**Part 1**

*Overview*

The *Ambulance Dispatch* model is supposed to model the process that occurs from receiving a emergency call till the ambulance completes the call. This is done in the following steps; a call is placed to the emergency services where information is gathered about the caller or incident (For this model this is represented by the location and priority of the call), next the ambulance nearest to the caller is sent in response to the call and finally an outcome occurs (For this model the outcome is just living or dying). The model, as seen in figure 1, will have three main sections. The first is the *Call Handler* which is responsible for receiving calls, queuing the calls if necessary, gathering the necessary information and outputting the location and priority of the call. The next section is the *Dispatch* which is responsible for taking the information generated by the *Call Handler* and determining which ambulances are available and dispatching the closest one. As well it will determine the response time the ambulance will take based off the distance of the ambulance from the location. Finally the *Response* section gathers all the information generated and determines the overall outcome of the events. If the ambulance is close enough the outcome would be the caller living. Otherwise, if the ambulance is too far away and cannot respond in time or there are no more ambulances left then the outcome would be that the caller dies.

Ambulance Dispatch

Figure Ambulance Dispatch Flowchart

CALL HANDLER

LOCATION

Location

PRIORITY

CALL QUEUE

Priority

CALL OUT

CALL IN

Ambulance

Response

# Part II

## Formal Specifications

The formal specifications <S, X, Y, δint, δext, λ, ta> for the atomic models are defined as follows:

### CALLER QUEUE

S = {passive, active}

X = {callin, done}

Y = {callout}

δint (active) = passive

δext (callin, passive) = active

δext (callin, active) = active

δext (done, passive) = active

δext (done, active) = active

λ(active)

{ stack *callin* until *done* then send to port *callout* }

ta(passive) = INFINITY

ta(active) = 00:00:00:00

### CALL LENGTH

S = {passive, active}

X = { }

Y = {out}

δint (active) = active

δext (out, active) = passive

λ(active)

{ send a output to port *out* with a uniform distribution with a mean of 30 seconds}

ta(passive) = INFINITY

ta(active) = 00:00:00:00

### CALL INFO

S = {passive, active}

X = {callin }

Y = {location, priority}

δint (active) = passive

δext (callin, passive) = active

λ(active)

{ take modulus 10 of *callin* and send to port *location*

divide and truncate *callin* by 10 and send to port *priority* }

ta(passive) = INFINITY

ta(active) = 00:00:00:00

### DISPATCH

S = {passive, active}

X = {location, priority}

Y = {ambulance, time\_out}

δint (active) = passive

δext (location, passive) = active

δext (priority, passive) = active

λ(active)

{ take the input from *location*  and select the nearest ambulance and send to port *ambulance*

take the input from *priority* and the *location* and determine the response time to send to port *time\_out* }

ta(passive) = INFINITY

ta(active) = 00:00:00:00

### RESPONSE

S = {passive, active}

X = {time\_in, priority}

Y = {response}

δint (active) = passive

δext (time\_in, passive) = active

δext (priority, passive) = active

λ(active)

{ take the *time\_in* and the *priority* to determine at what time the ambulance will arrive to the caller and whether the caller will live, 1, or die, 0, and send to port *response* }

ta(passive) = INFINITY

ta(active) = 00:00:00:00

The formal specifications <X, Y, D, {Mi}, {Ii}, {Zij}, SELECT > for the coupled model Network and ABP Simulator are defined as follows:

### Call Handler

X = {callin};

Y = {location, priority};

D = {Call Queue, Call Info};

I(Call Queue) = {self};

I(Call Info) = {Call Queue};

Z(Call Queue) = { Call Info };

Z(Call Info) = { self };

### Call Queue

X = {callin};

Y = {callout};

D = {Call Length, Call Queue};

I(Call Queue) = { self };

Z(Call Length) = { Call Queue };

Z(Call Queue) = { self };

### Ambulance Dispatch

X = {callin};

Y = {location, priority, ambulance, response};

D = {Call Handler, Dispatch, Response};

I(Call Handler) = {self};

I(Dispatch) = {Call Handler};

I(Response) = { Dispatch };

Z(Call Handler) = { Dispatch };

Z(Dispatch) = { Response };

Z(Response) = { self };

## Test Strategies

As the atomic models where generated they were first tested individually using the "black box" testing method. The same process was applied for the coupled models and finally the top level model. This ensured that all the pieces worked before the final assembly and final simulation.

# Part III

## Tests and Analysis

### ATOMIC MODEL: *CallerINFO*

The atomic model, *CallerINFO*, is supposed to take a two digit input and generate a location and a priority. The value in the *tens* position is considered the priority while the value in the *ones* positions is the location. Since there are 9 priority levels and 9 locations all values from 11-99 correspond to all the combinations for the location or priority. A input of 00 will initialize or reset the number of available ambulances or if 00 appears in the output, it represents a NULL or *false* input. The following is the event file used for the test:

00:00:00:00 in 00

00:00:10:00 in 11

00:00:20:00 in 21

00:00:20:00 in 31

00:00:30:00 in 41

00:00:40:00 in 51

**00:00:50:00 in 00**

**00:00:51:00 in ab**

00:00:52:00 in 81

**00:00:53:00 in 181**

00:00:54:00 in 21

00:00:55:00 in 31

**00:00:56:00 in 1**

00:00:57:00 in 31

These inputs resulted in the following outputs:

00:00:00:000 location 0

00:00:00:000 priority 0

00:00:10:000 location 1

00:00:10:000 priority 1

00:00:20:000 location 1

00:00:20:000 priority 3

00:00:30:000 location 1

00:00:30:000 priority 4

00:00:40:000 location 1

00:00:40:000 priority 5

**00:00:50:000 location 0**

**00:00:50:000 priority 0**

**00:00:51:000 location 0**

**00:00:51:000 priority 0**

00:00:52:000 location 1

00:00:52:000 priority 8

**00:00:53:000 location 0**

**00:00:53:000 priority 0**

00:00:54:000 location 1

00:00:54:000 priority 2

00:00:55:000 location 1

00:00:55:000 priority 3

**00:00:56:000 location 0**

**00:00:56:000 priority 0**

00:00:57:000 location 1

00:00:57:000 priority 3

Three false test conditions were sent through the model. The first was of appropriate length but false characters and resulted in a NULL output. The second and third input, as seen in bold, were of incorrect length and resulted in a NULL output.

### ATOMIC MODEL: *Dispatch*

The purpose of the atomic model, *Dispatch*, is to receive the *location* and *priority* information from the coupled model *CallHandler* and determine which ambulance is closest to the call location and dispatch it, as well as to calculate how long it will take for the ambulance to arrive. The *location* and *priority* arrive as two individual single digit integers. The *location* values 1-9 correspond to a location on a three dimensional grid, as seen in Table 1.

Table

|  |  |  |
| --- | --- | --- |
| Location = 1  Ambulance = 1 | Location = 2  Ambulance = 2 | Location = 3  Ambulance = 3 |
| Location = 4  Ambulance = 4 | Location = 5  Ambulance = 5 | Location = 6  Ambulance = 6 |
| Location = 7  Ambulance = 7 | Location = 8  Ambulance = 8 | Location = 9  Ambulance = 9 |

The *priority* value is converted into a response time by the equation seen in Equation 1.

Equation

When the model receives a *location* and *priority* value of 0 or runs out of ambulances it is to reset the ambulance availability table. The following event file was used to test the model:

00:00:00:00 location 0

00:00:00:00 priority 0

00:00:01:00 location 1

00:00:01:00 priority 1

00:00:02:00 location 2

00:00:02:00 priority 2

00:00:03:00 location 3

00:00:03:00 priority 3

00:00:04:00 location 4

00:00:04:00 priority 4

00:00:05:00 location 5

00:00:05:00 priority 5

00:00:06:00 location 6

00:00:06:00 priority 6

00:00:07:00 location 7

00:00:07:00 priority 7

00:00:08:00 location 8

00:00:08:00 priority 8

00:00:09:00 location 9

00:00:09:00 priority 9

**00:00:10:00 location 1 There should be no more ambulances and therefore the**

**00:00:10:00 priority 1 table should be reset**

**00:00:12:00 location 2 Since the table was reset this should work**

**00:00:12:00 priority 2**

**00:00:13:00 location 0 Reset the system**

**00:00:13:00 priority 0**

**00:00:14:00 location 2 Since the table was reset ambulance 2 should be**

**00:00:14:00 priority 2 available again**

**00:00:15:00 location 22 Should not generate an output**

**00:00:15:00 priority 22**

00:00:16:00 location 2

00:00:16:00 priority 2

00:00:17:00 location 2

00:00:17:00 priority 2

00:00:18:00 location 2

00:00:18:00 priority 2

00:00:19:00 location 2

00:00:19:00 priority 2

00:00:20:00 location 2

00:00:20:00 priority 2

00:00:21:00 location 2

00:00:21:00 priority 2

00:00:22:00 location 2

00:00:22:00 priority 2

This generated the following output:

00:00:00:000 ambulance 0

00:00:00:000 response\_time 0

00:00:01:000 ambulance 1

00:00:01:000 response\_time 660

00:00:02:000 ambulance 2

00:00:02:000 response\_time 360

00:00:03:000 ambulance 3

00:00:03:000 response\_time 240

00:00:04:000 ambulance 4

00:00:04:000 response\_time 180

00:00:05:000 ambulance 5

00:00:05:000 response\_time 180

00:00:06:000 ambulance 6

00:00:06:000 response\_time 120

00:00:07:000 ambulance 7

00:00:07:000 response\_time 120

00:00:08:000 ambulance 8

00:00:08:000 response\_time 120

00:00:09:000 ambulance 9

00:00:09:000 response\_time 120

**00:00:10:000 ambulance 0**

**00:00:10:000 response\_time 0**

**00:00:12:000 ambulance 2**

**00:00:12:000 response\_time 360**

**00:00:13:000 ambulance 0**

**00:00:13:000 response\_time 0**

**00:00:14:000 ambulance 2**

**00:00:14:000 response\_time 360**

**00:00:16:000 ambulance 1**

**00:00:16:000 response\_time 540**

00:00:17:000 ambulance 5

00:00:17:000 response\_time 540

00:00:18:000 ambulance 3

00:00:18:000 response\_time 540

00:00:19:000 ambulance 4

00:00:19:000 response\_time 614

00:00:20:000 ambulance 6

00:00:20:000 response\_time 614

00:00:21:000 ambulance 8

00:00:21:000 response\_time 720

00:00:22:000 ambulance 7

00:00:22:000 response\_time 762

When the table ran out of ambulances it was reset properly as seen at time 00:00:12:00. The system reset at time 00:00:13:00 worked since at time 00:00:14:00 ambulance 2 was available for use again. Finally the false input at time 00:00:15:00 did not generate an output.

### ATOMIC MODEL: *Response*

The purpose of the *Response* model is to determine the outcome of the call. If the ambulance arrives in time, as determined by Equation 2, the output will be a 1, or survival. If the ambulance takes too long to respond to the call the output will be a 0, or death.

Equation

The following event file was used to test the model:

**00:00:00:00 priority 0 NULL VALUE**

**00:00:00:00 time 0**

00:00:01:00 priority 9

00:00:01:00 time 120

**00:00:02:00 priority 5 Die**

**00:00:02:00 time 450**

**00:00:03:00 priority 1 Live**

**00:00:03:00 time 800**

00:00:04:00 priority 4

00:00:04:00 time 700

**00:00:06:00 priority 99 Exceeds parameters**

**00:00:06:00 time 1000**

**00:00:07:00 priority 99**

**00:00:07:00 time 0 False Time**

This resulted in the following output:

00:02:01:000 outcome 1

**00:07:32:000 outcome 0**

**00:13:23:000 outcome 1**

00:11:44:000 outcome 0

00:16:45:000 outcome 0

As seen in the output above any non-values resulted in no output being generated as well the model correctly determined the results for the calls.

### TOP MODEL: *Ambulance Dispatch*

The following event file was used to test the entire model:

**1 00:00:00:00 in 00 NULL input**

**2 00:00:10:00 in 11 Live**

3 00:00:20:00 in 21

4 00:00:20:00 in 31

5 00:00:20:00 in 41

**6 00:00:20:00 in 51 Die**

7 00:00:20:00 in 51

8 00:00:20:00 in 51

9 00:00:20:00 in 51

10 00:00:20:00 in 51

**11 00:00:20:00 in 51 Resets the system and therefore a NULL input**

**12 00:00:20:00 in 00 NULL input**

**13 00:00:20:00 in 51 Shows the system was reset**

**14 00:00:20:00 in ab**

15 00:00:20:00 in 71

**16 00:00:20:00 in 811 Value out of range**

**17 00:00:20:00 in 1 Value out of range**

18 00:00:20:00 in 11

**19 00:00:21:00 in 00 NULL input**

20 00:00:25:00 in 91

21 00:00:30:00 in 81

This resulted in the following output being generated:

**2 00:11:10:000 response\_out 1**

3 00:09:29:816 response\_out 1

4 00:06:41:757 response\_out 1

5 00:06:03:124 response\_out 1

**6 00:07:26:655 response\_out 0**

7 00:07:44:393 response\_out 0

8 00:08:37:803 response\_out 0

9 00:09:20:551 response\_out 0

10 00:10:53:379 response\_out 0

**13 00:08:04:541 response\_out 1**

15 00:07:45:677 response\_out 1

18 00:19:02:443 response\_out 1

20 00:10:56:710 response\_out 1

21 00:12:07:271 response\_out 1

As you can see 7 false or NULL inputs were given, as seen in bold in the event file, and they did not generate any outputs, as seen above. As well the model generated the proper outputs for the living and dying conditions.