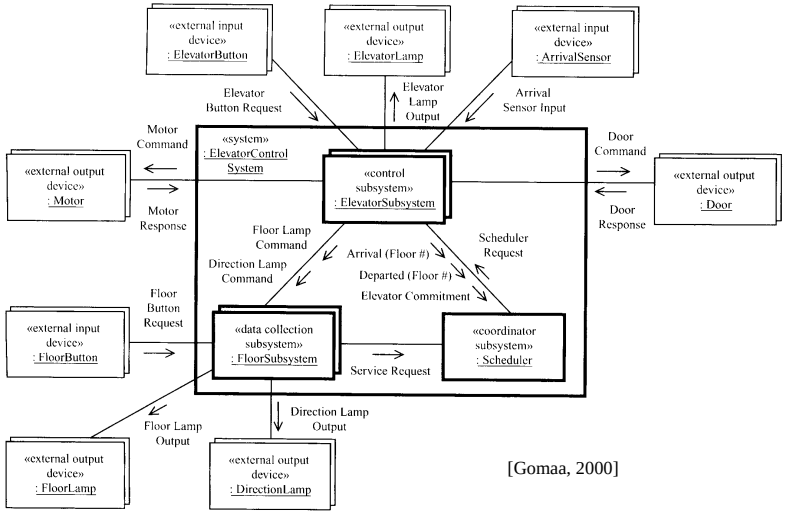
COMET DISTRIBUTED ELEVATOR CONTROLLER CASE STUDY

System Description:

The distributed system has multiple nodes interconnected via LAN and all communications between nodes are via loosely coupled message passing.

The overall system architecture consists of:

- Multiple instances of the Elevator Subsystem (one per Elevator)
- Multiple instances of the Floor Subsystem (one per floor)
- One instance of the Scheduler Subsystem



[Gomaa, 2000]

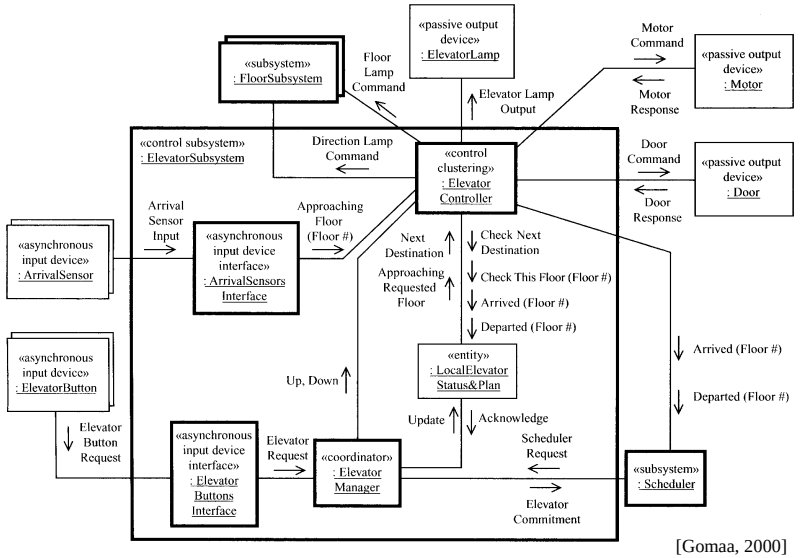
Unlike the non-distributed system, the Scheduler Subsystem and multiple instances of the Elevator Subsystem cannot communicate via shared memory, so there are two alternatives:

- Embed the Elevator Status & Plan data abstraction object in a server task - client tasks can send messages to the Status and Plan Server task → server could become a bottleneck under heavy load.

- Use replicated data - each instance of the Elevator Subsystem has its own copy of the Elavtor Status and Plan data abstraction → preferred option.

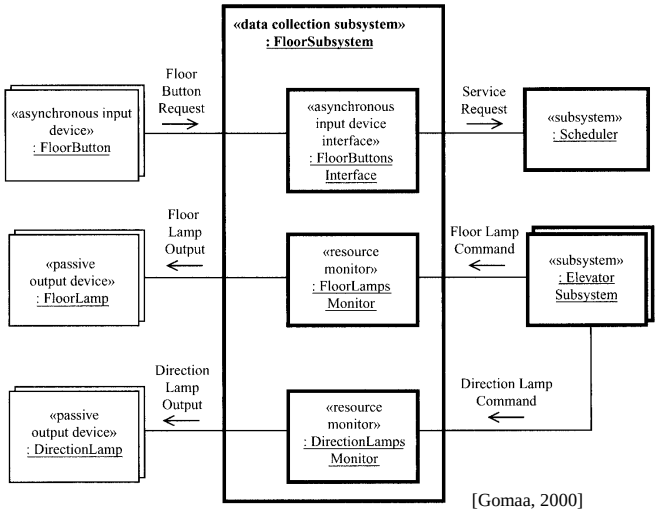
Subsystem structuring:

1. The Elevator Subsystem - each instance is made up of one instance of the Elevator Controller, Elevator Buttons Interface, Arrival Sensors Interface, and Elevator Manager tasks.



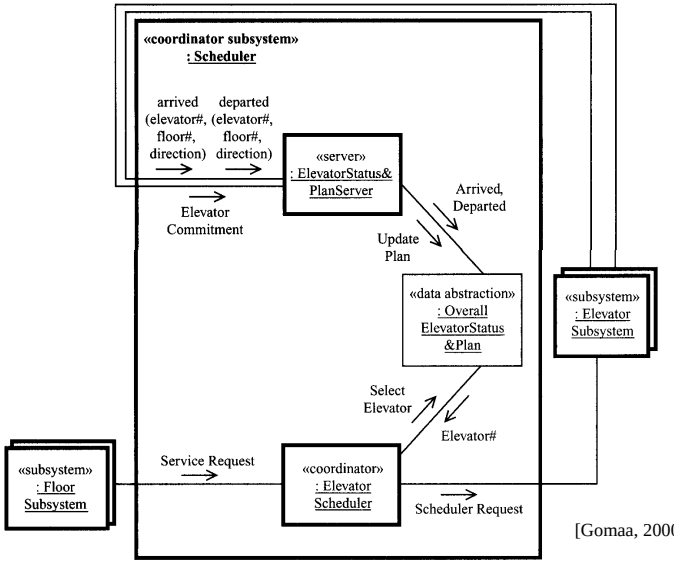
[Gomaa, 2000]

2. The Floor Subsystem - each instance is made up of one instance of the Floor Buttons Interface, Floor Lamps Monitor, and Direction Lamps Monitor tasks:



[Gomaa, 2000]

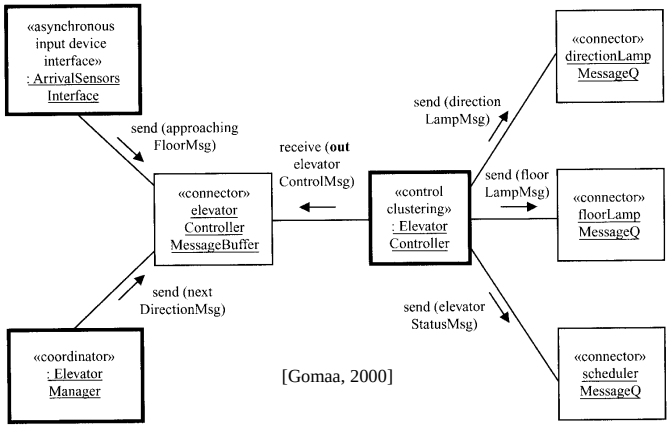
3. The Scheduler Subsystem - there is one instance of this subsystem which has the Elevator Status & Plan Server task and the Elevator Scheduler task:



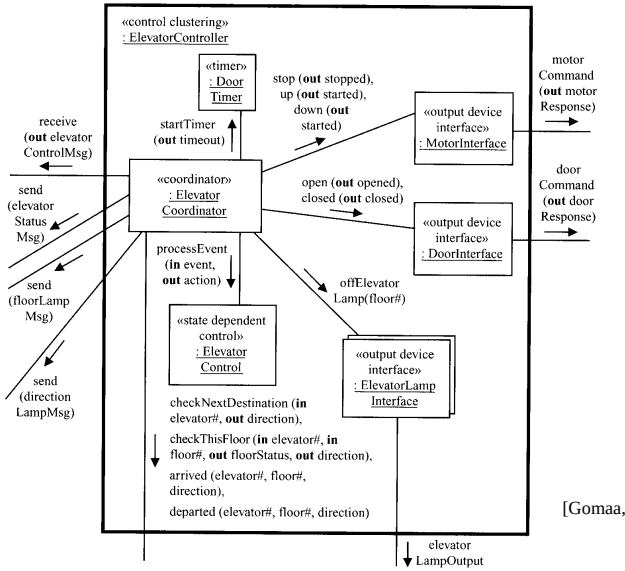
[Gomaa, 2000]

Detailed Software Design

Intertask messages are mapped to precise interfaces using connector objects where required to provide message buffering services or connectors to remote nodes:



[Gomaa, 2000]

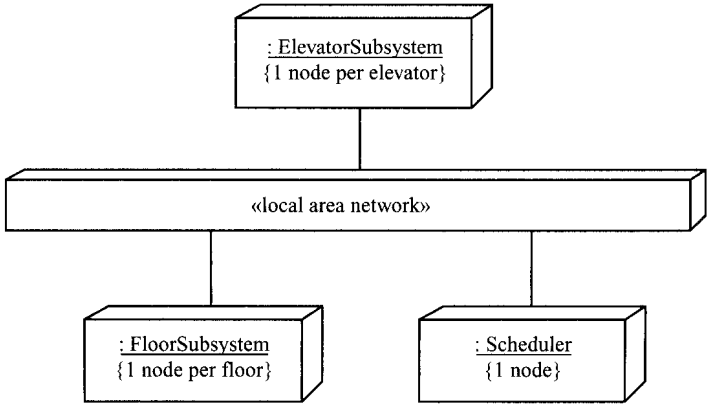


[Gomaa, 2000]

Target System Configuration

All subsystems must be mapped to physical nodes, and there are a number of possible mappings:

- One node per physical elevator (hence elevator subsystem), one node per physical floor (hence floor subsystem) and one node for the Scheduler subsystem.
- All floor subsystems are mapped to one node → the Floor Buttons Interface task could handle all floor Buttons, and similarly the Floor Lamps Monitor and Direction Lamps Monitor could handle all floor and direction lamps respectively.
- The Scheduler could be mapped to its own node or the floor subsystem node.



[Gomaa, 2000]

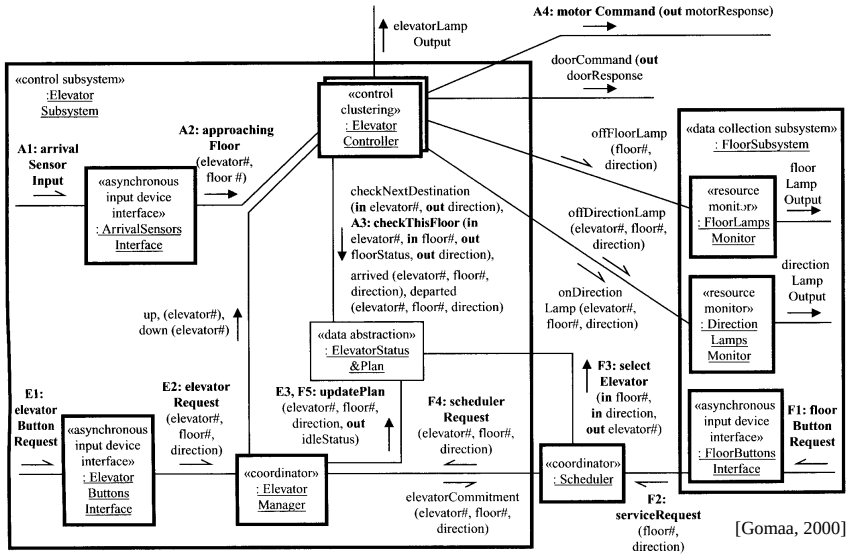
Performance Analysis Using RT Scheduling Theory

As an example for performance analysis assume we have three elevators and ten floors, and assume the following worst-case I/O scenario:

1. Elevator button interrupts - 10 Hz maximum rate corresponding to several users requesting floors at the same time → worst case 30 buttons in total can be pressed in three seconds.
2. Floor button interrupts - 5 Hz maximum rate corresponding to several users requesting floors at the same time → worst case 18 buttons in total can be pressed in 3.6 seconds.
3. All elevators arrive at floors at the same time → three floor arrival interrupts arrive within 50 ms of each other.

Event Sequences:

Consider the three critical event sequences on the TAD corresponding to the three external interrupt inputs shown above:



[Gomaa, 2000]

Stop Elevator at Floor event sequence (period T_a):

A1: *Arrival Sensors Interface* gets interrupt and processes it
A2: *Arrival Sensors Interface* sends *approaching Floor* message to the *Elevator Controller*
A3: *Elevator Controller* receives messages and checks the Elevator Status & Plan object to see if the elevator should stop
A4: *Elevator Controller* invokes *stop Motor* if it should stop

Select Destination event sequence (period T_b):

E1: *Elevator Buttons Interface* receives interrupt and processes it
E2: *Elevator Buttons Interface* sends *elevator Request* message to the *Elevator Manager*
E3: *Elevator Manager* receives message and records destination in *Elevator Status & Plan* object

Floor button pressed sequence (period T_c):

F1: *Floor Buttons Interface* receives interrupt and processes it
F2: *Floor Buttons Interface* sends *serviceRequest* message to the *Scheduler*
F3: *Scheduler* receives message and checks *Elevator Status and Plan* object to see if elevator is on the way to this, if not, then schedule one
F4: *Scheduler* sends a *schedulerRequest* message identifying the selected elevator to the *Elevator Manager*
F5: *Elevator Manager* receives message and records destination in *Elevator Status & Plan* object

Priority Assignments:

The aperiodic interrupt tasks are represented as periodic tasks with period set to the minimum event interarrival time. All task CPU times include context switch times, and message handling times are divided evenly between sender and receiver tasks.

Task	CPU time C_i (ms)	Period T_i (ms)	Util U_i	Priority
a. Floor Arrival Sequence				
Arrival Sensors Interface	2	50	0.04	1
Elevator Controller	5	50	0.10	4
b. Elevator Button Sequence				
Elevator Buttons Interface	3	100	0.03	2
Elevator Manager	6	100	0.06	5
c. Floor Button Sequence				
Floor Buttons Interface	4	200	0.02	3
Scheduler	20	200	0.10	6
Elevator Manager	6	200	0.03	5
Other Tasks				
Floor Lamps Monitor	5	500	0.01	7
Direction Lamps Monitor	5	500	0.01	8

Notes:
1. Periods of all tasks in the same event sequence are the same.
2. All interrupt driven tasks are given the highest priority → violates rate monotonic priority assignments.
3. All other tasks are allocated their rate monotonic priorities.

Real-time Scheduling:

The total utilisation is $0.4 < 0.69$, but because of the non rate monotonic regime → a more detailed analysis is required.

Real-time Scheduling analysis of non-distributed system:

A. Stop Elevator at Floor event sequence - T_a = 50 ms
   a) Tasks in sequence utilization = (2 ms + 5 ms)/50 ms = 0.14
   b) Higher priority task with longer period pre-emption utilization (e.g. *Elevator Buttons Interface* and *Floor Buttons Interface* pre-empts *Elevator Controller*) = (3 ms + 4 ms)/50 ms = 0.14
   c) Lower priority task blocking time utilization (e.g. worst-case is *Scheduler* task) = 20 ms/50 ms = 0.40

Total utilization = 0.68 < 0.69 → schedulable by GUBT

B. Select Destination event sequence - T_b = 100 ms
   a) Tasks in sequence utilization = (3 ms + 6 ms)/100 ms = 0.09
   b) Higher priority task with shorter period pre-emption utilization (e.g. *Arrival Sensors Interface* and *Elevator Controller* pre-empts *Elevator Manager* and they can both execute twice in 100 ms) = 2 × (2 ms + 5 ms)/100 ms = 0.14
   c) Higher priority task with longer period pre-emption utilization (e.g. *Floor Buttons Interface* pre-empts *Elevator Manager*) = 4 ms/100 ms = 0.04
   d) Lower priority task blocking time utilization (e.g. worst-case is *Scheduler* task) = 20 ms/100 ms = 0.20

Total utilization = 0.47 < 0.69 → schedulable by GUBT

C. Request Elevator event sequence - T_c = 200 ms
   a) Tasks in sequence utilization = (4 + 20 + 6) ms/200 ms = 0.15
   b) Higher priority task with shorter period pre-emption utilization (e.g. *Arrival Sensors Interface* and *Elevator Controller* pre-empts *Elevator Manager* and *Scheduler* and they can both execute 4 times in 200 ms) = 4 × (2 ms + 5 ms)/200 ms = 0.14. (e.g. *Elevator Buttons Interface* and *Elevator Manager* pre-empts *Scheduler* and they can both execute twice in 200 ms) = 2 × (3 ms + 6 ms)/200 ms = 0.09
   c) Lower priority task blocking time utilization (e.g. worst-case is *Scheduler* task - but in this event sequence anyway)

Total utilization = 0.38 < 0.69 → schedulable by GUBT

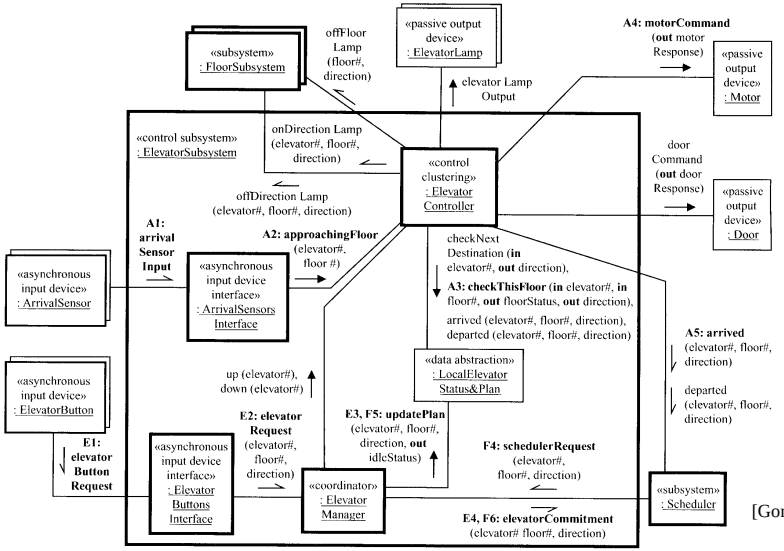
Performance Analysis of the Distributed System

Consider a simplistic scaling up of the multiple elevator system, e.g. for 3 elevators and 10 floors we have a utilisation of 0.4, and a 12 elevator and 40 floors system would have utilisation of 1.2 → not possible with one CPU.

For the distributed system, assume the following:

1. We have a distributed system with one node per elevator, one node per floor, and one scheduler.
2. The same processor types are used so that task execution times do not change.
3. A dedicated deterministic LAN is used with a 10 Mbit/sec capacity and a worst case message length of 2 Kbytes. There is also a 1 ms protocol assembly and 1ms protocol disassembly overhead.

Consider an event sequence diagram for the distributed system:



[Gomaa, 2000]

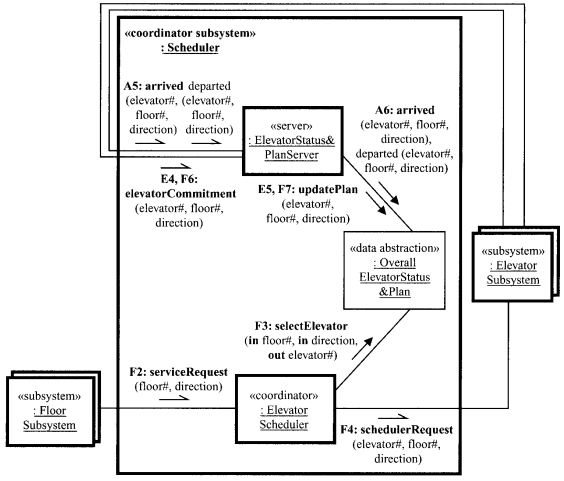
Each of the task parameters can now be defined for each of the subsystems in the distributed system:

Task	CPU time C_n (ms)	Period T_n (ms)	Util U_n	Priority
Elevator Subsystem				
Arrival Sensors Interface	2	50	0.04	1
Elevator Controller	5	50	0.10	3
Elevator Buttons Interface	3	100	0.03	2
Elevator Manager	6	100	0.06	4
Floor Subsystem				
Floor Buttons Interface	4	200	0.02	1
Floor Lamps Monitor	5	500	0.01	2
Direction Lamps Monitor	5	500	0.01	3
Scheduler Subsystem				
Elevator Status & Plan Server	2	30	0.20	1
Elevator Scheduler	20	50	0.40	2

Scheduler subsystem event sequences are: [Gomaa, 2000]

Total CPU utilization:
Elevator Subs. = 0.23
Floor Subs. = 0.04
Scheduler Subs. = 0.6

Note that with four times as many floors:
1. Scheduler period reduced to 50 ms from 200 ms
2. Elevator Status and Plan Server has a period of 30 ms and execution time of 2 ms.



Analyse each event sequence (A, E, F) in turn for the distributed system via the Generalized Utilization Bound Theorem (GUBT) . . .

Real-time scheduling analysis for the distributed system

Event sequence analysis for the distributed system via the Generalized Utilization Bound Theorem (GUBT):

A. Stop Elevator at Floor event sequence (A) - tasks are located in the Elevator and Scheduler Subsystems, T_A = 50 ms

1. In Elevator Subsystem:

a) Tasks in sequence utilization = (2 ms + 5 ms)/50 ms = 0.14

b) Higher priority task with longer period pre-emption utilization (e.g. *Elevator Buttons Interface* pre-empts *Elevator Controller*) = 3 ms/50 ms = 0.06

c) Lower priority task blocking time utilization (e.g. worst-case is *Elevator Manager* task) = 6 ms/50 ms = 0.12

Total utilization = 0.32 < 0.69 → schedulable by GUBT
Total elapsed time in Elevator Subsystem = 16 ms < 50 ms

2. In Scheduler Subsystem:

a) *Elevator Status & Plan Server* receives message. Assume 1 ms for receiving and processing → C_m = 1 ms

b) *Elevator Status and Plan Server* calls the *Overall Status & Plan* object → C_s = 2 ms

c) Possible blocking time by Elevator Scheduler → B_s = 20 ms

Total elapsed time in Scheduler Subsystem = 23 ms

3. Network communication:
Network message, 2000 bytes at 10Mb/s → D_t = 2 ms

Total worst-case time to service the Stop Elevator at Floor event sequence = 41 ms < 50 ms

B. Select Destination event sequence (E) - tasks are located in the Elevator and Scheduler Subsystems, T_E = 100 ms

a) Tasks in sequence utilization = (3 ms + 6 ms)/100 ms = 0.09

b) Higher priority task with shorter period pre-emption utilization (e.g. *Arrival Sensors Interface* and *Elevator Controller* pre-empts *Elevator Manager* and they can both execute twice in 100 ms) = 2 × (2 ms + 5 ms)/100 ms = 0.14
c) There are no higher priority tasks with longer periods
d) There are no other lower priority tasks that can cause a blocking time utilization

Total utilization = 0.23 < 0.69 → schedulable by GUBT
Total elapsed time in Elevator Subsystem = 23 ms < 100 ms

2. In Scheduler Subsystem:
Same event sequence (E4 & E5) times as previously
Total elapsed time in Scheduler Subsystem = 23 ms

3. Network communication:
Network message, 2000 bytes at 10Mb/s → D_t = 2 ms

Total worst-case time to service the Select Destination event sequence = 48 ms < 100 ms

C. Request Elevator event sequence (F) - now the tasks are located across all subsystems, T_F = 200 ms:

F1: *Floor Buttons Interface* receives interrupt and processes it, as highest priority - no pre-emption or blocking → execution time C_f = 4 ms
F2: *Floor Buttons Interface* sends message, allow message protocol assembly overhead, C_m = 1 ms

F2.1: Network sends *serviceRequest* message, 2000 bytes at 10Mb/s → D_t = 2 ms

Total Floor Subsystem time $E_f = C_f + C_m + D_t$ = 7 ms

F2.2 : *Elevator Scheduler* receives message, protocol disassembly overhead, C_m = 1 ms

F3: *Elevator Scheduler* interrogates *Overall Elevator Status and Plan* object to see if elevator is on the way to this floor, if not, then schedule one, C_s = 20 ms. Then Scheduler assembles message and sends it, C_m = 1 ms.

Possible blocking time by *Elevator & Status Plan Server*: B_s = 2 ms

Total Scheduler Subsystem time: $E_s = C_m + C_s + C_m + B_s$ = 24 ms

F3.1: Network sends *serviceRequest* message, 2000 bytes at 10Mb/s → D_t = 2 ms

F4: *Elevator Manager* receives message and disassembles it, C_m = 1 ms

F5: *Elevator Manager* then records destination in *Local Elevator Status & Plan* object, C_e = 6 ms

F6: *Elevator Manager* assembles *elevatorCommitment* message for the *Scheduler* Subsystem, C_m = 1 ms

Possible pre-emption time: *Arrival Sensors Interface* (2 ms), *Elevator Controller* (5 ms), and *Elevator Buttons Interface* (3 ms), and *Elevator Manager* (6 ms, but handling an elevator button message) can all pre-empt, total time of C_p = 16 ms

Possible blocking time: blocking over the *Local Elevator Status & Plan* store can only be performed by the two tasks already included in the above time

Total Elevator Subsystem time: $E_e = C_m + C_e + C_m + C_p$ = 24 ms

F6.1: Network sends *elevatorCommitment* message to *Scheduler* Subsystem, D_t = 2 ms

F7: The *Elevator Status & Plan Server* disassembles the message (C_m = 1 ms) and updates the *Overall Status & Plan* object store (2

ms) but may be blocked by one lower priority task (in this case the *Scheduler* (20 ms)) $\rightarrow C_u = 22$ ms

Total Overall Status & Plan Update time: $E_u = C_m + C_u = 23$ ms

Combining all of the delays for the Floor Button Press event sequence gives a time of:

$E_f + D_t + E_s + D_t + E_e + D_t + E_u = 82$ ms

Thus the worst case service time for a Floor button press event sequence is well below the specified 200 ms response time.