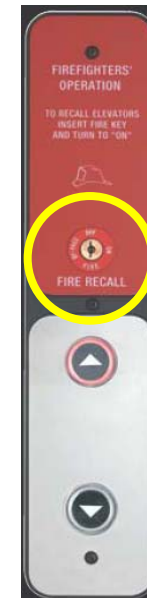
A Few Elevator Operating Modes

- ◆ **Parking / shutdown**
- ◆ **Maintenance / inspection modes**
- ◆ **Independent service**
 - Someone can control elevator with controls in the car
 - Ignore hall calls
 - Useful for movers, operators in fancy hotels or stores
- ◆ **Building security**
 - Restrict access to some floors
 - Card key or key code access only
 - Homing to lobby for visual inspection of the car
- ◆ **Maternity ward mode**
 - Babies are tagged with RFID bracelets
 - Elevator automatically returns to maternity ward if bracelet is detected
- ◆ **Sabbath / Streetcar mode**
 - Sabbath observance prohibits doing work, including pressing buttons
 - Behavior similar to the baseline dispatcher from course project



A Few Emergency Operating Modes

- ◆ Generally initiated by key switches, sensors, or special control panels
- ◆ Fire Modes – two stages
 - Stage 1 – return to lobby and clear the elevators
 - Stage 2 – rescue workers control the car
 - Different state and municipal elevator codes require different behaviors for car buttons
- ◆ Hospital operations – “Code blue”
 - Stage 1 – Override current calls and bring the elevator to a certain floor
 - Stage 2 – Independent service (controlled from car by hospital staff)
- ◆ Earthquake mode
 - Move the car in a direction that avoids passing the counterweight
- ◆ Riot mode
 - Exclude the lobby or lower floors to prohibit access to the rest of the building



Fire Stage 1



Fire Stage 2

Mode Interactions

- ◆ **What are the chances more than one mode is activated at a time?**
 - Fire in a hospital (Code Blue + Fire)
 - Fire during an earthquake (Earthquake + Fire)

- ◆ **Mode hierarchy**
 - Which mode “wins” when conditions to trigger multiple modes are true?
 - Complicated by complex trigger conditions
 - Stage 1 Fire is triggered by a key switch
 - Stage 2 Fire is triggered by a key switch + elevator doors open at lobby

- ◆ **Mode transitions**
 - k modes $\rightarrow k^2$ mode transitions (in the worst case)
 - How do you get the system into a well-defined state for mode transition?
 - How fast can you mode switch?
 - Response time may be an external requirement

Why distributed?

◆ Real elevators are complicated!

- Deal with differences in buildings and requested features
- Multiple operating and emergency modes

◆ Coming from a centralized event design mindset, distributed seems to make things more complex ... at first

- In our experience distributed + time triggered ends up with a more robust design

◆ Benefits of distributed architectures

- Fault tolerance / graceful degradation
- Flexibility
- Modularity

◆ Distributed requires up-front planning and good architecture

- Many open research problems exist in this area

Review

◆ For your project

- Commit point computations – needed for elevators that use FAST speed
 - Note that FAST speed changes in a few weeks to a higher fast speed
- Desired Floor / Desired Dwell messages – use them as intended
- Race conditions – resolve them in your design, not just your code

◆ Real-world elevators

- The course project is a good learning experience and gives a taste of things
- There are a lot of other problems to solve in full-complexity systems for the Real World

◆ Reminder

- Make sure your designs are 100% time triggered
- “Do once when you enter this state” and similar code hacks are forbidden

Excerpt from “50 Fun Things to Do in an Elevator”

- ◆ **Make race car noises when anyone gets on or off.**
- ◆ **When arriving at your floor, grunt and strain to yank the doors open, then act embarrassed when they open by themselves.**
- ◆ **Greet everyone getting on the elevator with a warm handshake and ask them to call you, "Admiral".**
- ◆ **Meow occasionally.**
- ◆ **Say, "Ding!" at each floor.**
- ◆ **Stare, grinning, at another passenger for a while, and then announce, "I've got new socks on!"**
- ◆ **Bet the other passengers you can fit a quarter in your nose.**
- ◆ **Holler, "Chutes away!" whenever the elevator descends.**
- ◆ **Bring a chair along.**

- ◆ **Disclaimer: The 18649 course staff do not advocate or recommend any of these practices.**