

# Supplementary Materials for Hybrid Planning Using Learning and Model Checking for Autonomous Systems

## SUMMARY OF SUPPLEMENTARY MATERIALS

This document contains the hybrid planning algorithm and supplementary materials for the two case studies of learning-based hybrid planning. Structured as a series of questions and answers, this information can be used as further evidence of the studies' findings and for reproduction purposes.

The material starts with hybrid planning algorithm. Then for both case studies we provide the specific parameter values, planning specifications, and additional analyses on the dependencies between planning performances. In the end, we also provide visualizations of load traces in the first case study.

## CONTENTS

<b>I</b>	<b>Hybrid Planning Algorithm</b>	1
<b>II</b>	<b>Materials for the Cloud-based System</b>	2
<b>III</b>	<b>Materials for the Team of UAVs</b>	3
<b>IV</b>	<b>Materials for Data Analysis</b>	4
<b>V</b>	<b>PRISM Planning Specifications</b>	5
<b>VI</b>	<b>Materials for Trace Patterns</b>	26
	<b>References</b>	41

## I. HYBRID PLANNING ALGORITHM

This section describes a general algorithm, executed in hybrid planners, that can be used in either existing condition-based approaches or our novel learning-based approach. This algorithm is an extension of the algorithm proposed by other researchers since our algorithm (a) can handle more than one reactive approach, (b) is not restricted to a specific combination of reactive and deliberative planning, and (c) not limited to the condition-based (i.e., constraint violations) approach. For the purposes of this paper, the algorithm serves as a standardized execution context of both HP approaches for the empirical study.

The goal of the hybrid planning algorithm (see Algorithm 1) is to determine a global plan of system adaptation (stored in variable  $\pi$ , protected from race conditions by a mutex  $\mu$ ) using a combination of some reactive planning approach ( $\rho_r \in \mathcal{F}$ ) and deliberative planning. A *plan* is a partial function  $P : S \rightarrow A$ ,

where  $S$  is a set of all possible states reachable from the current state, and  $A$  is a set of all possible adaptation tactics. A plan maps a state  $s \in S$  to a tactic  $a \in A$  to be executed in  $s$ . The input to the algorithm is a planning problem ( $\xi$ ) that contains the current state of the system ( $\xi.s_{curr}$ ).

---

**Algorithm 1** A general hybrid planning algorithm.

---

```

1: global  $\pi \leftarrow null$  ▷ System's plan for execution
2: global  $\mu \leftarrow new$  Mutex ▷ Mutex for  $\pi$ 
3: global  $\mathcal{T} \leftarrow new$  Thread ▷ Deliberative thread
4: function HYBRIDPLANNING(Problem  $\xi$ , Planners  $\mathcal{F}$ , Planner  $\rho_d$ )
5:    $\mu.lock()$ 
6:   if  $\pi \neq null$  and  $\pi.has(\xi.s_{curr})$  then
7:      $\mu.unlock()$ 
8:     return ▷ Replan only if needed
9:
10:    ▷ Pick an appropriate reactive planning approach
11:    Planner  $\rho_r = PICKREACTIVEPLANNING(\xi, \mathcal{F}, \rho_d)$ 
12:     $\pi \leftarrow \rho_r.PLAN(\xi)$  ▷ Determine the reactive plan
13:     $\mu.unlock()$ 
14:
15:    if not  $\mathcal{T}.isRunning()$  then
16:       $\mathcal{T}.run$  [ ▷ Deliberate in the background
17:         $\pi' \leftarrow \rho_d.PLAN(\xi)$ 
18:         $\mu.lock()$ 
19:         $\pi \leftarrow \pi'$ 
20:         $\mu.unlock()$  ]

```

---

To find a plan corresponding to the problem  $\xi$ , the HYBRID-PLANNING algorithm first refers to the existing plan (line 6). If a plan is present and matches the current state (i.e., contains  $\xi.s_{curr}$ ), then the algorithm does not change  $\pi$ . However, if the plan does not exist or  $\xi.s_{curr}$  is not in the plan, then the planner computes a suitable new plan (lines 10–20).

First, the algorithm needs to decide its instantaneous response (lines 10–13), which requires choosing an appropriate reactive planning approach  $\rho_r \in \mathcal{F}$ . This decision is made by the function PICKREACTIVEPLANNING, the role of which is to solve PLANSEL. This function can be implemented by checking predefined conditions on  $\xi$ , or by learning which reactive approach is (the most) suitable for a given planning problem.

Regardless of the above decision, deliberative planning (function DELIBERATIVEPLANNING) is started afterwards in

a separate thread or, generally, computation stream ( $\mathcal{T}$ , lines 16–18), in order to eventually arrive at a plan that is expected to yield higher utility than any of the reactive planning approaches. For a planning problem, deliberative planning is invoked once, allowing only one thread at a time. Once the computation of the deliberative plan is complete, the system’s plan is thread-safely updated to it (17–19). As discussed earlier, the structure of the plan enables a smooth transition (i.e., the global plan  $\pi$  is updated) from a reactive to a deliberative plan, thus resolving PLNCRD.

Algorithm 1 is implemented by a self-adaptive framework, which can use the global variables  $\pi$ ,  $\mu$ , and  $\mathcal{T}$  to configure HYBRIDPLANNING. For example, to find a reactive plan without stopping deliberative planning, the framework can lock  $\mu$ , set  $\pi$  to *null* and, if  $\mathcal{T}$  is still running, execute HYBRIDPLANNING. If needed, deliberative execution can be reset by stopping  $\mathcal{T}$ . Therefore, by setting the global variables appropriately, frameworks can apply the algorithm in various scenarios of self-repair and self-optimization.

A critical part of the hybrid planning logic is realized by the function PICKREACTIVEPLANNING, which solves PLNSEL in the above algorithm. In idealized conditions (when deliberative planning is always better than reactive planning, and PICKREACTIVEPLANNING picks the best reactive planning approach for a given  $\xi$ ), hybrid planning will never provide a lower expected utility compared to utilities provided by reactive and deliberative planning used alone. Intuitively, any reactive planner is no better than the choice of PICKREACTIVEPLANNING, and then it is dominated by a deliberative plan. The only way for deliberative planning to be better than hybrid planning is by avoiding inappropriate reactive planning invocation, which is also done by PICKREACTIVEPLANNING returning  $\rho_{wait}$ .

In practice, implementations of PICKREACTIVEPLANNING solve PLNSEL imperfectly, affecting the performance of HP. To solve PLNSEL, prior work used fixed conditions on the system’s state to decide whether to use reactive planning, evaluating a predicate  $P(\xi.s_{curr})$  defined by a domain expert. For instance, a reactive planner is used when the response time is above some threshold; otherwise, the system should wait. Learning-based approach is another potential implementation of PICKREACTIVEPLANNING.

## II. MATERIALS FOR THE CLOUD-BASED SYSTEM

This section contains the concrete parameters and specifications for the cloud-based load balancing system.

*Q: What Are the Exact Parameter Values for the Cloud-based System?*

In our experiments, the parameters values are the same as in our previous work where we evaluated condition-based hybrid planning [1]. We set the time-related parameters as follows: the systems evaluates the need for adaptation at each minute, and determines an action, if adaptation is needed. Some actions take time before their effect – any server boots up time is 2 minutes. The look-ahead horizon for deliberative planning (i.e.,  $\rho_{mdp}$ ) is chosen to be 5 minutes. This heuristic gives a long enough

horizon to go from one active server to three active servers, and one additional evaluation cycle to observe the resulting utility.

The system has three types of servers: A, B and C. Servers of type A are the most expensive, but have the highest capacity to handle requests. Servers of type C are the cheapest, but have the least request-handling capacity. The time to serve requests is normally distributed, depending on a server’s capacity. The costs and capacities are assigned according to Table I. The ratio between cost and capacity is constant, which is inspired by the cost model of AWS (<https://aws.amazon.com/ec2/pricing/on-demand>) where the system capacity improves in the same ratio as the increase in cost. In the experiments we have three dimmer levels and one server of each type.

Server type	Cost (units per minute)	Capacity (requests per minute)	
		With Optional Content	Without Optional Content
A	1.0	200	400
B	0.7	140	280
C	0.5	100	200

Table I  
COST/CAPACITY PARAMETERS FOR EACH SERVER TYPE.

In our experiments, the cost of a server can be covered by the revenue of handling 1/10 of its maximum capacity with optional content and the revenue of handling 2/3 of its maximum capacity without optional content. If the server cost per minute is  $C$ , capacity with optional content is  $c_O$  and without optional content is  $c_M$ , then the revenue for a server with optional content is  $R_O = \frac{10}{c_O}C$  and without optional content would be  $R_M = \frac{3/2}{c_M}C$ . For each request having response time above the threshold of 1 second, there is a penalty of -0.25 units. We assume the time for a server to serve a request is normally distributed with the mean as the server’s capacity and the variance as the maximum possible delay calculated using a variation of M/G/1/PS queueing model that supports different capacity servers operating in parallel.

When looking up the current state in a plan (Line 6 in the generalized HP algorithm, see Algorithm 1 in the main paper), the cloud-based system needs to deal with the possibility of not finding any matches; in such cases the plan fails and needs to be recomputed. As mentioned earlier, planning is done based on predicted request arrival rate; not the actual values. Since the prediction discretizes the values, it is very likely that the actual is not one of the discrete values. To account for this situation, we used a matching heuristic that needs to be applied to states in order not to discard plans in almost every transition. We use two criteria for states in a plan being matched to a given (current) state: (1) all state variables (except request arrival rate) have the same values; (2) the rate is within  $\min(0.5 * current\_rate, 100)$  of the current arrival rate. (We found that this criterion provides a reasonable balance between matching states and failing plans in our experiments.) If no state meeting both criteria is found in a plan, the matching fails. If several states meet both criteria, one that minimizes the difference between request arrival rates is picked.

We conducted the experiments on a Ubuntu 14.04 virtual machine having 8.5 GB RAM and 4 processors at 2.9 GHz. The state space for deterministic planning (i.e.,  $\rho_{det}$ ) varies between 25K to 100K, for deliberative planning (i.e.,  $\rho_{mdp}$ ) between 1.6 million and 2.8 million. The planning time for reactive planning  $\rho_{det}$  is considered negligible i.e., less than a second. The planning time for deliberative planning  $\rho_{mdp}$  varies between 35-45 seconds.

### III. MATERIALS FOR THE TEAM OF UAVS

This section contains the concrete parameters and specifications for the team of UAVs.

*Q: How to calculate the probabilities of the team being destroyed and detecting a target?*

The team configuration has an effect on the probability of being destroyed by a threat and the probability of detecting a target, which is important when deciding how to adapt. A threat can destroy the team only if both are in the same segment. However, a threat has range  $r_T$ , and its effectiveness is inversely proportional to the altitude of the team, denoted by  $\mathcal{A}$ . In addition, the formation of the team affects the probability of it being destroyed. The team can be in two different formations: loose ( $\phi = 0$ ), and tight ( $\phi = 1$ ). The latter reduces the probability of being destroyed by a factor of  $\psi$  [2]. When the team uses ( $E = 1$ ) electronic countermeasures (ECM), the probability of being destroyed is reduced by a factor of  $\alpha$ . Taking altitude, formation, and the use of ECM into account, the probability of the team being destroyed,  $d$  is given by (1).

$$d = \frac{\max(0, r_T - \mathcal{A})}{r_T} \left( (1 - \phi) + \frac{\phi}{\psi} \right) \left( (1 - E) + \frac{E}{\alpha} \right). \quad (1)$$

The probability of detecting a target with the downward-looking sensor, given that the target is in the segment being traversed by the UAVs, is inversely proportional to the altitude of the team [3]. Furthermore, flying in tight formation reduces the detection probability due to sensor occlusion or overlap, and the use of ECM also affects target detection, reducing the probability of detection by a factor of  $\beta$ . The probability  $g$  of detecting a target is given by (2).

$$g = \frac{\max(0, r_S - \mathcal{A})}{r_S} \left( (1 - \phi) + \frac{\phi}{\sigma} \right) \left( (1 - E) + \frac{E}{\beta} \right), \quad (2)$$

where  $r_S$  is the range of the sensor (i.e., at an altitude of  $r_S$  or higher, it is not possible to detect targets), and  $\sigma$  is the factor by which the detection probability is reduced (due to flying in tight formation).

*Q: What Are the Exact Parameter Values for the Team of UAVs?*

We set the time-related parameters as follows: the systems evaluates the need for adaptation at each minute, and determines an action, if adaptation is needed. Some actions take time before their effect is observed – time to `IncAlt2/DecAlt2` and `IncAlt/DecAlt` is equal to 1 minute, which is also the duration that the team takes to cross a segment. For the team,

the horizon for reactive planning (i.e.,  $\rho_{mdps}$ ) and deliberative planning (i.e.,  $\rho_{mdpl}$ ) is 2 and 5, respectively.

We fixed the mission length for DARTSim at 40 segments. Total number of targets and threats are 20 and 10, respectively that are placed randomly depending on the random seed. Total number of altitude levels is 4, threat ( $r_T$ ) and target ( $r_S$ ) range is 3 and 4 respectively, the tight configuration reduces the probability of being destroyed (i.e.,  $\psi$ ) by factor 1.5. When using ECM the probability of being destroyed and target detection is reduced by a factor of 0.15, and 0.3, respectively. The threshold for the Manhattan distance is 1.0, which was decided after trying values 0.25, 0.5, 1.0, and 1.5. 1.0 provided the best performance for deliberative planning. The reward for surviving a segment is ( $\mu =$ ) 0.2 and for detecting a target is ( $\lambda =$ ) 1.

In hybrid planning modes, to identify an appropriate *Manhattan* distance to find the current state in an MDP policy, we evaluated the performance of deliberative mode with different distances as shown in Table II. We focused on the deliberative mode because, when using hybrid planning, Manhattan distance is used by deliberative planning. We finalized on Manhattan distance as 1.0 since deliberative mode detects maximum targets (i.e., 493) without being destroyed more compared to other distances.

Manhattan Distance	Targets	Destroyed
0.25	315	29
0.5	229	29
1.0	493	29
1.5	229	49

Table II

PERFORMANCE OF DELIBERATIVE MODE ON 70 MISSIONS FOR DIFFERENT MANHATTAN DISTANCE.

When looking up the current state in a plan (i.e., Line 6 in Algorithm 1 in the main paper), DARTSim needs to deal with the possibility of not finding any matches; in such cases, the plan fails and needs to be recomputed. As mentioned earlier, planning is done based on probability for segments having a target and a threat; this probability is calculated by sampling the observations by target and threat sensors. However, when the team reaches a segment, the probability value can change due to additional data collected during the mission. To find the closest matching state corresponding to the current state, we use two criteria: (1) all state variables (except the target and threat probabilities in the current segment) have the same values, and (2) the *Manhattan* distance between the pairs of target and threat probabilities is less than a predefined threshold. If no state meeting both criteria is found in a plan, the matching fails. If several states meet both criteria, the one with the smallest the distance is picked.

We conducted the experiments on a Ubuntu 14.04 virtual machine having 8.5 GB RAM and 4 processors at 2.9 GHz. The state space for the short-horizon MDP planning (i.e.,  $\rho_{mdps}$ ) varies between 75K to 250K, and for the long-horizon MDP planning  $\rho_{mdpl}$  between 3 million to 5 million. The planning time for reactive planning  $\rho_{mdps}$  is considered negligible i.e., less than a second. The planning time for deliberative planning  $\rho_{mdpl}$  varies between 40 and 60 seconds.

#### IV. MATERIALS FOR DATA ANALYSIS

This section contains auxiliary findings about HP from data analysis of both case studies.

##### *Q: What is the Dependency Between the Performance of Deliberative Planning and Hybrid Planning*

As discussed in the results of the main paper, as the performance of deliberative mode improves, the performance of the HP modes improves as well. This finding is further illustrated by Figure 1 for the cloud-based system and the UAV team; x-axis represents traces/missions sorted in ascending order in terms of aggregate utility (y-axis) accrued by deliberative mode. Intuitively, if a particular reactive approach (e.g.,  $\rho_{det}$  or  $\rho_{mdps}$ ) is not effective, the hybrid planner can choose another reactive approach (e.g.,  $\rho_{wait}$ ) in  $\mathcal{F}$ ; therefore, performance of HP is not tightly linked to a reactive approach.

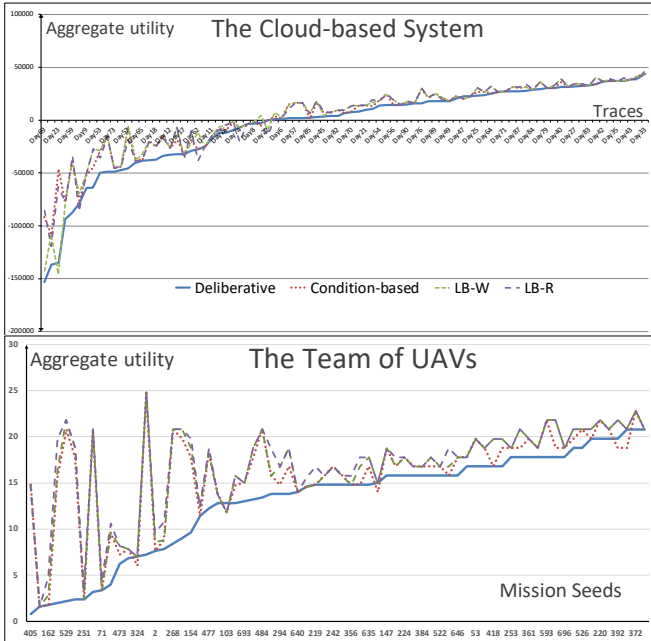


Figure 1. Performance of the hybrid planning modes improves with the performance of deliberative planning mode.

##### *Q: What is the Dependency Between the Performance of Reactive Planning and Hybrid Planning*

As stated in the results of the main paper, we discovered that the performance of reactive planning affects (with a positive correlation) the performance of hybrid planning. This dependency is moderated by the performance of the reactive planner, relative to other reactive planners. The better the reactive planner, the more influence it has on the hybrid planner.

We discovered this dependency by fitting a regression model to the utility of a hybrid planner  $U_{hp}$ , using the deliberative utility  $U_d$  and reactive utility  $U_r$  as independent variables. The fitting is done over all the traces/seeds in both case studies. Furthermore, to represent the moderating effect, these utilities were weighed with the following ratios:

- $U_d/U_{nw}$  for the deliberative utility, where  $U_{nw}$  is the utility of the NW-HP planner on that day. In this case,  $U_d$  is a proxy for the utility of the wait planner, which is invoked instead of reactive planner for  $U_{nw}$ . Thus, this ratio represents how much the reactive planning is better than waiting planning.
- $U_{nw}/U_d$  for the reactive utility, thus amplifying it on days when it is preferable to the wait planning, and reducing it on days when wait planning is preferable.

Thus, we arrive at the following regression model:

$$U_{hp} = a \cdot \frac{U_d}{U_{nw}} \cdot U_d + b \cdot \frac{U_{nw}}{U_d} \cdot U_r + c,$$

where  $a$ ,  $b$ , and  $c$  are regression coefficients determined by fitting the above function to the utility data. In the second case study we fit this exact function, and in the first case study we had to adjust the utility numbers such that none are negative or zero (otherwise the meaning of ratios is lost). Thus, we performed the following operation for the utility  $U$  of each type of planning:

$$U := U + \min(U) + 1.$$

In each fit model across both case studies the coefficients  $a$  and  $b$  were found positive. We tested the hypothesis that  $a$  and  $b$  are not zero with a t-test, yielding a highly significant result ( $p < 0.01$ ) that indeed the dependency exists.

## V. PRISM PLANNING SPECIFICATIONS

*Q: What are the Exact Planning Specifications for the Cloud-based System?*

Listing 1 and Listing 2 provide PRISM planning specifications for non-wait reactive (i.e., deterministic) and deliberative (i.e., MDP) planning (respectively) for a planning. These specifications have some constants (e.g.,  $MAX\_ARRIVALA\_CAPACITY$ ,  $penalty$ ) that remain same for all the planning problems (i.e., specifications). However, there are variables (e.g.,  $ini\_servers\_A$ , and  $ini\_traffic\_A$  that depend on the current state (i.e., initial state of the planning problem) of the system. These specifications also include modeling of the environment for this particular problem.

*Deterministic Planning Specification:* The listing below is the PRISM specification for deterministic planning  $\rho_{det}$ , which ignores uncertainty in the request arrival rate by assuming it to be constant at the current value. In the specification, inter-arrival time (the inverse of average request arrival rate) between two consecutive requests is used for environment modeling. For instance, in the statement “formula stateValue = 0.0163126” the request arrival rate will be 61 (= 1/0.0163126) requests per minute.

```

1  mdp
2
3  const double addServer_LATENCY = 120;
4  const int HORIZON = 5;
5  const double PERIOD = 60;
6  const int DIMMER_LEVELS = 3;
7  const int ini_dimmer = 1;
8  const int MAX_SERVERS_A = 1;
9  const int MAX_SERVERS_B = 1;
10 const int MAX_SERVERS_C = 1;
11 const int ini_servers_A = 1;
12 const int ini_servers_B = 0;
13 const int ini_servers_C = 0;
14 const int ini_addServerA_state = 0;
15 const int ini_addServerB_state = 0;
16 const int ini_addServerC_state = 0;
17 const double SERVERA_COST = 1;
18 const double SERVERB_COST = 0.7;
19 const double SERVERC_COST = 0.5;
20 const double MAX_ARRIVALA_CAPACITY = 200;
21 const double MAX_ARRIVALA_CAPACITY_LOW = 400;
22 const double MAX_ARRIVALB_CAPACITY = 140;
23 const double MAX_ARRIVALB_CAPACITY_LOW = 280;
24 const double MAX_ARRIVALC_CAPACITY = 100;
25 const double MAX_ARRIVALC_CAPACITY_LOW = 200;
26 const double penalty = -0.25;
27 const int ini_traffic_A = 4;
28 const int ini_traffic_B = 0;
29 const int ini_traffic_C = 0;
30 const double interArrivalScaleFactorForDecision = 1; // 1 has no effect
31
32 // The request arrival rate remains constant at the current value
33 formula stateValue = 0.0163126;
34
35
36 module clk
37   time : [0..HORIZON + 1] init 0;
38   readyToTick : bool init true;
39   [tick] readyToTick & time < HORIZON + 1 -> 1 : (time' = time + 1) &
   (readyToTick' = false);
40   [tack] !readyToTick -> 1 : (readyToTick' = true);
41 endmodule
42
43 label "final" = time = HORIZON + 1;
44 formula sys_go = readyToTick;
45
46 module controller
47   active_servers_A : [0..MAX_SERVERS_A] init ini_servers_A;
48   active_servers_B : [0..MAX_SERVERS_B] init ini_servers_B;
49   active_servers_C : [0..MAX_SERVERS_C] init ini_servers_C;
50

```

```

51   dimmer : [1..DIMMER_LEVELS] init ini_dimmer;
52
53   traffic_A : [0..4] init ini_traffic_A;
54   traffic_B : [0..4] init ini_traffic_B;
55   traffic_C : [0..4] init ini_traffic_C;
56
57   [addServerA_complete] active_servers_A < MAX_SERVERS_A -> 1 :
   (active_servers_A' = active_servers_A + 1);
58   [addServerB_complete] active_servers_B < MAX_SERVERS_B -> 1 :
   (active_servers_B' = active_servers_B + 1);
59   [addServerC_complete] active_servers_C < MAX_SERVERS_C -> 1 :
   (active_servers_C' = active_servers_C + 1);
60
61   [removeServerA_start] active_servers_A > 0 -> 1 : (active_servers_A' =
   active_servers_A - 1);
62   [removeServerB_start] active_servers_B > 0 -> 1 : (active_servers_B' =
   active_servers_B - 1);
63   [removeServerC_start] active_servers_C > 0 -> 1 : (active_servers_C' =
   active_servers_C - 1);
64
65   [increaseDimmer_start] dimmer < DIMMER_LEVELS -> 1 : (dimmer' =
   dimmer + 1);
66   [decreaseDimmer_start] dimmer > 1 -> 1 : (dimmer' = dimmer - 1);
67
68   //A-B-C
69   //Possible values 0-25-50-75-100
70
71   // 100-0-0
72   [divert_100_0_0] active_servers_A > 0
73   -> 1 : (traffic_A' = 4) & (traffic_B' = 0) & (traffic_C' = 0);
74   // 75-25-0
75   [divert_75_25_0] active_servers_A > 0 & active_servers_B > 0
76   -> 1 : (traffic_A' = 3) & (traffic_B' = 1) & (traffic_C' = 0);
77   // 75-0-25
78   [divert_75_0_25] active_servers_A > 0 & active_servers_C > 0
79   -> 1 : (traffic_A' = 3) & (traffic_B' = 0) & (traffic_C' = 1);
80   // 50-50-0
81   [divert_50_50_0] active_servers_A > 0 & active_servers_B > 0
82   -> 1 : (traffic_A' = 2) & (traffic_B' = 2) & (traffic_C' = 0);
83   // 50-0-50
84   [divert_50_0_50] active_servers_A > 0 & active_servers_C > 0
85   -> 1 : (traffic_A' = 2) & (traffic_B' = 0) & (traffic_C' = 2);
86   // 50-25-25
87   [divert_50_25_25] active_servers_A > 0 & active_servers_B > 0 &
   active_servers_C > 0
88   -> 1 : (traffic_A' = 2) & (traffic_B' = 1) & (traffic_C' = 1);
89   // 25-75-0
90   [divert_25_75_0] active_servers_A > 0 & active_servers_B > 0
91   -> 1 : (traffic_A' = 1) & (traffic_B' = 3) & (traffic_C' = 0);
92   // 25-0-75
93   [divert_25_0_75] active_servers_A > 0 & active_servers_C > 0
94   -> 1 : (traffic_A' = 1) & (traffic_B' = 0) & (traffic_C' = 3);
95   // 25-50-25
96   [divert_25_50_25] active_servers_A > 0 & active_servers_B > 0 &
   active_servers_C > 0
97   -> 1 : (traffic_A' = 1) & (traffic_B' = 2) & (traffic_C' = 1);
98   // 25-25-50
99   [divert_25_25_50] active_servers_A > 0 & active_servers_B > 0 &
   active_servers_C > 0
100  -> 1 : (traffic_A' = 1) & (traffic_B' = 1) & (traffic_C' = 2);
101   // 0-100-0
102   [divert_0_100_0] active_servers_B > 0
103   -> 1 : (traffic_A' = 0) & (traffic_B' = 4) & (traffic_C' = 0);
104   // 0-0-100
105   [divert_0_0_100] active_servers_C > 0
106   -> 1 : (traffic_A' = 0) & (traffic_B' = 0) & (traffic_C' = 4);
107   // 0-75-25
108   [divert_0_75_25] active_servers_B > 0 & active_servers_C > 0
109   -> 1 : (traffic_A' = 0) & (traffic_B' = 3) & (traffic_C' = 1);
110   // 0-25-75
111   [divert_0_25_75] active_servers_B > 0 & active_servers_C > 0
112   -> 1 : (traffic_A' = 0) & (traffic_B' = 1) & (traffic_C' = 3);
113   // 0-50-50
114   [divert_0_50_50] active_servers_B > 0 & active_servers_C > 0
115   -> 1 : (traffic_A' = 0) & (traffic_B' = 2) & (traffic_C' = 2);
116 endmodule
117
118
119 formula addServerA_applicable = active_servers_A < MAX_SERVERS_A
   & !removeServer_used
   & addServerB_state = 0 & addServerC_state = 0;
120 formula addServerB_applicable = active_servers_B < MAX_SERVERS_B
   & !removeServer_used

```

```

122         & addServerA_state = 0 & addServerC_state = 0;
123 formula addServerC_applicable = active_servers_C < MAX_SERVERS_C
    & !removeServer_used
124         & addServerA_state = 0 & addServerB_state = 0;
125
126 formula removeServerA_applicable = active_servers_A > 0 &
    addServerA_state = 0 & active_servers > 1
127         & addServerB_state = 0 & addServerC_state = 0;
128 formula removeServerB_applicable = active_servers_B > 0 &
    addServerB_state = 0 & active_servers > 1
129         & addServerA_state = 0 & addServerC_state = 0;
130 formula removeServerC_applicable = active_servers_C > 0 &
    addServerC_state = 0 & active_servers > 1
131         & addServerA_state = 0 & addServerB_state = 0;
132
133 formula increaseDimmer_compatible = !decreaseDimmer_used;
134 formula decreaseDimmer_compatible = !increaseDimmer_used;
135 formula increase_dimmer_applicable = dimmer < DIMMER_LEVELS &
    increaseDimmer_compatible;
136 formula decrease_dimmer_applicable = dimmer > 1 &
    decreaseDimmer_compatible;
137
138 const int addServer_LATENCY_PERIODS = ceil(addServer_LATENCY /
    PERIOD);
139
140 // This remove server constraints that only one server could be removed in
    one monitoring cycle.
141 module removeServer
142     removeServer_go : bool init true;
143     removeServer_used : bool init false;
144
145     [removeServerA_start] sys_go & removeServer_go
146         & removeServerA_applicable // applicability conditions
147         -> (removeServer_go' = false) & (removeServer_used' = true);
148
149     [removeServerB_start] sys_go & removeServer_go
150         & removeServerB_applicable // applicability conditions
151         -> (removeServer_go' = false) & (removeServer_used' = true);
152
153     [removeServerC_start] sys_go & removeServer_go
154         & removeServerC_applicable // applicability conditions
155         -> (removeServer_go' = false) & (removeServer_used' = true);
156
157     // Case when remove server tactic is applicable but not used
158     [pass_remove_server] sys_go & removeServer_go // can go
159         -> (removeServer_go' = false);
160
161     [tick] !removeServer_go -> 1 : (removeServer_go' = true) &
        (removeServer_used' = false);
162 endmodule
163
164 module addServer
165     addServerA_state : [0..addServer_LATENCY_PERIODS] init
        ini_addServerA_state;
166     addServerB_state : [0..addServer_LATENCY_PERIODS] init
        ini_addServerB_state;
167     addServerC_state : [0..addServer_LATENCY_PERIODS] init
        ini_addServerC_state;
168
169     addServer_go : bool init true;
170
171     // tactic applicable, start it
172     [addServerA_start] sys_go & addServer_go // can go
173         & addServerA_state = 0 // tactic has not been started
174         & addServerA_applicable
175         -> (addServerA_state' = 1) & (addServer_go' = false);
176
177     // tactic applicable, start it
178     [addServerB_start] sys_go & addServer_go // can go
179         & addServerB_state = 0 // tactic has not been started
180         & addServerB_applicable
181         -> (addServerB_state' = 1) & (addServer_go' = false);
182
183     // tactic applicable, start it
184     [addServerC_start] sys_go & addServer_go // can go
185         & addServerC_state = 0 // tactic has not been started
186         & addServerC_applicable
187         -> (addServerC_state' = 1) & (addServer_go' = false);
188
189     // tactic applicable, but don't use it
190     [pass_add] sys_go & addServer_go // can go
191         & addServerA_state = 0 // tactic has not been started
192         & addServerB_state = 0 & addServerC_state = 0
193         // & addServerA_applicable
194         -> (addServer_go' = false);
195
196     // progress of the tactic
197     [progressA] sys_go & addServer_go
198         & addServerA_state > 0 & addServerA_state <
        addServer_LATENCY_PERIODS
199         -> 1 : (addServerA_state' = addServerA_state + 1) &
        (addServer_go' = false);
200
201     [progressB] sys_go & addServer_go
202         & addServerB_state > 0 & addServerB_state <
        addServer_LATENCY_PERIODS
203         -> 1 : (addServerB_state' = addServerB_state + 1) &
        (addServer_go' = false);
204
205     [progressC] sys_go & addServer_go
206         & addServerC_state > 0 & addServerC_state <
        addServer_LATENCY_PERIODS
207         -> 1 : (addServerC_state' = addServerC_state + 1) &
        (addServer_go' = false);
208
209     // completion of the tactic
210     [addServerA_complete] sys_go & addServer_go
211         & addServerA_state = addServer_LATENCY_PERIODS //
        completed
212         -> 1 : (addServerA_state' = 0) & (addServer_go' = true); // so that
        it can start again at this time if needed
213
214     [addServerB_complete] sys_go & addServer_go
215         & addServerB_state = addServer_LATENCY_PERIODS //
        completed
216         -> 1 : (addServerB_state' = 0) & (addServer_go' = true); // so that
        it can start again at this time if needed
217
218     [addServerC_complete] sys_go & addServer_go
219         & addServerC_state = addServer_LATENCY_PERIODS //
        completed
220         -> 1 : (addServerC_state' = 0) & (addServer_go' = true); // so that
        it can start again at this time if needed
221
222     [tick] !addServer_go -> 1 : (addServer_go' = true);
223 endmodule
224
225 // Make sure that divert traffic is executed at the end i.e.after adding or
    removing the servers.
226 formula divert_traffic_applicable = divert_go & !addServer_go &
    !removeServer_go & !increaseDimmer_go & !decreaseDimmer_go;
227
228 module divert_traffic
229     divert_go : bool init true;
230
231     //A-B-C
232     //Possible values 0-25-50-75-100
233
234     // 100-0-0
235     [divert_100_0_0] sys_go & divert_traffic_applicable
236         -> 1 : (divert_go' = false);
237     // 75-25-0
238     [divert_75_25_0] sys_go & divert_traffic_applicable
239         -> 1 : (divert_go' = false);
240     // 75-0-25
241     [divert_75_0_25] sys_go & divert_traffic_applicable
242         -> 1 : (divert_go' = false);
243     // 50-50-0
244     [divert_50_50_0] sys_go & divert_traffic_applicable
245         -> 1 : (divert_go' = false);
246     // 50-0-50
247     [divert_50_0_50] sys_go & divert_traffic_applicable
248         -> 1 : (divert_go' = false);
249     // 50-25-25
250     [divert_50_25_25] sys_go & divert_traffic_applicable
251         -> 1 : (divert_go' = false);
252     // 25-75-0
253     [divert_25_75_0] sys_go & divert_traffic_applicable
254         -> 1 : (divert_go' = false);
255     // 25-0-75
256     [divert_25_0_75] sys_go & divert_traffic_applicable
257         -> 1 : (divert_go' = false);
258     // 25-50-25
259     [divert_25_50_25] sys_go & divert_traffic_applicable
260         -> 1 : (divert_go' = false);
261     // 25-25-50

```

```

262 [divert_25_25_50] sys_go & divert_traffic_applicable
263   -> 1 : (divert_go' = false);
264 // 0-100-0
265 [divert_0_100_0] sys_go & divert_traffic_applicable
266   -> 1 : (divert_go' = false);
267 // 0-0-100
268 [divert_0_0_100] sys_go & divert_traffic_applicable
269   -> 1 : (divert_go' = false);
270 // 0-75-25
271 [divert_0_75_25] sys_go & divert_traffic_applicable
272   -> 1 : (divert_go' = false);
273 // 0-25-75
274 [divert_0_25_75] sys_go & divert_traffic_applicable
275   -> 1 : (divert_go' = false);
276 // 0-50-50
277 [divert_0_50_50] sys_go & divert_traffic_applicable
278   -> 1 : (divert_go' = false);
279
280 [tick] !divert_go -> 1 : (divert_go' = true);
281 endmodule
282
283 module increaseDimmer
284   increaseDimmer_go : bool init true;
285   increaseDimmer_used : bool init false;
286
287   [increaseDimmer_start] sys_go & increaseDimmer_go
288     & increase_dimmer_applicable // applicability conditions
289     -> (increaseDimmer_go' = false) & (increaseDimmer_used' = true);
290
291   // tactic applicable but not used
292   [pass_inc_dimmer] sys_go & increaseDimmer_go // can go
293     -> (increaseDimmer_go' = false);
294
295   [tick] !increaseDimmer_go -> 1 : (increaseDimmer_go' = true) &
296     (increaseDimmer_used' = false);
297 endmodule
298 // tactic
299 module decreaseDimmer
300   decreaseDimmer_go : bool init true;
301   decreaseDimmer_used : bool init false;
302
303   [decreaseDimmer_start] sys_go & decreaseDimmer_go
304     & decrease_dimmer_applicable // applicability conditions
305     -> (decreaseDimmer_go' = false) & (decreaseDimmer_used' =
306       true);
307
308   // tactic applicable but not used
309   [pass_dec_dimmer] sys_go & decreaseDimmer_go // can go
310     -> (decreaseDimmer_go' = false);
311
312   [tick] !decreaseDimmer_go -> 1 : (decreaseDimmer_go' = true) &
313     (decreaseDimmer_used' = false);
314 endmodule
315 //*****
316 // Queuing network with each server having queueing model of M/G/1/PS
317 //*****
318 formula dimmerFactor = (dimmer - 1) / (DIMMER_LEVELS - 1);
319 formula interarrivalMean = stateValue * interArrivalScaleFactorForDecision;
320
321 formula Pa = (traffic_A * 25)/100;
322 formula Pb = (traffic_B * 25)/100;
323 formula Pc = (traffic_C * 25)/100;
324
325 formula loaded_servers = (Pa != 0 ? 1 : 0) + (Pb != 0 ? 1 : 0) + (Pc != 0 ? 1 :
326   0);
327
328 formula service_rate_A = dimmerFactor *
329   (MAX_ARRIVALA_CAPACITY_LOW)
330   + (1 - dimmerFactor) * (MAX_ARRIVALA_CAPACITY);
331 formula service_rate_B = dimmerFactor *
332   (MAX_ARRIVALB_CAPACITY_LOW)
333   + (1 - dimmerFactor) * (MAX_ARRIVALB_CAPACITY);
334 formula service_rate_C = dimmerFactor *
335   (MAX_ARRIVALC_CAPACITY_LOW)
336   + (1 - dimmerFactor) * (MAX_ARRIVALC_CAPACITY);
337
338 formula rhoA = Pa/(service_rate_A*interarrivalMean);
339 formula rhoB = Pb/(service_rate_B*interarrivalMean);
340 formula rhoC = Pc/(service_rate_C*interarrivalMean);
341 formula overloaded = (rhoA >= 1 | rhoB >= 1 | rhoC >= 1);
342
343 formula rt_A = 1/(service_rate_A - (throughput*Pa));
344 formula rt_B = 1/(service_rate_B - (throughput*Pb));
345 formula rt_C = 1/(service_rate_C - (throughput*Pc));
346
347 // Response time to clients utility function
348 const double RT_THRESHOLD = 1.0;
349
350 formula expected_wait_time = (Pa*rt_A + Pb*rt_B + Pc*rt_C);
351 formula rt = (interarrivalMean = 0 ? 0 : (overloaded ? RT_THRESHOLD + 2
352   : expected_wait_time));
353
354 const double NORMAL_A_REVENUE = (SERVERA_COST /
355   MAX_ARRIVALA_CAPACITY) * 10;
356 const double DIMMER_A_REVENUE = (SERVERA_COST /
357   MAX_ARRIVALA_CAPACITY_LOW) * 3 / 2;
358 const double NORMAL_B_REVENUE = (SERVERB_COST /
359   MAX_ARRIVALB_CAPACITY) * 10;
360 const double DIMMER_B_REVENUE = (SERVERB_COST /
361   MAX_ARRIVALB_CAPACITY_LOW) * 3 / 2;
362 const double NORMAL_C_REVENUE = (SERVERC_COST /
363   MAX_ARRIVALC_CAPACITY) * 10;
364 const double DIMMER_C_REVENUE = (SERVERC_COST /
365   MAX_ARRIVALC_CAPACITY_LOW) * 3 / 2;
366
367 const double DIMMER_REVENUE = DIMMER_A_REVENUE +
368   DIMMER_B_REVENUE + DIMMER_C_REVENUE;
369 const double NORMAL_REVENUE = NORMAL_A_REVENUE +
370   NORMAL_B_REVENUE + NORMAL_C_REVENUE;
371
372 formula serverA_cost = ((addServerA_state > 0 ? 1 : 0) +
373   active_servers_A) * SERVERA_COST;
374 formula serverB_cost = ((addServerB_state > 0 ? 1 : 0) +
375   active_servers_B) * SERVERB_COST;
376 formula serverC_cost = ((addServerC_state > 0 ? 1 : 0) +
377   active_servers_C) * SERVERC_COST;
378 formula cost = serverA_cost + serverB_cost + serverC_cost;
379 formula throughput = 1/interarrivalMean;
380
381 formula basicUtilityA = throughput * Pa * (dimmerFactor *
382   DIMMER_A_REVENUE + (1 - dimmerFactor) *
383   NORMAL_A_REVENUE);
384 formula basicUtilityB = throughput * Pb * (dimmerFactor *
385   DIMMER_B_REVENUE + (1 - dimmerFactor) *
386   NORMAL_B_REVENUE);
387 formula basicUtilityC = throughput * Pc * (dimmerFactor *
388   DIMMER_C_REVENUE + (1 - dimmerFactor) *
389   NORMAL_C_REVENUE);
390
391 formula basicUtility = basicUtilityA + basicUtilityB + basicUtilityC;
392 formula active_servers = active_servers_A + active_servers_B +
393   active_servers_C;
394 formula poweredServers = (addServerA_state > 0 ? 1 : 0) +
395   (addServerB_state > 0 ? 1 : 0) + (addServerC_state > 0 ? 1 : 0)
396   + active_servers;
397 formula MAX_SERVERS = MAX_SERVERS_A + MAX_SERVERS_B +
398   MAX_SERVERS_C;
399
400 formula MAX_SERVER_COST = MAX_SERVERS_A * SERVERA_COST
401   + MAX_SERVERS_B * SERVERB_COST
402   + MAX_SERVERS_C * SERVERC_COST;
403
404 formula netPenalty = stateValue > 0 ? penalty / stateValue : 0;
405
406 formula uTotal = (overloaded & (poweredServers < MAX_SERVERS |
407   dimmer < DIMMER_LEVELS | active_servers != loaded_servers))
408   ? -(1000) // avoid unstable solutions
409   : (((rt > RT_THRESHOLD | rt <= 0) ? netPenalty :
410     basicUtility) - cost);
411
412 rewards "util"
413 // 100000000.0 is added to avoid a negative value during calculation;
414 // negative utility is not supported by PRISM.
415 [tick] true : 100000000.0 + (PERIOD)*(uTotal);
416 endrewards

```

Listing 1. PRISM specification for deterministic planning

*MDP Planning Specification:* In contrast to deterministic planning (i.e.,  $\rho_{det}$ ), MDP planning (i.e.,  $\rho_{mdp}$ ) considers predicted (uncertain) values of request arrival rate. For  $\rho_{mdp}$ , we create an environment model using future values of inter-arrival time (the inverse of average request arrival rate) between two consecutive requests. When deliberative planning (i.e.,  $\rho_{mdp}$ ) is triggered, a time-series predictor feeds predicted values as an environment model formulating an MDP, mapping each possible interarrival rate to an outcome of a probabilistic action taken by the environment. The specification has environment modeled as an MDP for the planning horizon (i.e., 5) for MDP planning.

```

1  mdp
2
3  const double addServer_LATENCY = 120;
4  const int HORIZON = 5;
5  const double PERIOD = 60;
6  const int DIMMER_LEVELS = 3;
7  const int ini_dimmer = 1;
8  const int MAX_SERVERS_A = 1;
9  const int MAX_SERVERS_B = 1;
10 const int MAX_SERVERS_C = 1;
11 const int ini_servers_A = 1;
12 const int ini_servers_B = 0;
13 const int ini_servers_C = 0;
14 const int ini_addServerA_state = 0;
15 const int ini_addServerB_state = 0;
16 const int ini_addServerC_state = 0;
17 const double SERVERA_COST = 1;
18 const double SERVERB_COST = 0.7;
19 const double SERVERC_COST = 0.5;
20 const double MAX_ARRIVALA_CAPACITY = 200;
21 const double MAX_ARRIVALA_CAPACITY_LOW = 400;
22 const double MAX_ARRIVALB_CAPACITY = 140;
23 const double MAX_ARRIVALB_CAPACITY_LOW = 280;
24 const double MAX_ARRIVALC_CAPACITY = 100;
25 const double MAX_ARRIVALC_CAPACITY_LOW = 200;
26 const double penalty = -0.25;
27 const int ini_traffic_A = 4;
28 const int ini_traffic_B = 0;
29 const int ini_traffic_C = 0;
30 const double interArrivalScaleFactorForDecision = 1; // 1 has no effect
31
32 \\ Model of the environment as an MDP. Values from the time series
   predictor
33 \\ have been used to get the interarrival time.
34 module environment
35 s : [0..201] init 0;
36 [tick] s = 0 ->
37   0.185 : (s' = 1)
38   + 0.63 : (s' = 2)
39   + 0.185 : (s' = 3);
40 [tick] s = 3 ->
41   0.185 : (s' = 4)
42   + 0.63 : (s' = 5)
43   + 0.185 : (s' = 6);
44 [tick] s = 6 ->
45   0.185 : (s' = 7)
46   + 0.63 : (s' = 8)
47   + 0.185 : (s' = 9);
48 [tick] s = 9 ->
49   0.185 : (s' = 10)
50   + 0.63 : (s' = 11)
51   + 0.185 : (s' = 12);
52 [tick] s = 12 ->
53   1 : (s' = 13);
54 [tick] s = 11 ->
55   1 : (s' = 14);
56 [tick] s = 10 ->
57   1 : (s' = 15);
58 [tick] s = 8 ->
59   0.185 : (s' = 16)
60   + 0.63 : (s' = 17)
61   + 0.185 : (s' = 18);
62 [tick] s = 18 ->
63   1 : (s' = 19);
64 [tick] s = 17 ->
65   1 : (s' = 20);
66 [tick] s = 16 ->
67   1 : (s' = 21);
68 [tick] s = 7 ->
69   0.185 : (s' = 22)
70   + 0.63 : (s' = 23)
71   + 0.185 : (s' = 24);
72 [tick] s = 24 ->
73   1 : (s' = 25);
74 [tick] s = 23 ->
75   1 : (s' = 26);
76 [tick] s = 22 ->
77   1 : (s' = 27);
78 [tick] s = 5 ->
79   0.185 : (s' = 28)
80   + 0.63 : (s' = 29)
81   + 0.185 : (s' = 30);
82 [tick] s = 30 ->
83   0.185 : (s' = 31)
84   + 0.63 : (s' = 32)
85   + 0.185 : (s' = 33);
86 [tick] s = 33 ->
87   1 : (s' = 34);
88 [tick] s = 32 ->
89   1 : (s' = 35);
90 [tick] s = 31 ->
91   1 : (s' = 36);
92 [tick] s = 29 ->
93   0.185 : (s' = 37)
94   + 0.63 : (s' = 38)
95   + 0.185 : (s' = 39);
96 [tick] s = 39 ->
97   1 : (s' = 40);
98 [tick] s = 38 ->
99   1 : (s' = 41);
100 [tick] s = 37 ->
101   1 : (s' = 42);
102 [tick] s = 28 ->
103   0.185 : (s' = 43)
104   + 0.63 : (s' = 44)
105   + 0.185 : (s' = 45);
106 [tick] s = 45 ->
107   1 : (s' = 46);
108 [tick] s = 44 ->
109   1 : (s' = 47);
110 [tick] s = 43 ->
111   1 : (s' = 48);
112 [tick] s = 4 ->
113   0.185 : (s' = 49)
114   + 0.63 : (s' = 50)
115   + 0.185 : (s' = 51);
116 [tick] s = 51 ->
117   0.185 : (s' = 52)
118   + 0.63 : (s' = 53)
119   + 0.185 : (s' = 54);
120 [tick] s = 54 ->
121   1 : (s' = 55);
122 [tick] s = 53 ->
123   1 : (s' = 56);
124 [tick] s = 52 ->
125   1 : (s' = 57);
126 [tick] s = 50 ->
127   0.185 : (s' = 58)
128   + 0.63 : (s' = 59)
129   + 0.185 : (s' = 60);
130 [tick] s = 60 ->
131   1 : (s' = 61);
132 [tick] s = 59 ->
133   1 : (s' = 62);
134 [tick] s = 58 ->
135   1 : (s' = 63);
136 [tick] s = 49 ->
137   0.185 : (s' = 64)
138   + 0.63 : (s' = 65)
139   + 0.185 : (s' = 66);
140 [tick] s = 66 ->
141   1 : (s' = 67);
142 [tick] s = 65 ->
143   1 : (s' = 68);
144 [tick] s = 64 ->
145   1 : (s' = 69);
146 [tick] s = 2 ->
147   0.185 : (s' = 70)
148   + 0.63 : (s' = 71)

```



```

149 + 0.185 : (s' = 72);
150 [tick] s = 72 ->
151 0.185 : (s' = 73)
152 + 0.63 : (s' = 74)
153 + 0.185 : (s' = 75);
154 [tick] s = 75 ->
155 0.185 : (s' = 76)
156 + 0.63 : (s' = 77)
157 + 0.185 : (s' = 78);
158 [tick] s = 78 ->
159 1 : (s' = 79);
160 [tick] s = 77 ->
161 1 : (s' = 80);
162 [tick] s = 76 ->
163 1 : (s' = 81);
164 [tick] s = 74 ->
165 0.185 : (s' = 82)
166 + 0.63 : (s' = 83)
167 + 0.185 : (s' = 84);
168 [tick] s = 84 ->
169 1 : (s' = 85);
170 [tick] s = 83 ->
171 1 : (s' = 86);
172 [tick] s = 82 ->
173 1 : (s' = 87);
174 [tick] s = 73 ->
175 0.185 : (s' = 88)
176 + 0.63 : (s' = 89)
177 + 0.185 : (s' = 90);
178 [tick] s = 90 ->
179 1 : (s' = 91);
180 [tick] s = 89 ->
181 1 : (s' = 92);
182 [tick] s = 88 ->
183 1 : (s' = 93);
184 [tick] s = 71 ->
185 0.185 : (s' = 94)
186 + 0.63 : (s' = 95)
187 + 0.185 : (s' = 96);
188 [tick] s = 96 ->
189 0.185 : (s' = 97)
190 + 0.63 : (s' = 98)
191 + 0.185 : (s' = 99);
192 [tick] s = 99 ->
193 1 : (s' = 100);
194 [tick] s = 98 ->
195 1 : (s' = 101);
196 [tick] s = 97 ->
197 1 : (s' = 102);
198 [tick] s = 95 ->
199 0.185 : (s' = 103)
200 + 0.63 : (s' = 104)
201 + 0.185 : (s' = 105);
202 [tick] s = 105 ->
203 1 : (s' = 106);
204 [tick] s = 104 ->
205 1 : (s' = 107);
206 [tick] s = 103 ->
207 1 : (s' = 108);
208 [tick] s = 94 ->
209 0.185 : (s' = 109)
210 + 0.63 : (s' = 110)
211 + 0.185 : (s' = 111);
212 [tick] s = 111 ->
213 1 : (s' = 112);
214 [tick] s = 110 ->
215 1 : (s' = 113);
216 [tick] s = 109 ->
217 1 : (s' = 114);
218 [tick] s = 70 ->
219 0.185 : (s' = 115)
220 + 0.63 : (s' = 116)
221 + 0.185 : (s' = 117);
222 [tick] s = 117 ->
223 0.185 : (s' = 118)
224 + 0.63 : (s' = 119)
225 + 0.185 : (s' = 120);
226 [tick] s = 120 ->
227 1 : (s' = 121);
228 [tick] s = 119 ->
229 1 : (s' = 122);
230 [tick] s = 118 ->
231 1 : (s' = 123);

```

```

232 [tick] s = 116 ->
233 0.185 : (s' = 124)
234 + 0.63 : (s' = 125)
235 + 0.185 : (s' = 126);
236 [tick] s = 126 ->
237 1 : (s' = 127);
238 [tick] s = 125 ->
239 1 : (s' = 128);
240 [tick] s = 124 ->
241 1 : (s' = 129);
242 [tick] s = 115 ->
243 0.185 : (s' = 130)
244 + 0.63 : (s' = 131)
245 + 0.185 : (s' = 132);
246 [tick] s = 132 ->
247 1 : (s' = 133);
248 [tick] s = 131 ->
249 1 : (s' = 134);
250 [tick] s = 130 ->
251 1 : (s' = 135);
252 [tick] s = 1 ->
253 0.185 : (s' = 136)
254 + 0.63 : (s' = 137)
255 + 0.185 : (s' = 138);
256 [tick] s = 138 ->
257 0.185 : (s' = 139)
258 + 0.63 : (s' = 140)
259 + 0.185 : (s' = 141);
260 [tick] s = 141 ->
261 0.185 : (s' = 142)
262 + 0.63 : (s' = 143)
263 + 0.185 : (s' = 144);
264 [tick] s = 144 ->
265 1 : (s' = 145);
266 [tick] s = 143 ->
267 1 : (s' = 146);
268 [tick] s = 142 ->
269 1 : (s' = 147);
270 [tick] s = 140 ->
271 0.185 : (s' = 148)
272 + 0.63 : (s' = 149)
273 + 0.185 : (s' = 150);
274 [tick] s = 150 ->
275 1 : (s' = 151);
276 [tick] s = 149 ->
277 1 : (s' = 152);
278 [tick] s = 148 ->
279 1 : (s' = 153);
280 [tick] s = 139 ->
281 0.185 : (s' = 154)
282 + 0.63 : (s' = 155)
283 + 0.185 : (s' = 156);
284 [tick] s = 156 ->
285 1 : (s' = 157);
286 [tick] s = 155 ->
287 1 : (s' = 158);
288 [tick] s = 154 ->
289 1 : (s' = 159);
290 [tick] s = 137 ->
291 0.185 : (s' = 160)
292 + 0.63 : (s' = 161)
293 + 0.185 : (s' = 162);
294 [tick] s = 162 ->
295 0.185 : (s' = 163)
296 + 0.63 : (s' = 164)
297 + 0.185 : (s' = 165);
298 [tick] s = 165 ->
299 1 : (s' = 166);
300 [tick] s = 164 ->
301 1 : (s' = 167);
302 [tick] s = 163 ->
303 1 : (s' = 168);
304 [tick] s = 161 ->
305 0.185 : (s' = 169)
306 + 0.63 : (s' = 170)
307 + 0.185 : (s' = 171);
308 [tick] s = 171 ->
309 1 : (s' = 172);
310 [tick] s = 170 ->
311 1 : (s' = 173);
312 [tick] s = 169 ->
313 1 : (s' = 174);
314 [tick] s = 160 ->

```

```

315     0.185 : (s' = 175)
316     + 0.63 : (s' = 176)
317     + 0.185 : (s' = 177);
318 [tick] s = 177 ->
319     1 : (s' = 178);
320 [tick] s = 176 ->
321     1 : (s' = 179);
322 [tick] s = 175 ->
323     1 : (s' = 180);
324 [tick] s = 136 ->
325     0.185 : (s' = 181)
326     + 0.63 : (s' = 182)
327     + 0.185 : (s' = 183);
328 [tick] s = 183 ->
329     0.185 : (s' = 184)
330     + 0.63 : (s' = 185)
331     + 0.185 : (s' = 186);
332 [tick] s = 186 ->
333     1 : (s' = 187);
334 [tick] s = 185 ->
335     1 : (s' = 188);
336 [tick] s = 184 ->
337     1 : (s' = 189);
338 [tick] s = 182 ->
339     0.185 : (s' = 190)
340     + 0.63 : (s' = 191)
341     + 0.185 : (s' = 192);
342 [tick] s = 192 ->
343     1 : (s' = 193);
344 [tick] s = 191 ->
345     1 : (s' = 194);
346 [tick] s = 190 ->
347     1 : (s' = 195);
348 [tick] s = 181 ->
349     0.185 : (s' = 196)
350     + 0.63 : (s' = 197)
351     + 0.185 : (s' = 198);
352 [tick] s = 198 ->
353     1 : (s' = 199);
354 [tick] s = 197 ->
355     1 : (s' = 200);
356 [tick] s = 196 ->
357     1 : (s' = 201);
358 [tick] (s = 13 | s = 14 | s = 15 | s = 19 | s = 20 | s = 21 | s = 25 | s = 26 | s =
    27 | s = 34 | s = 35 | s = 36 | s = 40 | s = 41 | s = 42 | s = 46 | s = 47 |
    s = 48 | s = 55 | s = 56 | s = 57 | s = 61 | s = 62 | s = 63 | s = 67 | s =
    68 | s = 69 | s = 79 | s = 80 | s = 81 | s = 85 | s = 86 | s = 87 | s = 91 |
    s = 92 | s = 93 | s = 100 | s = 101 | s = 102 | s = 106 | s = 107 | s =
    108 | s = 112 | s = 113 | s = 114 | s = 121 | s = 122 | s = 123 | s = 127
    | s = 128 | s = 129 | s = 133 | s = 134 | s = 135 | s = 145 | s = 146 | s =
    147 | s = 151 | s = 152 | s = 153 | s = 157 | s = 158 | s = 159 | s = 166
    | s = 167 | s = 168 | s = 172 | s = 173 | s = 174 | s = 178 | s = 179 | s =
    180 | s = 187 | s = 188 | s = 189 | s = 193 | s = 194 | s = 195 | s = 199
    | s = 200 | s = 201) -> 1 : true;
359 endmodule
360 formula stateValue = (s = 0 ? 0.0176932 : 0) +
361     (s = 3 ? 0.0214149 : 0) +
362     (s = 6 ? 0.0270787 : 0) +
363     (s = 9 ? 0.0333291 : 0) +
364     (s = 12 ? 0.0407615 : 0) +
365     (s = 13 ? 0.0387364 : 0) +
366     (s = 11 ? 0.0321959 : 0) +
367     (s = 14 ? 0.0311986 : 0) +
368     (s = 10 ? 0.0236303 : 0) +
369     (s = 15 ? 0.0236609 : 0) +
370     (s = 8 ? 0.0266955 : 0) +
371     (s = 18 ? 0.0328878 : 0) +
372     (s = 19 ? 0.0318075 : 0) +
373     (s = 17 ? 0.0263583 : 0) +
374     (s = 20 ? 0.0260615 : 0) +
375     (s = 16 ? 0.0198287 : 0) +
376     (s = 21 ? 0.0203155 : 0) +
377     (s = 7 ? 0.0200618 : 0) +
378     (s = 24 ? 0.0256958 : 0) +
379     (s = 25 ? 0.0254786 : 0) +
380     (s = 23 ? 0.0205206 : 0) +
381     (s = 26 ? 0.0209244 : 0) +
382     (s = 22 ? 0.0153454 : 0) +
383     (s = 27 ? 0.0163702 : 0) +
384     (s = 5 ? 0.0217113 : 0) +
385     (s = 30 ? 0.0273884 : 0) +
386     (s = 33 ? 0.0336873 : 0) +
387     (s = 34 ? 0.0325111 : 0) +
388     (s = 32 ? 0.026968 : 0) +
389     (s = 35 ? 0.0265981 : 0) +
390     (s = 31 ? 0.0202487 : 0) +
391     (s = 36 ? 0.0206851 : 0) +
392     (s = 29 ? 0.0219722 : 0) +
393     (s = 39 ? 0.0276628 : 0) +
394     (s = 40 ? 0.0272095 : 0) +
395     (s = 38 ? 0.0222018 : 0) +
396     (s = 41 ? 0.0224038 : 0) +
397     (s = 37 ? 0.0167408 : 0) +
398     (s = 42 ? 0.0175982 : 0) +
399     (s = 28 ? 0.0165561 : 0) +
400     (s = 45 ? 0.0223769 : 0) +
401     (s = 46 ? 0.0225579 : 0) +
402     (s = 44 ? 0.0174356 : 0) +
403     (s = 47 ? 0.0182096 : 0) +
404     (s = 43 ? 0.0124943 : 0) +
405     (s = 48 ? 0.0138613 : 0) +
406     (s = 4 ? 0.016344 : 0) +
407     (s = 51 ? 0.0221892 : 0) +
408     (s = 54 ? 0.0278924 : 0) +
409     (s = 55 ? 0.0274116 : 0) +
410     (s = 53 ? 0.0223928 : 0) +
411     (s = 56 ? 0.0225719 : 0) +
412     (s = 52 ? 0.0168932 : 0) +
413     (s = 57 ? 0.0177322 : 0) +
414     (s = 50 ? 0.017249 : 0) +
415     (s = 60 ? 0.0230006 : 0) +
416     (s = 61 ? 0.0231068 : 0) +
417     (s = 59 ? 0.0180454 : 0) +
418     (s = 62 ? 0.0187462 : 0) +
419     (s = 58 ? 0.0130901 : 0) +
420     (s = 63 ? 0.0143856 : 0) +
421     (s = 49 ? 0.0123087 : 0) +
422     (s = 66 ? 0.0189053 : 0) +
423     (s = 67 ? 0.0195029 : 0) +
424     (s = 65 ? 0.0136979 : 0) +
425     (s = 68 ? 0.0149204 : 0) +
426     (s = 64 ? 0.00849053 : 0) +
427     (s = 69 ? 0.0103379 : 0) +
428     (s = 2 ? 0.0163126 : 0) +
429     (s = 72 ? 0.0221616 : 0) +
430     (s = 75 ? 0.0278631 : 0) +
431     (s = 78 ? 0.0342389 : 0) +
432     (s = 79 ? 0.0329965 : 0) +
433     (s = 77 ? 0.0273858 : 0) +
434     (s = 80 ? 0.0269657 : 0) +
435     (s = 76 ? 0.0205326 : 0) +
436     (s = 81 ? 0.0209349 : 0) +
437     (s = 74 ? 0.0223684 : 0) +
438     (s = 84 ? 0.0280829 : 0) +
439     (s = 85 ? 0.0275792 : 0) +
440     (s = 83 ? 0.0225505 : 0) +
441     (s = 86 ? 0.0227107 : 0) +
442     (s = 82 ? 0.0170181 : 0) +
443     (s = 87 ? 0.0178422 : 0) +
444     (s = 73 ? 0.0168738 : 0) +
445     (s = 90 ? 0.0226608 : 0) +
446     (s = 91 ? 0.0228078 : 0) +
447     (s = 89 ? 0.0177152 : 0) +
448     (s = 92 ? 0.0184556 : 0) +
449     (s = 88 ? 0.0127696 : 0) +
450     (s = 93 ? 0.0141035 : 0) +
451     (s = 71 ? 0.0172213 : 0) +
452     (s = 96 ? 0.0229754 : 0) +
453     (s = 99 ? 0.0287337 : 0) +
454     (s = 100 ? 0.0281519 : 0) +
455     (s = 98 ? 0.0230846 : 0) +
456     (s = 101 ? 0.0231807 : 0) +
457     (s = 97 ? 0.0174355 : 0) +
458     (s = 102 ? 0.0182095 : 0) +
459     (s = 95 ? 0.018021 : 0) +
460     (s = 105 ? 0.0237147 : 0) +
461     (s = 106 ? 0.0237352 : 0) +
462     (s = 104 ? 0.0187248 : 0) +
463     (s = 107 ? 0.019344 : 0) +
464     (s = 103 ? 0.0137348 : 0) +
465     (s = 108 ? 0.0149529 : 0) +
466     (s = 94 ? 0.0130667 : 0) +
467     (s = 111 ? 0.0194823 : 0) +
468     (s = 112 ? 0.0200107 : 0) +
469     (s = 110 ? 0.0143649 : 0) +
470     (s = 113 ? 0.0155074 : 0) +

```

```

471 (s = 109 ? 0.00924753 : 0) +
472 (s = 114 ? 0.0110041 : 0) +
473 (s = 70 ? 0.0122811 : 0) +
474 (s = 117 ? 0.0188846 : 0) +
475 (s = 120 ? 0.0245369 : 0) +
476 (s = 121 ? 0.0244587 : 0) +
477 (s = 119 ? 0.0194847 : 0) +
478 (s = 122 ? 0.0200128 : 0) +
479 (s = 118 ? 0.0144325 : 0) +
480 (s = 123 ? 0.0155668 : 0) +
481 (s = 116 ? 0.0136736 : 0) +
482 (s = 126 ? 0.0199571 : 0) +
483 (s = 127 ? 0.0204285 : 0) +
484 (s = 125 ? 0.014899 : 0) +
485 (s = 128 ? 0.0159774 : 0) +
486 (s = 124 ? 0.009841 : 0) +
487 (s = 129 ? 0.0115263 : 0) +
488 (s = 115 ? 0.00846263 : 0) +
489 (s = 132 ? 0.0162147 : 0) +
490 (s = 133 ? 0.0171352 : 0) +
491 (s = 131 ? 0.0103134 : 0) +
492 (s = 134 ? 0.011942 : 0) +
493 (s = 130 ? 0.00441199 : 0) +
494 (s = 135 ? 0.0067488 : 0) +
495 (s = 1 ? 0.0112104 : 0) +
496 (s = 138 ? 0.0180987 : 0) +
497 (s = 141 ? 0.0237876 : 0) +
498 (s = 144 ? 0.0296178 : 0) +
499 (s = 145 ? 0.0289299 : 0) +
500 (s = 143 ? 0.0237994 : 0) +
501 (s = 146 ? 0.0238097 : 0) +
502 (s = 142 ? 0.0179809 : 0) +
503 (s = 147 ? 0.0186894 : 0) +
504 (s = 140 ? 0.0187931 : 0) +
505 (s = 150 ? 0.0244486 : 0) +
506 (s = 151 ? 0.024381 : 0) +
507 (s = 149 ? 0.0194042 : 0) +
508 (s = 152 ? 0.0199419 : 0) +
509 (s = 148 ? 0.0143597 : 0) +
510 (s = 153 ? 0.0155028 : 0) +
511 (s = 139 ? 0.0137985 : 0) +
512 (s = 156 ? 0.0200562 : 0) +
513 (s = 157 ? 0.0205157 : 0) +
514 (s = 155 ? 0.015009 : 0) +
515 (s = 158 ? 0.0160741 : 0) +
516 (s = 154 ? 0.0099617 : 0) +
517 (s = 159 ? 0.0116325 : 0) +
518 (s = 137 ? 0.0127314 : 0) +
519 (s = 162 ? 0.0192249 : 0) +
520 (s = 165 ? 0.0248675 : 0) +
521 (s = 166 ? 0.0247496 : 0) +
522 (s = 164 ? 0.0197842 : 0) +
523 (s = 167 ? 0.0202763 : 0) +
524 (s = 163 ? 0.0147008 : 0) +
525 (s = 168 ? 0.015803 : 0) +
526 (s = 161 ? 0.0140698 : 0) +
527 (s = 171 ? 0.0202733 : 0) +
528 (s = 172 ? 0.0207067 : 0) +
529 (s = 170 ? 0.0152477 : 0) +
530 (s = 173 ? 0.0162842 : 0) +
531 (s = 169 ? 0.0102221 : 0) +
532 (s = 174 ? 0.0118617 : 0) +
533 (s = 160 ? 0.00891477 : 0) +
534 (s = 177 ? 0.016513 : 0) +
535 (s = 178 ? 0.0173977 : 0) +
536 (s = 176 ? 0.0107112 : 0) +
537 (s = 179 ? 0.0122921 : 0) +
538 (s = 175 ? 0.00490948 : 0) +
539 (s = 180 ? 0.00718659 : 0) +
540 (s = 136 ? 0.00736404 : 0) +
541 (s = 183 ? 0.0155067 : 0) +
542 (s = 186 ? 0.0214635 : 0) +
543 (s = 187 ? 0.0217541 : 0) +
544 (s = 185 ? 0.0165121 : 0) +
545 (s = 188 ? 0.0173969 : 0) +
546 (s = 184 ? 0.0115608 : 0) +
547 (s = 189 ? 0.0130397 : 0) +
548 (s = 182 ? 0.0093466 : 0) +
549 (s = 192 ? 0.0168019 : 0) +
550 (s = 193 ? 0.0176519 : 0) +
551 (s = 191 ? 0.0110913 : 0) +
552 (s = 194 ? 0.0126265 : 0) +
553 (s = 190 ? 0.0053806 : 0) +

```

```

554 (s = 195 ? 0.00760117 : 0) +
555 (s = 181 ? 0.0031865 : 0) +
556 (s = 198 ? 0.0129875 : 0) +
557 (s = 199 ? 0.0142952 : 0) +
558 (s = 197 ? 0.00567036 : 0) +
559 (s = 200 ? 0.00785617 : 0) +
560 (s = 196 ? 0 : 0) +
561 (s = 201 ? 0.00286625 : 0);
562
563
564 module clk
565 time : [0..HORIZON + 1] init 0;
566 readyToTick : bool init true;
567 [tick] readyToTick & time < HORIZON + 1 -> 1 : (time' = time + 1) &
    (readyToTick'=false);
568 [tack] !readyToTick -> 1 : (readyToTick'=true);
569 endmodule
570
571 label "final" = time = HORIZON + 1;
572 formula sys_go = readyToTick;
573
574 module controller
575 active_servers_A : [0..MAX_SERVERS_A] init ini_servers_A;
576 active_servers_B : [0..MAX_SERVERS_B] init ini_servers_B;
577 active_servers_C : [0..MAX_SERVERS_C] init ini_servers_C;
578
579 dimmer : [1..DIMMER_LEVELS] init ini_dimmer;
580
581 traffic_A : [0..4] init ini_traffic_A;
582 traffic_B : [0..4] init ini_traffic_B;
583 traffic_C : [0..4] init ini_traffic_C;
584
585 [addServerA_complete] active_servers_A < MAX_SERVERS_A -> 1 :
    (active_servers_A' = active_servers_A + 1);
586 [addServerB_complete] active_servers_B < MAX_SERVERS_B -> 1 :
    (active_servers_B' = active_servers_B + 1);
587 [addServerC_complete] active_servers_C < MAX_SERVERS_C -> 1 :
    (active_servers_C' = active_servers_C + 1);
588
589 [removeServerA_start] active_servers_A > 0 -> 1 : (active_servers_A' =
    active_servers_A - 1);
590 [removeServerB_start] active_servers_B > 0 -> 1 : (active_servers_B' =
    active_servers_B - 1);
591 [removeServerC_start] active_servers_C > 0 -> 1 : (active_servers_C' =
    active_servers_C - 1);
592
593 [increaseDimmer_start] dimmer < DIMMER_LEVELS -> 1 : (dimmer' =
    dimmer + 1);
594 [decreaseDimmer_start] dimmer > 1 -> 1 : (dimmer' = dimmer - 1);
595
596 //A-B-C
597 //Possible values 0-25-50-75-100
598
599 // 100-0-0
600 [divert_100_0_0] active_servers_A > 0
601 -> 1 : (traffic_A' = 4) & (traffic_B' = 0) & (traffic_C' = 0);
602 // 75-25-0
603 [divert_75_25_0] active_servers_A > 0 & active_servers_B > 0
604 -> 1 : (traffic_A' = 3) & (traffic_B' = 1) & (traffic_C' = 0);
605 // 75-0-25
606 [divert_75_0_25] active_servers_A > 0 & active_servers_C > 0
607 -> 1 : (traffic_A' = 3) & (traffic_B' = 0) & (traffic_C' = 1);
608 // 50-50-0
609 [divert_50_50_0] active_servers_A > 0 & active_servers_B > 0
610 -> 1 : (traffic_A' = 2) & (traffic_B' = 2) & (traffic_C' = 0);
611 // 50-0-50
612 [divert_50_0_50] active_servers_A > 0 & active_servers_C > 0
613 -> 1 : (traffic_A' = 2) & (traffic_B' = 0) & (traffic_C' = 2);
614 // 50-25-25
615 [divert_50_25_25] active_servers_A > 0 & active_servers_B > 0 &
    active_servers_C > 0
616 -> 1 : (traffic_A' = 2) & (traffic_B' = 1) & (traffic_C' = 1);
617 // 25-75-0
618 [divert_25_75_0] active_servers_A > 0 & active_servers_B > 0
619 -> 1 : (traffic_A' = 1) & (traffic_B' = 3) & (traffic_C' = 0);
620 // 25-0-75
621 [divert_25_0_75] active_servers_A > 0 & active_servers_C > 0
622 -> 1 : (traffic_A' = 1) & (traffic_B' = 0) & (traffic_C' = 3);
623 // 25-50-25
624 [divert_25_50_25] active_servers_A > 0 & active_servers_B > 0 &
    active_servers_C > 0
625 -> 1 : (traffic_A' = 1) & (traffic_B' = 2) & (traffic_C' = 1);
626 // 25-25-50

```

```

627 [divert_25_25_50] active_servers_A > 0 & active_servers_B > 0 &
        active_servers_C > 0
628     -> 1 : (traffic_A' = 1) & (traffic_B' = 1) & (traffic_C' = 2);
629 // 0-100-0
630 [divert_0_100_0] active_servers_B > 0
631     -> 1 : (traffic_A' = 0) & (traffic_B' = 4) & (traffic_C' = 0);
632 // 0-0-100
633 [divert_0_0_100] active_servers_C > 0
634     -> 1 : (traffic_A' = 0) & (traffic_B' = 0) & (traffic_C' = 4);
635 // 0-75-25
636 [divert_0_75_25] active_servers_B > 0 & active_servers_C > 0
637     -> 1 : (traffic_A' = 0) & (traffic_B' = 3) & (traffic_C' = 1);
638 // 0-25-75
639 [divert_0_25_75] active_servers_B > 0 & active_servers_C > 0
640     -> 1 : (traffic_A' = 0) & (traffic_B' = 1) & (traffic_C' = 3);
641 // 0-50-50
642 [divert_0_50_50] active_servers_B > 0 & active_servers_C > 0
643     -> 1 : (traffic_A' = 0) & (traffic_B' = 2) & (traffic_C' = 2);
644 endmodule
645
646 formula addServerA_applicable = active_servers_A < MAX_SERVERS_A
        & !removeServer_used
647     & addServerB_state = 0 & addServerC_state = 0;
648 formula addServerB_applicable = active_servers_B < MAX_SERVERS_B
        & !removeServer_used
649     & addServerA_state = 0 & addServerC_state = 0;
650 formula addServerC_applicable = active_servers_C < MAX_SERVERS_C
        & !removeServer_used
651     & addServerA_state = 0 & addServerB_state = 0;
652
653 formula removeServerA_applicable = active_servers_A > 0 &
        addServerA_state = 0 & active_servers > 1
654     & addServerB_state = 0 & addServerC_state = 0;
655 formula removeServerB_applicable = active_servers_B > 0 &
        addServerB_state = 0 & active_servers > 1
656     & addServerA_state = 0 & addServerC_state = 0;
657 formula removeServerC_applicable = active_servers_C > 0 &
        addServerC_state = 0 & active_servers > 1
658     & addServerA_state = 0 & addServerB_state = 0;
659
660 formula increaseDimmer_compatible = !decreaseDimmer_used;
661 formula decreaseDimmer_compatible = !increaseDimmer_used;
662 formula increase_dimmer_applicable = dimmer < DIMMER_LEVELS &
        increaseDimmer_compatible;
663 formula decrease_dimmer_applicable = dimmer > 1 &
        decreaseDimmer_compatible;
664
665 const int addServer_LATENCY_PERIODS = ceil(addServer_LATENCY /
        PERIOD);
666
667 // This remove server constraints that only one server could be removed in
668 // one monitoring cycle.
669 module removeServer
670     removeServer_go : bool init true;
671     removeServer_used : bool init false;
672
673     [removeServerA_start] sys_go & removeServer_go
674         & removeServerA_applicable // applicability conditions
675         -> (removeServer_go' = false) & (removeServer_used' = true);
676
677     [removeServerB_start] sys_go & removeServer_go
678         & removeServerB_applicable // applicability conditions
679         -> (removeServer_go' = false) & (removeServer_used' = true);
680
681     [removeServerC_start] sys_go & removeServer_go
682         & removeServerC_applicable // applicability conditions
683         -> (removeServer_go' = false) & (removeServer_used' = true);
684
685     // Case when remove server tactic is applicable but not used
686     [pass_remove_server] sys_go & removeServer_go // can go
687         -> (removeServer_go' = false);
688
689     [tick] !removeServer_go -> 1 : (removeServer_go' = true) &
        (removeServer_used' = false);
690 endmodule
691
692 module addServer
693     addServerA_state : [0..addServer_LATENCY_PERIODS] init
        ini_addServerA_state;
694     addServerB_state : [0..addServer_LATENCY_PERIODS] init
        ini_addServerB_state;
695     addServerC_state : [0..addServer_LATENCY_PERIODS] init
        ini_addServerC_state;
696
697     addServer_go : bool init true;
698
699     // tactic applicable, start it
700     [addServerA_start] sys_go & addServer_go // can go
701         & addServerA_state = 0 // tactic has not been started
702         & addServerA_applicable
703         -> (addServerA_state' = 1) & (addServer_go' = false);
704
705     // tactic applicable, start it
706     [addServerB_start] sys_go & addServer_go // can go
707         & addServerB_state = 0 // tactic has not been started
708         & addServerB_applicable
709         -> (addServerB_state' = 1) & (addServer_go' = false);
710
711     // tactic applicable, start it
712     [addServerC_start] sys_go & addServer_go // can go
713         & addServerC_state = 0 // tactic has not been started
714         & addServerC_applicable
715         -> (addServerC_state' = 1) & (addServer_go' = false);
716
717     // tactic applicable, but don't use it
718     [pass_add] sys_go & addServer_go // can go
719         & addServerA_state = 0 // tactic has not been started
720         & addServerB_state = 0 & addServerC_state = 0
721         // & addServerA_applicable
722         -> (addServer_go' = false);
723
724     // progress of the tactic
725     [progressA] sys_go & addServer_go
726         & addServerA_state > 0 & addServerA_state <
        addServer_LATENCY_PERIODS
727         -> 1 : (addServerA_state' = addServerA_state + 1) &
        (addServer_go' = false);
728
729     [progressB] sys_go & addServer_go
730         & addServerB_state > 0 & addServerB_state <
        addServer_LATENCY_PERIODS
731         -> 1 : (addServerB_state' = addServerB_state + 1) &
        (addServer_go' = false);
732
733     [progressC] sys_go & addServer_go
734         & addServerC_state > 0 & addServerC_state <
        addServer_LATENCY_PERIODS
735         -> 1 : (addServerC_state' = addServerC_state + 1) &
        (addServer_go' = false);
736
737     // completion of the tactic
738     [addServerA_complete] sys_go & addServer_go
739         & addServerA_state = addServer_LATENCY_PERIODS //
        completed
740         -> 1 : (addServerA_state' = 0) & (addServer_go' = true); // so that
        it can start again at this time if needed
741
742     [addServerB_complete] sys_go & addServer_go
743         & addServerB_state = addServer_LATENCY_PERIODS //
        completed
744         -> 1 : (addServerB_state' = 0) & (addServer_go' = true); // so that
        it can start again at this time if needed
745
746     [addServerC_complete] sys_go & addServer_go
747         & addServerC_state = addServer_LATENCY_PERIODS //
        completed
748         -> 1 : (addServerC_state' = 0) & (addServer_go' = true); // so that
        it can start again at this time if needed
749
750     [tick] !addServer_go -> 1 : (addServer_go' = true);
751 endmodule
752
753 // Make sure that divert traffic is executed at the end i.e.after adding or
754 // removing the servers.
755 formula divert_traffic_applicable = divert_go & !addServer_go &
        !removeServer_go & !increaseDimmer_go & !decreaseDimmer_go;
756
757 module divert_traffic
758     divert_go : bool init true;
759
760     //A-B-C
761     //Possible values 0-25-50-75-100
762
763     // 100-0-0
764     [divert_100_0_0] sys_go & divert_traffic_applicable

```

```

764     -> 1 : (divert_go' = false);
765 // 75-25-0
766 [divert_75_25_0] sys_go & divert_traffic_applicable
767     -> 1 : (divert_go' = false);
768 // 75-0-25
769 [divert_75_0_25] sys_go & divert_traffic_applicable
770     -> 1 : (divert_go' = false);
771 // 50-50-0
772 [divert_50_50_0] sys_go & divert_traffic_applicable
773     -> 1 : (divert_go' = false);
774 // 50-0-50
775 [divert_50_0_50] sys_go & divert_traffic_applicable
776     -> 1 : (divert_go' = false);
777 // 50-25-25
778 [divert_50_25_25] sys_go & divert_traffic_applicable
779     -> 1 : (divert_go' = false);
780 // 25-75-0
781 [divert_25_75_0] sys_go & divert_traffic_applicable
782     -> 1 : (divert_go' = false);
783 // 25-0-75
784 [divert_25_0_75] sys_go & divert_traffic_applicable
785     -> 1 : (divert_go' = false);
786 // 25-50-25
787 [divert_25_50_25] sys_go & divert_traffic_applicable
788     -> 1 : (divert_go' = false);
789 // 25-25-50
790 [divert_25_25_50] sys_go & divert_traffic_applicable
791     -> 1 : (divert_go' = false);
792 // 0-100-0
793 [divert_0_100_0] sys_go & divert_traffic_applicable
794     -> 1 : (divert_go' = false);
795 // 0-0-100
796 [divert_0_0_100] sys_go & divert_traffic_applicable
797     -> 1 : (divert_go' = false);
798 // 0-75-25
799 [divert_0_75_25] sys_go & divert_traffic_applicable
800     -> 1 : (divert_go' = false);
801 // 0-25-75
802 [divert_0_25_75] sys_go & divert_traffic_applicable
803     -> 1 : (divert_go' = false);
804 // 0-50-50
805 [divert_0_50_50] sys_go & divert_traffic_applicable
806     -> 1 : (divert_go' = false);
807
808 [tick] !divert_go -> 1 : (divert_go' = true);
809 endmodule
810
811 module increaseDimmer
812     increaseDimmer_go : bool init true;
813     increaseDimmer_used : bool init false;
814
815     [increaseDimmer_start] sys_go & increaseDimmer_go
816         & increase_dimmer_applicable // applicability conditions
817         -> (increaseDimmer_go' = false) & (increaseDimmer_used' = true);
818
819     // tactic applicable but not used
820     [pass_inc_dimmer] sys_go & increaseDimmer_go // can go
821         -> (increaseDimmer_go' = false);
822
823     [tick] !increaseDimmer_go -> 1 : (increaseDimmer_go' = true) &
824         (increaseDimmer_used' = false);
825 endmodule
826
827 // tactic
828 module decreaseDimmer
829     decreaseDimmer_go : bool init true;
830     decreaseDimmer_used : bool init false;
831
832     [decreaseDimmer_start] sys_go & decreaseDimmer_go
833         & decrease_dimmer_applicable // applicability conditions
834         -> (decreaseDimmer_go' = false) & (decreaseDimmer_used' =
835             true);
836
837     // tactic applicable but not used
838     [pass_dec_dimmer] sys_go & decreaseDimmer_go // can go
839         -> (decreaseDimmer_go' = false);
840
841     [tick] !decreaseDimmer_go -> 1 : (decreaseDimmer_go' = true) &
842         (decreaseDimmer_used' = false);
843 endmodule
844
845 //*****
846 // Queuing network with each server having queueing model of M/G/1/PS
847
848 //*****
849
850 formula dimmerFactor = (dimmer - 1) / (DIMMER_LEVELS - 1);
851 formula interarrivalMean = stateValue * interArrivalScaleFactorForDecision;
852
853 formula Pa = (traffic_A * 25)/100;
854 formula Pb = (traffic_B * 25)/100;
855 formula Pc = (traffic_C * 25)/100;
856
857 formula loaded_servers = (Pa != 0 ? 1 : 0) + (Pb != 0 ? 1 : 0) + (Pc != 0 ? 1 :
858     0);
859
860 formula service_rate_A = dimmerFactor *
861     (MAX_ARRIVALA_CAPACITY_LOW)
862     + (1 - dimmerFactor) * (MAX_ARRIVALA_CAPACITY);
863 formula service_rate_B = dimmerFactor *
864     (MAX_ARRIVALB_CAPACITY_LOW)
865     + (1 - dimmerFactor) * (MAX_ARRIVALB_CAPACITY);
866 formula service_rate_C = dimmerFactor *
867     (MAX_ARRIVALC_CAPACITY_LOW)
868     + (1 - dimmerFactor) * (MAX_ARRIVALC_CAPACITY);
869
870 formula rhoA = Pa/(service_rate_A*interarrivalMean);
871 formula rhoB = Pb/(service_rate_B*interarrivalMean);
872 formula rhoC = Pc/(service_rate_C*interarrivalMean);
873
874 formula overloaded = (rhoA >= 1 | rhoB >= 1 | rhoC >= 1);
875
876 formula rt_A = 1/(service_rate_A - (throughput*Pa));
877 formula rt_B = 1/(service_rate_B - (throughput*Pb));
878 formula rt_C = 1/(service_rate_C - (throughput*Pc));
879
880 // Response time to clients utility function
881 const double RT_THRESHOLD = 1.0;
882
883 formula expected_wait_time = (Pa*rt_A + Pb*rt_B + Pc*rt_C);
884 formula rt = (interarrivalMean = 0 ? 0 : (overloaded ? RT_THRESHOLD + 2
885     : expected_wait_time));
886
887 const double NORMAL_A_REVENUE = (SERVERA_COST /
888     MAX_ARRIVALA_CAPACITY) * 10;
889 const double DIMMER_A_REVENUE = (SERVERA_COST /
890     MAX_ARRIVALA_CAPACITY_LOW) * 3 / 2;
891 const double NORMAL_B_REVENUE = (SERVERB_COST /
892     MAX_ARRIVALB_CAPACITY) * 10;
893 const double DIMMER_B_REVENUE = (SERVERB_COST /
894     MAX_ARRIVALB_CAPACITY_LOW) * 3 / 2;
895 const double NORMAL_C_REVENUE = (SERVERC_COST /
896     MAX_ARRIVALC_CAPACITY) * 10;
897 const double DIMMER_C_REVENUE = (SERVERC_COST /
898     MAX_ARRIVALC_CAPACITY_LOW) * 3 / 2;
899
900 const double DIMMER_REVENUE = DIMMER_A_REVENUE +
901     DIMMER_B_REVENUE + DIMMER_C_REVENUE;
902 const double NORMAL_REVENUE = NORMAL_A_REVENUE +
903     NORMAL_B_REVENUE + NORMAL_C_REVENUE;
904
905 formula serverA_cost = ((addServerA_state > 0 ? 1 : 0) +
906     active_servers_A) * SERVERA_COST;
907 formula serverB_cost = ((addServerB_state > 0 ? 1 : 0) +
908     active_servers_B) * SERVERB_COST;
909 formula serverC_cost = ((addServerC_state > 0 ? 1 : 0) +
910     active_servers_C) * SERVERC_COST;
911 formula cost = serverA_cost + serverB_cost + serverC_cost;
912
913 formula throughput = 1/interarrivalMean;
914
915 formula basicUtilityA = throughput * Pa * (dimmerFactor *
916     DIMMER_A_REVENUE + (1 - dimmerFactor) *
917     NORMAL_A_REVENUE);
918 formula basicUtilityB = throughput * Pb * (dimmerFactor *
919     DIMMER_B_REVENUE + (1 - dimmerFactor) *
920     NORMAL_B_REVENUE);
921 formula basicUtilityC = throughput * Pc * (dimmerFactor *
922     DIMMER_C_REVENUE + (1 - dimmerFactor) *
923     NORMAL_C_REVENUE);
924
925 formula basicUtility = basicUtilityA + basicUtilityB + basicUtilityC;
926 formula active_servers = active_servers_A + active_servers_B +
927     active_servers_C;
928 formula poweredServers = (addServerA_state > 0 ? 1 : 0) +
929     (addServerB_state > 0 ? 1 : 0) + (addServerC_state > 0 ? 1 : 0)
930     + active_servers;
931 formula MAX_SERVERS = MAX_SERVERS_A + MAX_SERVERS_B +

```

```

903     MAX_SERVERS_C;
904 formula MAX_SERVER_COST = MAX_SERVERS_A * SERVERA_COST
905     + MAX_SERVERS_B * SERVERB_COST
906     + MAX_SERVERS_C * SERVERC_COST;
907
908 formula netPenalty = stateValue > 0 ? penalty / stateValue : 0;
909
910 formula uTotal = (overloaded & (poweredServers < MAX_SERVERS |
911     dimmer < DIMMER_LEVELS | active_servers != loaded_servers))
912     ? -(1000 // avoid unstable solutions)
913     : (((rt > RT_THRESHOLD | rt <= 0) ? netPenalty :
914         basicUtility) - cost);
915
916 rewards "util"
917     // 100000000.0 is added to avoid a negative value during calculation;
918     // negative utility is not supported by PRISM.
919     [tack] true : 100000000.0 + (PERIOD)*(uTotal);
920 endrewards

```

Listing 2. PRISM specification for MDP planning

*Q: What are the Exact Planning Specification for the Team of UAVs?*

Listing 3 and Listing 4 provide PRISM planning specifications for non-wait reactive (i.e., the short-horizon MDP with a subset of actions) and deliberative (i.e., the long-horizon MDP) planning (respectively) for a planning. These specifications have some constants (e.g., *threatRange*, *sensorRange*) that remain same for all the planning problems (i.e., specifications). However, there are variables (e.g., current altitude *ini\_a*, and formation *ini\_f* that depend on the current state (i.e., initial state of the planning problem) of the system. These specifications also include modeling of the environment for this particular problem.

*Short Horizon MDP Planning Specification:* Reactive planning  $\rho_{mdps}$  plans with a shorter horizon compared to deliberative planning  $\rho_{mdpl}$ . Moreover, while planning,  $\rho_{mdps}$  do not consider adaptation actions *IncAlt*, *DecAlt*, and *EcmOn*, and *EcmOff*.

```

1  mdp
2  const double PERIOD = 60;
3  const int HORIZON = 2; // Planning horizon for reactive planning
4  const double IncAlt_LATENCY = 60;
5  const double DecAlt_LATENCY = 60;
6  const int MAX_ALT_LEVEL = 3;
7  const double destructionFormationFactor = 1.5;
8  const double threatRange = 3;
9  const double detectionFormationFactor = 1.2;
10 const double sensorRange = 4;
11 const ini_a = 0;
12 const ini_c = 0;
13 const ini_f = 0;
14 const bool ECM_ENABLED = false; // ECM is not enabled for reactive
15 const bool ONE_LEVEL_ENABLED = false; // This is not enabled for
16 const bool TWO_LEVEL_ENABLED = true; // Two level increase/decrease
17 const int ini_IncAlt_state = 0;
18 const int ini_DecAlt_state = 0;
19 const int ini_IncAlt2_state = 0;
20 const int ini_DecAlt2_state = 0;
21 const double ecm_threat_prob = 0.15;
22 const double ecm_target_prob = 0.3;
23 const double survival_reward = 1;
24
25
26 //*****
27 // CLOCK
28 //*****
29 const int TO_TICK = 0;
30 const int TO_TICK2 = 1; // intermediate tick for constraint satisf. update
31 const int TO_TACK = 2;
32
33 label "final" = time = HORIZON & clockstep=TO_TICK;
34 formula sys_go = clockstep=TO_TICK;
35
36 module clk
37     time : [0..HORIZON] init 0;
38     clockstep : [0..2] init TO_TICK;
39
40     [tick] clockstep=TO_TICK & time < HORIZON -> 1 : (time'=time+1) &
41         (clockstep'=TO_TICK2);
42     [tick2] clockstep=TO_TICK2 -> 1 : (clockstep'=TO_TACK);
43     [tack] clockstep=TO_TACK -> 1 : (clockstep'=TO_TICK);
44 endmodule
45
46 module env
47     s : [0..45] init 0;
48     [tick] s = 0 ->
49         0.034225 : (s' = 1)
50         + 0.11655 : (s' = 2)
51         + 0.034225 : (s' = 3)
52         + 0.11655 : (s' = 4)
53         + 0.3969 : (s' = 5)
54         + 0.11655 : (s' = 6)

```

54 + 0.034225 : (s' = 7)  
55 + 0.11655 : (s' = 8)  
56 + 0.034225 : (s' = 9);  
57 [tick] s = 1 ->  
58 0.034225 : (s' = 10)  
59 + 0.11655 : (s' = 11)  
60 + 0.034225 : (s' = 12)  
61 + 0.11655 : (s' = 13)  
62 + 0.3969 : (s' = 14)  
63 + 0.11655 : (s' = 15)  
64 + 0.034225 : (s' = 16)  
65 + 0.11655 : (s' = 17)  
66 + 0.034225 : (s' = 18);  
67 [tick] s = 2 ->  
68 0.034225 : (s' = 10)  
69 + 0.11655 : (s' = 11)  
70 + 0.034225 : (s' = 12)  
71 + 0.11655 : (s' = 13)  
72 + 0.3969 : (s' = 14)  
73 + 0.11655 : (s' = 15)  
74 + 0.034225 : (s' = 16)  
75 + 0.11655 : (s' = 17)  
76 + 0.034225 : (s' = 18);  
77 [tick] s = 3 ->  
78 0.034225 : (s' = 10)  
79 + 0.11655 : (s' = 11)  
80 + 0.034225 : (s' = 12)  
81 + 0.11655 : (s' = 13)  
82 + 0.3969 : (s' = 14)  
83 + 0.11655 : (s' = 15)  
84 + 0.034225 : (s' = 16)  
85 + 0.11655 : (s' = 17)  
86 + 0.034225 : (s' = 18);  
87 [tick] s = 4 ->  
88 0.034225 : (s' = 10)  
89 + 0.11655 : (s' = 11)  
90 + 0.034225 : (s' = 12)  
91 + 0.11655 : (s' = 13)  
92 + 0.3969 : (s' = 14)  
93 + 0.11655 : (s' = 15)  
94 + 0.034225 : (s' = 16)  
95 + 0.11655 : (s' = 17)  
96 + 0.034225 : (s' = 18);  
97 [tick] s = 5 ->  
98 0.034225 : (s' = 10)  
99 + 0.11655 : (s' = 11)  
100 + 0.034225 : (s' = 12)  
101 + 0.11655 : (s' = 13)  
102 + 0.3969 : (s' = 14)  
103 + 0.11655 : (s' = 15)  
104 + 0.034225 : (s' = 16)  
105 + 0.11655 : (s' = 17)  
106 + 0.034225 : (s' = 18);  
107 [tick] s = 6 ->  
108 0.034225 : (s' = 10)  
109 + 0.11655 : (s' = 11)  
110 + 0.034225 : (s' = 12)  
111 + 0.11655 : (s' = 13)  
112 + 0.3969 : (s' = 14)  
113 + 0.11655 : (s' = 15)  
114 + 0.034225 : (s' = 16)  
115 + 0.11655 : (s' = 17)  
116 + 0.034225 : (s' = 18);  
117 [tick] s = 7 ->  
118 0.034225 : (s' = 10)  
119 + 0.11655 : (s' = 11)  
120 + 0.034225 : (s' = 12)  
121 + 0.11655 : (s' = 13)  
122 + 0.3969 : (s' = 14)  
123 + 0.11655 : (s' = 15)  
124 + 0.034225 : (s' = 16)  
125 + 0.11655 : (s' = 17)  
126 + 0.034225 : (s' = 18);  
127 [tick] s = 8 ->  
128 0.034225 : (s' = 10)  
129 + 0.11655 : (s' = 11)  
130 + 0.034225 : (s' = 12)  
131 + 0.11655 : (s' = 13)  
132 + 0.3969 : (s' = 14)  
133 + 0.11655 : (s' = 15)  
134 + 0.034225 : (s' = 16)  
135 + 0.11655 : (s' = 17)  
136 + 0.034225 : (s' = 18);

137 [tick] s = 9 ->  
138 0.034225 : (s' = 10)  
139 + 0.11655 : (s' = 11)  
140 + 0.034225 : (s' = 12)  
141 + 0.11655 : (s' = 13)  
142 + 0.3969 : (s' = 14)  
143 + 0.11655 : (s' = 15)  
144 + 0.034225 : (s' = 16)  
145 + 0.11655 : (s' = 17)  
146 + 0.034225 : (s' = 18);  
147 [tick] s = 10 ->  
148 0.034225 : (s' = 19)  
149 + 0.11655 : (s' = 20)  
150 + 0.034225 : (s' = 21)  
151 + 0.11655 : (s' = 22)  
152 + 0.3969 : (s' = 23)  
153 + 0.11655 : (s' = 24)  
154 + 0.034225 : (s' = 25)  
155 + 0.11655 : (s' = 26)  
156 + 0.034225 : (s' = 27);  
157 [tick] s = 11 ->  
158 0.034225 : (s' = 19)  
159 + 0.11655 : (s' = 20)  
160 + 0.034225 : (s' = 21)  
161 + 0.11655 : (s' = 22)  
162 + 0.3969 : (s' = 23)  
163 + 0.11655 : (s' = 24)  
164 + 0.034225 : (s' = 25)  
165 + 0.11655 : (s' = 26)  
166 + 0.034225 : (s' = 27);  
167 [tick] s = 12 ->  
168 0.034225 : (s' = 19)  
169 + 0.11655 : (s' = 20)  
170 + 0.034225 : (s' = 21)  
171 + 0.11655 : (s' = 22)  
172 + 0.3969 : (s' = 23)  
173 + 0.11655 : (s' = 24)  
174 + 0.034225 : (s' = 25)  
175 + 0.11655 : (s' = 26)  
176 + 0.034225 : (s' = 27);  
177 [tick] s = 13 ->  
178 0.034225 : (s' = 19)  
179 + 0.11655 : (s' = 20)  
180 + 0.034225 : (s' = 21)  
181 + 0.11655 : (s' = 22)  
182 + 0.3969 : (s' = 23)  
183 + 0.11655 : (s' = 24)  
184 + 0.034225 : (s' = 25)  
185 + 0.11655 : (s' = 26)  
186 + 0.034225 : (s' = 27);  
187 [tick] s = 14 ->  
188 0.034225 : (s' = 19)  
189 + 0.11655 : (s' = 20)  
190 + 0.034225 : (s' = 21)  
191 + 0.11655 : (s' = 22)  
192 + 0.3969 : (s' = 23)  
193 + 0.11655 : (s' = 24)  
194 + 0.034225 : (s' = 25)  
195 + 0.11655 : (s' = 26)  
196 + 0.034225 : (s' = 27);  
197 [tick] s = 15 ->  
198 0.034225 : (s' = 19)  
199 + 0.11655 : (s' = 20)  
200 + 0.034225 : (s' = 21)  
201 + 0.11655 : (s' = 22)  
202 + 0.3969 : (s' = 23)  
203 + 0.11655 : (s' = 24)  
204 + 0.034225 : (s' = 25)  
205 + 0.11655 : (s' = 26)  
206 + 0.034225 : (s' = 27);  
207 [tick] s = 16 ->  
208 0.034225 : (s' = 19)  
209 + 0.11655 : (s' = 20)  
210 + 0.034225 : (s' = 21)  
211 + 0.11655 : (s' = 22)  
212 + 0.3969 : (s' = 23)  
213 + 0.11655 : (s' = 24)  
214 + 0.034225 : (s' = 25)  
215 + 0.11655 : (s' = 26)  
216 + 0.034225 : (s' = 27);  
217 [tick] s = 17 ->  
218 0.034225 : (s' = 19)  
219 + 0.11655 : (s' = 20)

```

220 + 0.034225 : (s' = 21)
221 + 0.11655 : (s' = 22)
222 + 0.3969 : (s' = 23)
223 + 0.11655 : (s' = 24)
224 + 0.034225 : (s' = 25)
225 + 0.11655 : (s' = 26);
226 + 0.034225 : (s' = 27);
227 [tick] s = 18 ->
228 0.034225 : (s' = 19)
229 + 0.11655 : (s' = 20)
230 + 0.034225 : (s' = 21)
231 + 0.11655 : (s' = 22)
232 + 0.3969 : (s' = 23)
233 + 0.11655 : (s' = 24)
234 + 0.034225 : (s' = 25)
235 + 0.11655 : (s' = 26)
236 + 0.034225 : (s' = 27);
237 [tick] s = 19 ->
238 0.034225 : (s' = 28)
239 + 0.11655 : (s' = 29)
240 + 0.034225 : (s' = 30)
241 + 0.11655 : (s' = 31)
242 + 0.3969 : (s' = 32)
243 + 0.11655 : (s' = 33)
244 + 0.034225 : (s' = 34)
245 + 0.11655 : (s' = 35)
246 + 0.034225 : (s' = 36);
247 [tick] s = 20 ->
248 0.034225 : (s' = 28)
249 + 0.11655 : (s' = 29)
250 + 0.034225 : (s' = 30)
251 + 0.11655 : (s' = 31)
252 + 0.3969 : (s' = 32)
253 + 0.11655 : (s' = 33)
254 + 0.034225 : (s' = 34)
255 + 0.11655 : (s' = 35)
256 + 0.034225 : (s' = 36);
257 [tick] s = 21 ->
258 0.034225 : (s' = 28)
259 + 0.11655 : (s' = 29)
260 + 0.034225 : (s' = 30)
261 + 0.11655 : (s' = 31)
262 + 0.3969 : (s' = 32)
263 + 0.11655 : (s' = 33)
264 + 0.034225 : (s' = 34)
265 + 0.11655 : (s' = 35)
266 + 0.034225 : (s' = 36);
267 [tick] s = 22 ->
268 0.034225 : (s' = 28)
269 + 0.11655 : (s' = 29)
270 + 0.034225 : (s' = 30)
271 + 0.11655 : (s' = 31)
272 + 0.3969 : (s' = 32)
273 + 0.11655 : (s' = 33)
274 + 0.034225 : (s' = 34)
275 + 0.11655 : (s' = 35)
276 + 0.034225 : (s' = 36);
277 [tick] s = 23 ->
278 0.034225 : (s' = 28)
279 + 0.11655 : (s' = 29)
280 + 0.034225 : (s' = 30)
281 + 0.11655 : (s' = 31)
282 + 0.3969 : (s' = 32)
283 + 0.11655 : (s' = 33)
284 + 0.034225 : (s' = 34)
285 + 0.11655 : (s' = 35)
286 + 0.034225 : (s' = 36);
287 [tick] s = 24 ->
288 0.034225 : (s' = 28)
289 + 0.11655 : (s' = 29)
290 + 0.034225 : (s' = 30)
291 + 0.11655 : (s' = 31)
292 + 0.3969 : (s' = 32)
293 + 0.11655 : (s' = 33)
294 + 0.034225 : (s' = 34)
295 + 0.11655 : (s' = 35)
296 + 0.034225 : (s' = 36);
297 [tick] s = 25 ->
298 0.034225 : (s' = 28)
299 + 0.11655 : (s' = 29)
300 + 0.034225 : (s' = 30)
301 + 0.11655 : (s' = 31)
302 + 0.3969 : (s' = 32)
303 + 0.11655 : (s' = 33)
304 + 0.034225 : (s' = 34)
305 + 0.11655 : (s' = 35)
306 + 0.034225 : (s' = 36);
307 [tick] s = 26 ->
308 0.034225 : (s' = 28)
309 + 0.11655 : (s' = 29)
310 + 0.034225 : (s' = 30)
311 + 0.11655 : (s' = 31)
312 + 0.3969 : (s' = 32)
313 + 0.11655 : (s' = 33)
314 + 0.034225 : (s' = 34)
315 + 0.11655 : (s' = 35)
316 + 0.034225 : (s' = 36);
317 [tick] s = 27 ->
318 0.034225 : (s' = 28)
319 + 0.11655 : (s' = 29)
320 + 0.034225 : (s' = 30)
321 + 0.11655 : (s' = 31)
322 + 0.3969 : (s' = 32)
323 + 0.11655 : (s' = 33)
324 + 0.034225 : (s' = 34)
325 + 0.11655 : (s' = 35)
326 + 0.034225 : (s' = 36);
327 [tick] s = 28 ->
328 0.034225 : (s' = 37)
329 + 0.11655 : (s' = 38)
330 + 0.034225 : (s' = 39)
331 + 0.11655 : (s' = 40)
332 + 0.3969 : (s' = 41)
333 + 0.11655 : (s' = 42)
334 + 0.034225 : (s' = 43)
335 + 0.11655 : (s' = 44)
336 + 0.034225 : (s' = 45);
337 [tick] s = 29 ->
338 0.034225 : (s' = 37)
339 + 0.11655 : (s' = 38)
340 + 0.034225 : (s' = 39)
341 + 0.11655 : (s' = 40)
342 + 0.3969 : (s' = 41)
343 + 0.11655 : (s' = 42)
344 + 0.034225 : (s' = 43)
345 + 0.11655 : (s' = 44)
346 + 0.034225 : (s' = 45);
347 [tick] s = 30 ->
348 0.034225 : (s' = 37)
349 + 0.11655 : (s' = 38)
350 + 0.034225 : (s' = 39)
351 + 0.11655 : (s' = 40)
352 + 0.3969 : (s' = 41)
353 + 0.11655 : (s' = 42)
354 + 0.034225 : (s' = 43)
355 + 0.11655 : (s' = 44)
356 + 0.034225 : (s' = 45);
357 [tick] s = 31 ->
358 0.034225 : (s' = 37)
359 + 0.11655 : (s' = 38)
360 + 0.034225 : (s' = 39)
361 + 0.11655 : (s' = 40)
362 + 0.3969 : (s' = 41)
363 + 0.11655 : (s' = 42)
364 + 0.034225 : (s' = 43)
365 + 0.11655 : (s' = 44)
366 + 0.034225 : (s' = 45);
367 [tick] s = 32 ->
368 0.034225 : (s' = 37)
369 + 0.11655 : (s' = 38)
370 + 0.034225 : (s' = 39)
371 + 0.11655 : (s' = 40)
372 + 0.3969 : (s' = 41)
373 + 0.11655 : (s' = 42)
374 + 0.034225 : (s' = 43)
375 + 0.11655 : (s' = 44)
376 + 0.034225 : (s' = 45);
377 [tick] s = 33 ->
378 0.034225 : (s' = 37)
379 + 0.11655 : (s' = 38)
380 + 0.034225 : (s' = 39)
381 + 0.11655 : (s' = 40)
382 + 0.3969 : (s' = 41)
383 + 0.11655 : (s' = 42)
384 + 0.034225 : (s' = 43)
385 + 0.11655 : (s' = 44)

```



```

386     + 0.034225 : (s' = 45);
387 [tick] s = 34 ->
388     0.034225 : (s' = 37)
389     + 0.11655 : (s' = 38)
390     + 0.034225 : (s' = 39)
391     + 0.11655 : (s' = 40)
392     + 0.3969 : (s' = 41)
393     + 0.11655 : (s' = 42)
394     + 0.034225 : (s' = 43)
395     + 0.11655 : (s' = 44)
396     + 0.034225 : (s' = 45);
397 [tick] s = 35 ->
398     0.034225 : (s' = 37)
399     + 0.11655 : (s' = 38)
400     + 0.034225 : (s' = 39)
401     + 0.11655 : (s' = 40)
402     + 0.3969 : (s' = 41)
403     + 0.11655 : (s' = 42)
404     + 0.034225 : (s' = 43)
405     + 0.11655 : (s' = 44)
406     + 0.034225 : (s' = 45);
407 [tick] s = 36 ->
408     0.034225 : (s' = 37)
409     + 0.11655 : (s' = 38)
410     + 0.034225 : (s' = 39)
411     + 0.11655 : (s' = 40)
412     + 0.3969 : (s' = 41)
413     + 0.11655 : (s' = 42)
414     + 0.034225 : (s' = 43)
415     + 0.11655 : (s' = 44)
416     + 0.034225 : (s' = 45);
417 endmodule
418
419 // environment has 2 components. statevalue for threats and statevalue1 is
420 // for targets
421 formula stateValue = (s = 0 ? 0 : 0) +
422     (s = 1 ? 0.00605639 : 0) +
423     (s = 2 ? 0.00605639 : 0) +
424     (s = 3 ? 0.00605639 : 0) +
425     (s = 4 ? 0.0282836 : 0) +
426     (s = 5 ? 0.0282836 : 0) +
427     (s = 6 ? 0.0282836 : 0) +
428     (s = 7 ? 0.0778979 : 0) +
429     (s = 8 ? 0.0778979 : 0) +
430     (s = 9 ? 0.0778979 : 0) +
431     (s = 10 ? 0 : 0) +
432     (s = 11 ? 0 : 0) +
433     (s = 12 ? 0 : 0) +
434     (s = 13 ? 0 : 0) +
435     (s = 14 ? 0 : 0) +
436     (s = 15 ? 0 : 0) +
437     (s = 16 ? 0 : 0) +
438     (s = 17 ? 0 : 0) +
439     (s = 18 ? 0 : 0) +
440     (s = 19 ? 0.750751 : 0) +
441     (s = 20 ? 0.750751 : 0) +
442     (s = 21 ? 0.750751 : 0) +
443     (s = 22 ? 0.885424 : 0) +
444     (s = 23 ? 0.885424 : 0) +
445     (s = 24 ? 0.885424 : 0) +
446     (s = 25 ? 0.963485 : 0) +
447     (s = 26 ? 0.963485 : 0) +
448     (s = 27 ? 0.963485 : 0) +
449     (s = 28 ? 0.435626 : 0) +
450     (s = 29 ? 0.435626 : 0) +
451     (s = 30 ? 0.435626 : 0) +
452     (s = 31 ? 0.676196 : 0) +
453     (s = 32 ? 0.676196 : 0) +
454     (s = 33 ? 0.676196 : 0) +
455     (s = 34 ? 0.864925 : 0) +
456     (s = 35 ? 0.864925 : 0) +
457     (s = 36 ? 0.864925 : 0) +
458     (s = 37 ? 0.368403 : 0) +
459     (s = 38 ? 0.368403 : 0) +
460     (s = 39 ? 0.368403 : 0) +
461     (s = 40 ? 0.793701 : 0) +
462     (s = 41 ? 0.793701 : 0) +
463     (s = 42 ? 0.793701 : 0) +
464     (s = 43 ? 0.983048 : 0) +
465     (s = 44 ? 0.983048 : 0) +
466     (s = 45 ? 0.983048 : 0);
467 formula stateValue1 = (s = 0 ? 0 : 0) +

```

```

468     (s = 1 ? 0.830037 : 0) +
469     (s = 2 ? 0.904439 : 0) +
470     (s = 3 ? 0.954777 : 0) +
471     (s = 4 ? 0.830037 : 0) +
472     (s = 5 ? 0.904439 : 0) +
473     (s = 6 ? 0.954777 : 0) +
474     (s = 7 ? 0.830037 : 0) +
475     (s = 8 ? 0.904439 : 0) +
476     (s = 9 ? 0.954777 : 0) +
477     (s = 10 ? 0 : 0) +
478     (s = 11 ? 0 : 0) +
479     (s = 12 ? 0 : 0) +
480     (s = 13 ? 0 : 0) +
481     (s = 14 ? 0 : 0) +
482     (s = 15 ? 0 : 0) +
483     (s = 16 ? 0 : 0) +
484     (s = 17 ? 0 : 0) +
485     (s = 18 ? 0 : 0) +
486     (s = 19 ? 0.0156741 : 0) +
487     (s = 20 ? 0.0719057 : 0) +
488     (s = 21 ? 0.190204 : 0) +
489     (s = 22 ? 0.0156741 : 0) +
490     (s = 23 ? 0.0719057 : 0) +
491     (s = 24 ? 0.190204 : 0) +
492     (s = 25 ? 0.0156741 : 0) +
493     (s = 26 ? 0.0719057 : 0) +
494     (s = 27 ? 0.190204 : 0) +
495     (s = 28 ? 0 : 0) +
496     (s = 29 ? 0 : 0) +
497     (s = 30 ? 0 : 0) +
498     (s = 31 ? 0 : 0) +
499     (s = 32 ? 0 : 0) +
500     (s = 33 ? 0 : 0) +
501     (s = 34 ? 0 : 0) +
502     (s = 35 ? 0 : 0) +
503     (s = 36 ? 0 : 0) +
504     (s = 37 ? 0 : 0) +
505     (s = 38 ? 0 : 0) +
506     (s = 39 ? 0 : 0) +
507     (s = 40 ? 0 : 0) +
508     (s = 41 ? 0 : 0) +
509     (s = 42 ? 0 : 0) +
510     (s = 43 ? 0 : 0) +
511     (s = 44 ? 0 : 0) +
512     (s = 45 ? 0 : 0);
513
514
515 //*****
516 // SYSTEM
517 //*****
518
519 // Variable range and initialization
520 const a_MIN=0; const a_MAX=MAX_ALT_LEVEL; const a_INIT=init_a;
521 const f_MIN=0; const f_MAX=1; const f_INIT=init_f;
522 const c_MIN=0; const c_MAX=1; const c_INIT=init_c;
523
524 module sys
525     a : [a_MIN..a_MAX] init a_INIT;
526     f : [f_MIN..f_MAX] init f_INIT;
527     c : [c_MIN..c_MAX] init c_INIT;
528
529     [EcmOn_start] c=0 & ECM_ENABLED -> 1: (c'=c_EcmOn_impact);
530     [EcmOff_start] c=1 & ECM_ENABLED -> 1: (c'=c_EcmOff_impact);
531
532     [GoTight_start] f=0 -> 1: (a'=a_GoTight_impact)
533         & (f'=f_GoTight_impact);
534     [GoLoose_start] f=1 -> 1: (a'=a_GoLoose_impact)
535         & (f'=f_GoLoose_impact);
536
537     [IncAlt_complete] a < MAX_ALT_LEVEL & ONE_LEVEL_ENABLED ->
538         1: (a'=a_IncAlt_impact)
539         & (f'=f_IncAlt_impact);
540     [IncAlt2_complete] a < MAX_ALT_LEVEL-1 &
541         TWO_LEVEL_ENABLED -> 1: (a'=a_IncAlt2_impact);
542
543     [DecAlt_complete] a > 0 & ONE_LEVEL_ENABLED -> 1:
544         (a'=a_DecAlt_impact)
545         & (f'=f_DecAlt_impact);
546     [DecAlt2_complete] a > 1 & TWO_LEVEL_ENABLED -> 1:
547         (a'=a_DecAlt2_impact);
548
549 endmodule

```

```

547 formula c_EcmOn_impact = c + (1) >= c_MIN ? ( c+(1)<=c_MAX? c+(1) :
      c_MAX) : c_MIN;
548 formula c_EcmOff_impact = c + (-1) >= c_MIN ? ( c+(-1)<=c_MAX?
      c+(-1) : c_MAX) : c_MIN;
549 formula a_GoTight_impact = a + (0) >= a_MIN ? ( a+(0)<=a_MAX? a+(0) :
      a_MAX) : a_MIN;
550 formula f_GoTight_impact = f + (1) >= f_MIN ? ( f+(1)<=f_MAX? f+(1) :
      f_MAX) : f_MIN;
551 formula a_GoLoose_impact = a + (0) >= a_MIN ? ( a+(0)<=a_MAX? a+(0) :
      a_MAX) : a_MIN;
552 formula f_GoLoose_impact = f + (-1) >= f_MIN ? ( f+(-1)<=f_MAX? f+(-1)
      : f_MAX) : f_MIN;
553 formula a_IncAlt_impact = a + (1) >= a_MIN ? ( a+(1)<=a_MAX? a+(1) :
      a_MAX) : a_MIN;
554 formula f_IncAlt_impact = f + (0) >= f_MIN ? ( f+(0)<=f_MAX? f+(0) :
      f_MAX) : f_MIN;
555 formula a_DecAlt_impact = a + (-1) >= a_MIN ? ( a+(-1)<=a_MAX?
      a+(-1) : a_MAX) : a_MIN;
556 formula f_DecAlt_impact = f + (0) >= f_MIN ? ( f+(0)<=f_MAX? f+(0) :
      f_MAX) : f_MIN;
557 formula a_IncAlt2_impact = a + (2) >= a_MIN ? ( a+(2)<=a_MAX? a+(2) :
      a_MAX) : a_MIN;
558 formula a_DecAlt2_impact = a + (-2) >= a_MIN ? ( a+(-2)<=a_MAX?
      a+(-2) : a_MAX) : a_MIN;
559
560 // tactic concurrency rules
561 formula IncAlt_used = IncAlt_state != 0;
562 formula DecAlt_used = DecAlt_state != 0;
563 formula IncAlt2_used = IncAlt2_state != 0;
564 formula DecAlt2_used = DecAlt2_state != 0;
565
566 formula EcmOn_compatible = !EcmOn_used;
567 formula EcmOff_compatible = !EcmOff_used;
568 formula GoTight_compatible = !GoLoose_used;
569 formula GoLoose_compatible = !GoTight_used;
570 formula IncAlt_compatible = (!DecAlt_used) & (!IncAlt2_used) &
      (!DecAlt2_used);
571 formula DecAlt_compatible = (!IncAlt_used) & (!IncAlt2_used) &
      (!DecAlt2_used);
572 formula IncAlt2_compatible = (!DecAlt_used) & (!IncAlt_used) &
      (!DecAlt2_used);
573 formula DecAlt2_compatible = (!DecAlt_used) & (!IncAlt_used) &
      (!IncAlt2_used);
574
575
576 //*****
577 // TACTIC: EcmOn
578 //*****
579
580 // Applicability conditions
581 formula EcmOn_applicable = EcmOn_compatible & c=0;
582
583 module EcmOn
584   EcmOn_used : bool init false;
585   EcmOn_go : bool init true;
586
587   // Tactic applicable, start it
588   [EcmOn_start] sys_go & EcmOn_go & EcmOn_applicable &
      ECM_ENABLED -> (EcmOn_used'=true) & (EcmOn_go'=false);
589
590   // Tactic applicable, but do not start it
591   [EcmOn_pass] sys_go & EcmOn_go & EcmOn_applicable ->
      (EcmOn_go'=false);
592
593   // Pass if the tactic is not applicable
594   [EcmOn_invalid] sys_go & EcmOn_go & !EcmOn_applicable -> 1 :
      (EcmOn_go'=false);
595
596   [tick] !EcmOn_go -> 1: (EcmOn_go'=true) & (EcmOn_used'=false);
597 endmodule
598
599
600 //*****
601 // TACTIC: EcmOff
602 //*****
603
604 // Applicability conditions
605 formula EcmOff_applicable = EcmOff_compatible & c=1;
606
607 module EcmOff
608   EcmOff_used : bool init false;
609   EcmOff_go : bool init true;
610
611   // Tactic applicable, start it
612   [EcmOff_start] sys_go & EcmOff_go & EcmOff_applicable &
      ECM_ENABLED -> (EcmOff_used'=true) & (EcmOff_go'=false);
613
614   // Tactic applicable, but do not start it
615   [EcmOff_pass] sys_go & EcmOff_go & EcmOff_applicable ->
      (EcmOff_go'=false);
616
617   // Pass if the tactic is not applicable
618   [EcmOff_invalid] sys_go & EcmOff_go & !EcmOff_applicable -> 1 :
      (EcmOff_go'=false);
619
620   [tick] !EcmOff_go -> 1: (EcmOff_go'=true) & (EcmOff_used'=false);
621 endmodule
622
623
624 //*****
625 // TACTIC: GoTight
626 //*****
627
628 // Applicability conditions
629 formula GoTight_applicable = GoTight_compatible & f=0;
630
631 module GoTight
632   GoTight_used : bool init false;
633   GoTight_go : bool init true;
634
635   // Tactic applicable, start it
636   [GoTight_start] sys_go & GoTight_go & GoTight_applicable ->
      (GoTight_used'=true) & (GoTight_go'=false);
637
638   // Tactic applicable, but do not start it
639   [GoTight_pass] sys_go & GoTight_go & GoTight_applicable ->
      (GoTight_go'=false);
640
641   // Pass if the tactic is not applicable
642   [GoTight_invalid] sys_go & GoTight_go & !GoTight_applicable -> 1 :
      (GoTight_go'=false);
643
644   [tick] !GoTight_go -> 1: (GoTight_go'=true) & (GoTight_used'=false);
645 endmodule
646
647
648 //*****
649 // TACTIC: GoLoose
650 //*****
651
652 // Applicability conditions
653 formula GoLoose_applicable = GoLoose_compatible & f=1;
654
655 module GoLoose
656   GoLoose_used : bool init false;
657   GoLoose_go : bool init true;
658
659   // Tactic applicable, start it
660   [GoLoose_start] sys_go & GoLoose_go & GoLoose_applicable ->
      (GoLoose_used'=true) & (GoLoose_go'=false);
661
662   // Tactic applicable, but do not start it
663   [GoLoose_pass] sys_go & GoLoose_go & GoLoose_applicable ->
      (GoLoose_go'=false);
664
665   // Pass if the tactic is not applicable
666   [GoLoose_invalid] sys_go & GoLoose_go & !GoLoose_applicable -> 1 :
      (GoLoose_go'=false);
667
668   [tick] !GoLoose_go -> 1: (GoLoose_go'=true) &
      (GoLoose_used'=false);
669 endmodule
670
671
672 //*****
673 // TACTIC: IncAlt
674 //*****
675
676 const int IncAlt_LATENCY_PERIODS = ceil(IncAlt_LATENCY/PERIOD);
677
678 // Applicability conditions
679 formula IncAlt_applicable = IncAlt_compatible & a < MAX_ALT_LEVEL;
680
681 module IncAlt
682   IncAlt_state : [0..IncAlt_LATENCY_PERIODS] init ini_IncAlt_state;
683   IncAlt_go : bool init true;

```

```

684
685 // Tactic applicable, start it
686 [IncAlt_start] sys_go & IncAlt_go & IncAlt_state=0 & IncAlt_applicable &
        ONE_LEVEL_ENABLED ->
        (IncAlt_state'=IncAlt_LATENCY_PERIODS) & (IncAlt_go'=false);
687
688 // Tactic applicable, but do not start it
689 [IncAlt_pass] sys_go & IncAlt_go & IncAlt_state=0 & IncAlt_applicable
        -> (IncAlt_go'=false);
690
691 // Pass if the tactic is not applicable
692 [IncAlt_invalid] sys_go & IncAlt_go & IncAlt_state=0 & !IncAlt_applicable
        -> 1 : (IncAlt_go'=false);
693
694 // Progress of the tactic
695 [IncAlt_progress] sys_go & IncAlt_go & IncAlt_state > 1 -> 1:
        (IncAlt_state'=IncAlt_state-1) & (IncAlt_go'=false);
696
697 // Completion of the tactic
698 [IncAlt_complete] sys_go & IncAlt_go & IncAlt_state=1 -> 1:
        (IncAlt_state'=0) & (IncAlt_go'=true);
699
700 [tick] !IncAlt_go -> 1: (IncAlt_go'=true);
701 endmodule
702
703
704 //*****
705 // TACTIC: DecAlt
706 //*****
707
708 const int DecAlt_LATENCY_PERIODS = ceil(DecAlt_LATENCY/PERIOD);
709
710 // Applicability conditions
711 formula DecAlt_applicable = DecAlt_compatible & a > 0;
712
713 module DecAlt
714   DecAlt_state : [0..DecAlt_LATENCY_PERIODS] init ini_DecAlt_state;
715   DecAlt_go : bool init true;
716
717   // Tactic applicable, start it
718   [DecAlt_start] sys_go & DecAlt_go & DecAlt_state=0 &
        DecAlt_applicable & ONE_LEVEL_ENABLED ->
        (DecAlt_state'=DecAlt_LATENCY_PERIODS) &
        (DecAlt_go'=false);
719
720   // Tactic applicable, but do not start it
721   [DecAlt_pass] sys_go & DecAlt_go & DecAlt_state=0 &
        DecAlt_applicable -> (DecAlt_go'=false);
722
723   // Pass if the tactic is not applicable
724   [DecAlt_invalid] sys_go & DecAlt_go & DecAlt_state=0 &
        !DecAlt_applicable -> 1 : (DecAlt_go'=false);
725
726   // Progress of the tactic
727   [DecAlt_progress] sys_go & DecAlt_go & DecAlt_state > 1 -> 1:
        (DecAlt_state'=DecAlt_state-1) & (DecAlt_go'=false);
728
729   // Completion of the tactic
730   [DecAlt_complete] sys_go & DecAlt_go & DecAlt_state=1 -> 1:
        (DecAlt_state'=0) & (DecAlt_go'=true);
731
732   [tick] !DecAlt_go -> 1: (DecAlt_go'=true);
733 endmodule
734
735 //*****
736 // TACTIC: IncAlt2
737 //*****
738
739 // Applicability conditions
740 formula IncAlt2_applicable = IncAlt2_compatible & a <
        MAX_ALT_LEVEL-1;
741
742 module IncAlt2
743   IncAlt2_state : [0..IncAlt2_LATENCY_PERIODS] init ini_IncAlt2_state;
744   IncAlt2_go : bool init true;
745
746   // Tactic applicable, start it
747   [IncAlt2_start] sys_go & IncAlt2_go & IncAlt2_state=0 &
        IncAlt2_applicable & TWO_LEVEL_ENABLED ->
        (IncAlt2_state'=IncAlt2_LATENCY_PERIODS) &
        (IncAlt2_go'=false);
748
749   // Tactic applicable, but do not start it
750
751   [IncAlt2_pass] sys_go & IncAlt2_go & IncAlt2_state=0 &
        IncAlt2_applicable -> (IncAlt2_go'=false);
752
753   // Pass if the tactic is not applicable
754   [IncAlt2_invalid] sys_go & IncAlt2_go & IncAlt2_state=0 &
        !IncAlt2_applicable -> 1 : (IncAlt2_go'=false);
755
756   // Progress of the tactic
757   [IncAlt2_progress] sys_go & IncAlt2_go & IncAlt2_state > 1 -> 1:
        (IncAlt2_state'=IncAlt2_state-1) & (IncAlt2_go'=false);
758
759   // Completion of the tactic
760   [IncAlt2_complete] sys_go & IncAlt2_go & IncAlt2_state=1 -> 1:
        (IncAlt2_state'=0) & (IncAlt2_go'=true);
761
762   [tick] !IncAlt2_go -> 1: (IncAlt2_go'=true);
763 endmodule
764
765 //*****
766 // TACTIC: DecAlt2
767 //*****
768
769 const int DecAlt2_LATENCY_PERIODS = ceil(DecAlt2_LATENCY/PERIOD);
770
771 // Applicability conditions
772 formula DecAlt2_applicable = DecAlt2_compatible & a > 1;
773
774 module DecAlt2
775   DecAlt2_state : [0..DecAlt2_LATENCY_PERIODS] init
        ini_DecAlt2_state;
776   DecAlt2_go : bool init true;
777
778   // Tactic applicable, start it
779   [DecAlt2_start] sys_go & DecAlt2_go & DecAlt2_state=0 &
        DecAlt2_applicable & TWO_LEVEL_ENABLED ->
        (DecAlt2_state'=DecAlt2_LATENCY_PERIODS) &
        (DecAlt2_go'=false);
780
781   // Tactic applicable, but do not start it
782   [DecAlt2_pass] sys_go & DecAlt2_go & DecAlt2_state=0 &
        DecAlt2_applicable -> (DecAlt2_go'=false);
783
784   // Pass if the tactic is not applicable
785   [DecAlt2_invalid] sys_go & DecAlt2_go & DecAlt2_state=0 &
        !DecAlt2_applicable -> 1 : (DecAlt2_go'=false);
786
787   // Progress of the tactic
788   [DecAlt2_progress] sys_go & DecAlt2_go & DecAlt2_state > 1 -> 1:
        (DecAlt2_state'=DecAlt2_state-1) & (DecAlt2_go'=false);
789
790   // Completion of the tactic
791   [DecAlt2_complete] sys_go & DecAlt2_go & DecAlt2_state=1 -> 1:
        (DecAlt2_state'=0) & (DecAlt2_go'=true);
792
793   [tick] !DecAlt2_go -> 1: (DecAlt2_go'=true);
794 endmodule
795
796 //*****
797 // Utility Function
798 //*****
799
800 const int LOOSE = 0;
801 const int TIGHT = 1;
802 const int EMC_ON = 1;
803
804 formula probOfThreat = stateValue;
805
806 formula probabilityOfDestruction = probOfThreat
807   * ((f = LOOSE) ? 1.0 : (1.0 / destructionFormationFactor))
808   * ((c = EMC_ON) ? ecm_threat_prob : 1.0)
809   * max(0.0, threatRange - (a + 1)) / threatRange; // +1 because
        level 0 is one level above ground
810
811 module constraint // in this case the constraint is surviving
812   satisfied : bool init true;
813   [tick2] satisfied -> (1.0 - probabilityOfDestruction): (satisfied'=true);
814   + probabilityOfDestruction: (satisfied'=false);
815   [tick2] !satisfied -> true;
816 endmodule
817
818
819 formula probOfTarget= stateValue1;

```

```

820
821 formula probOfDetection = probOfTarget
822 * ((f = LOOSE) ? 1.0 : (1.0 / detectionFormationFactor))
823 * ((c = EMC_ON) ? ecm_target_prob : 1.0)
824 * max(0.0, sensorRange - (a + 1)) / sensorRange; // +1 because level 0
      is one level above ground
825
826 module sensor
827   targetDetected: bool init false;
828   [tick2] true -> probOfDetection: (targetDetected'=true) + (1.0 -
      probOfDetection): (targetDetected'=false);
829 endmodule
830
831 rewards "util"
832   [tick] (time < HORIZON) & satisfied & targetDetected : 1;
833   [tick] (time = HORIZON) & satisfied : (targetDetected ? 1 : 0) +
      survival_reward;
834
835   // give slight preference to not adapting
836   [tick] time = 0 & IncAlt_state=ini_IncAlt_state &
      DecAlt_state=ini_DecAlt_state & a=init_a & f=init_f :
      0.00000001;
837 endrewards

```

Listing 3. PRISM specification for short horizon MDP planning

*Long Horizon MDP Planning Specification:* To model uncertainty in targets and threats along a route, we adopt the approach suggested by Moreno et al. [35] since they also used the combination of DARTSim and MDP planning to evaluate their ideas. In short, two independent random variables are used in the environment state to represent the probabilities that a segment contains a target and a threat, respectively. Using the target and threat variables, we construct independent environment models for targets and threats, and then join them to produce a joint environment model, which is used by  $\rho_{mdps}$  and  $\rho_{mdpl}$ .

```

1  mdp
2  const double PERIOD = 60;
3  const int HORIZON = 5; // Planning horizon for deliberative planning
4  const double IncAlt_LATENCY = 60;
5  const double DecAlt_LATENCY = 60;
6  const int MAX_ALT_LEVEL = 3;
7  const double destructionFormationFactor = 1.5;
8  const double threatRange = 3;
9  const double detectionFormationFactor = 1.2;
10 const double sensorRange = 4;
11 const init_a = 0;
12 const init_c = 0;
13 const init_f = 0;
14 const bool ECM_ENABLED = true;
15 const bool ONE_LEVEL_ENABLED = true; // Unlike reactive planning, one
      level increase/decrease altitude enabled
16 const bool TWO_LEVEL_ENABLED = true; // Two level increase/decrease
      altitude also enabled
17 const int ini_IncAlt_state = 0;
18 const int ini_DecAlt_state = 0;
19 const int ini_IncAlt2_state = 0;
20 const int ini_DecAlt2_state = 0;
21 const double ecm_threat_prob = 0.15;
22 const double ecm_target_prob = 0.3;
23 const double survival_reward = 1;
24
25
26 //*****
27 // CLOCK
28 //*****
29 const int TO_TICK = 0;
30 const int TO_TICK2 = 1; // intermediate tick for constraint satisf. update
31 const int TO_TACK = 2;
32
33 label "final" = time = HORIZON & clockstep=TO_TICK;
34 formula sys_go = clockstep=TO_TICK;
35
36 module clk
37   time : [0..HORIZON] init 0;
38   clockstep : [0..2] init TO_TICK;
39
40   [tick] clockstep=TO_TICK & time < HORIZON -> 1: (time'=time+1) &
      (clockstep'=TO_TICK2);
41   [tick2] clockstep=TO_TICK2 -> 1: (clockstep'=TO_TACK);
42   [tick] clockstep=TO_TACK -> 1: (clockstep'=TO_TICK);
43 endmodule
44
45 module env
46 s : [0..45] init 0;
47 [tick] s = 0 ->
48   0.034225 : (s' = 1)
49   + 0.11655 : (s' = 2)
50   + 0.034225 : (s' = 3)
51   + 0.11655 : (s' = 4)
52   + 0.3969 : (s' = 5)
53   + 0.11655 : (s' = 6)
54   + 0.034225 : (s' = 7)
55   + 0.11655 : (s' = 8)
56   + 0.034225 : (s' = 9);
57 [tick] s = 1 ->
58   0.034225 : (s' = 10)
59   + 0.11655 : (s' = 11)
60   + 0.034225 : (s' = 12)
61   + 0.11655 : (s' = 13)
62   + 0.3969 : (s' = 14)
63   + 0.11655 : (s' = 15)
64   + 0.034225 : (s' = 16)
65   + 0.11655 : (s' = 17)

```

```
66 + 0.034225 : (s' = 18);
67 [tick] s = 2 ->
68 0.034225 : (s' = 10)
69 + 0.11655 : (s' = 11)
70 + 0.034225 : (s' = 12)
71 + 0.11655 : (s' = 13)
72 + 0.3969 : (s' = 14)
73 + 0.11655 : (s' = 15)
74 + 0.034225 : (s' = 16)
75 + 0.11655 : (s' = 17)
76 + 0.034225 : (s' = 18);
77 [tick] s = 3 ->
78 0.034225 : (s' = 10)
79 + 0.11655 : (s' = 11)
80 + 0.034225 : (s' = 12)
81 + 0.11655 : (s' = 13)
82 + 0.3969 : (s' = 14)
83 + 0.11655 : (s' = 15)
84 + 0.034225 : (s' = 16)
85 + 0.11655 : (s' = 17)
86 + 0.034225 : (s' = 18);
87 [tick] s = 4 ->
88 0.034225 : (s' = 10)
89 + 0.11655 : (s' = 11)
90 + 0.034225 : (s' = 12)
91 + 0.11655 : (s' = 13)
92 + 0.3969 : (s' = 14)
93 + 0.11655 : (s' = 15)
94 + 0.034225 : (s' = 16)
95 + 0.11655 : (s' = 17)
96 + 0.034225 : (s' = 18);
97 [tick] s = 5 ->
98 0.034225 : (s' = 10)
99 + 0.11655 : (s' = 11)
100 + 0.034225 : (s' = 12)
101 + 0.11655 : (s' = 13)
102 + 0.3969 : (s' = 14)
103 + 0.11655 : (s' = 15)
104 + 0.034225 : (s' = 16)
105 + 0.11655 : (s' = 17)
106 + 0.034225 : (s' = 18);
107 [tick] s = 6 ->
108 0.034225 : (s' = 10)
109 + 0.11655 : (s' = 11)
110 + 0.034225 : (s' = 12)
111 + 0.11655 : (s' = 13)
112 + 0.3969 : (s' = 14)
113 + 0.11655 : (s' = 15)
114 + 0.034225 : (s' = 16)
115 + 0.11655 : (s' = 17)
116 + 0.034225 : (s' = 18);
117 [tick] s = 7 ->
118 0.034225 : (s' = 10)
119 + 0.11655 : (s' = 11)
120 + 0.034225 : (s' = 12)
121 + 0.11655 : (s' = 13)
122 + 0.3969 : (s' = 14)
123 + 0.11655 : (s' = 15)
124 + 0.034225 : (s' = 16)
125 + 0.11655 : (s' = 17)
126 + 0.034225 : (s' = 18);
127 [tick] s = 8 ->
128 0.034225 : (s' = 10)
129 + 0.11655 : (s' = 11)
130 + 0.034225 : (s' = 12)
131 + 0.11655 : (s' = 13)
132 + 0.3969 : (s' = 14)
133 + 0.11655 : (s' = 15)
134 + 0.034225 : (s' = 16)
135 + 0.11655 : (s' = 17)
136 + 0.034225 : (s' = 18);
137 [tick] s = 9 ->
138 0.034225 : (s' = 10)
139 + 0.11655 : (s' = 11)
140 + 0.034225 : (s' = 12)
141 + 0.11655 : (s' = 13)
142 + 0.3969 : (s' = 14)
143 + 0.11655 : (s' = 15)
144 + 0.034225 : (s' = 16)
145 + 0.11655 : (s' = 17)
146 + 0.034225 : (s' = 18);
147 [tick] s = 10 ->
148 0.034225 : (s' = 19)
```

```
149 + 0.11655 : (s' = 20)
150 + 0.034225 : (s' = 21)
151 + 0.11655 : (s' = 22)
152 + 0.3969 : (s' = 23)
153 + 0.11655 : (s' = 24)
154 + 0.034225 : (s' = 25)
155 + 0.11655 : (s' = 26)
156 + 0.034225 : (s' = 27);
157 [tick] s = 11 ->
158 0.034225 : (s' = 19)
159 + 0.11655 : (s' = 20)
160 + 0.034225 : (s' = 21)
161 + 0.11655 : (s' = 22)
162 + 0.3969 : (s' = 23)
163 + 0.11655 : (s' = 24)
164 + 0.034225 : (s' = 25)
165 + 0.11655 : (s' = 26)
166 + 0.034225 : (s' = 27);
167 [tick] s = 12 ->
168 0.034225 : (s' = 19)
169 + 0.11655 : (s' = 20)
170 + 0.034225 : (s' = 21)
171 + 0.11655 : (s' = 22)
172 + 0.3969 : (s' = 23)
173 + 0.11655 : (s' = 24)
174 + 0.034225 : (s' = 25)
175 + 0.11655 : (s' = 26)
176 + 0.034225 : (s' = 27);
177 [tick] s = 13 ->
178 0.034225 : (s' = 19)
179 + 0.11655 : (s' = 20)
180 + 0.034225 : (s' = 21)
181 + 0.11655 : (s' = 22)
182 + 0.3969 : (s' = 23)
183 + 0.11655 : (s' = 24)
184 + 0.034225 : (s' = 25)
185 + 0.11655 : (s' = 26)
186 + 0.034225 : (s' = 27);
187 [tick] s = 14 ->
188 0.034225 : (s' = 19)
189 + 0.11655 : (s' = 20)
190 + 0.034225 : (s' = 21)
191 + 0.11655 : (s' = 22)
192 + 0.3969 : (s' = 23)
193 + 0.11655 : (s' = 24)
194 + 0.034225 : (s' = 25)
195 + 0.11655 : (s' = 26)
196 + 0.034225 : (s' = 27);
197 [tick] s = 15 ->
198 0.034225 : (s' = 19)
199 + 0.11655 : (s' = 20)
200 + 0.034225 : (s' = 21)
201 + 0.11655 : (s' = 22)
202 + 0.3969 : (s' = 23)
203 + 0.11655 : (s' = 24)
204 + 0.034225 : (s' = 25)
205 + 0.11655 : (s' = 26)
206 + 0.034225 : (s' = 27);
207 [tick] s = 16 ->
208 0.034225 : (s' = 19)
209 + 0.11655 : (s' = 20)
210 + 0.034225 : (s' = 21)
211 + 0.11655 : (s' = 22)
212 + 0.3969 : (s' = 23)
213 + 0.11655 : (s' = 24)
214 + 0.034225 : (s' = 25)
215 + 0.11655 : (s' = 26)
216 + 0.034225 : (s' = 27);
217 [tick] s = 17 ->
218 0.034225 : (s' = 19)
219 + 0.11655 : (s' = 20)
220 + 0.034225 : (s' = 21)
221 + 0.11655 : (s' = 22)
222 + 0.3969 : (s' = 23)
223 + 0.11655 : (s' = 24)
224 + 0.034225 : (s' = 25)
225 + 0.11655 : (s' = 26)
226 + 0.034225 : (s' = 27);
227 [tick] s = 18 ->
228 0.034225 : (s' = 19)
229 + 0.11655 : (s' = 20)
230 + 0.034225 : (s' = 21)
231 + 0.11655 : (s' = 22)
```

232	+ 0.3969 : (s' = 23)	315	+ 0.11655 : (s' = 35)
233	+ 0.11655 : (s' = 24)	316	+ 0.034225 : (s' = 36);
234	+ 0.034225 : (s' = 25)	317	[tick] s = 27 ->
235	+ 0.11655 : (s' = 26)	318	0.034225 : (s' = 28)
236	+ 0.034225 : (s' = 27);	319	+ 0.11655 : (s' = 29)
237	[tick] s = 19 ->	320	+ 0.034225 : (s' = 30)
238	0.034225 : (s' = 28)	321	+ 0.11655 : (s' = 31)
239	+ 0.11655 : (s' = 29)	322	+ 0.3969 : (s' = 32)
240	+ 0.034225 : (s' = 30)	323	+ 0.11655 : (s' = 33)
241	+ 0.11655 : (s' = 31)	324	+ 0.034225 : (s' = 34)
242	+ 0.3969 : (s' = 32)	325	+ 0.11655 : (s' = 35)
243	+ 0.11655 : (s' = 33)	326	+ 0.034225 : (s' = 36);
244	+ 0.034225 : (s' = 34)	327	[tick] s = 28 ->
245	+ 0.11655 : (s' = 35)	328	0.034225 : (s' = 37)
246	+ 0.034225 : (s' = 36);	329	+ 0.11655 : (s' = 38)
247	[tick] s = 20 ->	330	+ 0.034225 : (s' = 39)
248	0.034225 : (s' = 28)	331	+ 0.11655 : (s' = 40)
249	+ 0.11655 : (s' = 29)	332	+ 0.3969 : (s' = 41)
250	+ 0.034225 : (s' = 30)	333	+ 0.11655 : (s' = 42)
251	+ 0.11655 : (s' = 31)	334	+ 0.034225 : (s' = 43)
252	+ 0.3969 : (s' = 32)	335	+ 0.11655 : (s' = 44)
253	+ 0.11655 : (s' = 33)	336	+ 0.034225 : (s' = 45);
254	+ 0.034225 : (s' = 34)	337	[tick] s = 29 ->
255	+ 0.11655 : (s' = 35)	338	0.034225 : (s' = 37)
256	+ 0.034225 : (s' = 36);	339	+ 0.11655 : (s' = 38)
257	[tick] s = 21 ->	340	+ 0.034225 : (s' = 39)
258	0.034225 : (s' = 28)	341	+ 0.11655 : (s' = 40)
259	+ 0.11655 : (s' = 29)	342	+ 0.3969 : (s' = 41)
260	+ 0.034225 : (s' = 30)	343	+ 0.11655 : (s' = 42)
261	+ 0.11655 : (s' = 31)	344	+ 0.034225 : (s' = 43)
262	+ 0.3969 : (s' = 32)	345	+ 0.11655 : (s' = 44)
263	+ 0.11655 : (s' = 33)	346	+ 0.034225 : (s' = 45);
264	+ 0.034225 : (s' = 34)	347	[tick] s = 30 ->
265	+ 0.11655 : (s' = 35)	348	0.034225 : (s' = 37)
266	+ 0.034225 : (s' = 36);	349	+ 0.11655 : (s' = 38)
267	[tick] s = 22 ->	350	+ 0.034225 : (s' = 39)
268	0.034225 : (s' = 28)	351	+ 0.11655 : (s' = 40)
269	+ 0.11655 : (s' = 29)	352	+ 0.3969 : (s' = 41)
270	+ 0.034225 : (s' = 30)	353	+ 0.11655 : (s' = 42)
271	+ 0.11655 : (s' = 31)	354	+ 0.034225 : (s' = 43)
272	+ 0.3969 : (s' = 32)	355	+ 0.11655 : (s' = 44)
273	+ 0.11655 : (s' = 33)	356	+ 0.034225 : (s' = 45);
274	+ 0.034225 : (s' = 34)	357	[tick] s = 31 ->
275	+ 0.11655 : (s' = 35)	358	0.034225 : (s' = 37)
276	+ 0.034225 : (s' = 36);	359	+ 0.11655 : (s' = 38)
277	[tick] s = 23 ->	360	+ 0.034225 : (s' = 39)
278	0.034225 : (s' = 28)	361	+ 0.11655 : (s' = 40)
279	+ 0.11655 : (s' = 29)	362	+ 0.3969 : (s' = 41)
280	+ 0.034225 : (s' = 30)	363	+ 0.11655 : (s' = 42)
281	+ 0.11655 : (s' = 31)	364	+ 0.034225 : (s' = 43)
282	+ 0.3969 : (s' = 32)	365	+ 0.11655 : (s' = 44)
283	+ 0.11655 : (s' = 33)	366	+ 0.034225 : (s' = 45);
284	+ 0.034225 : (s' = 34)	367	[tick] s = 32 ->
285	+ 0.11655 : (s' = 35)	368	0.034225 : (s' = 37)
286	+ 0.034225 : (s' = 36);	369	+ 0.11655 : (s' = 38)
287	[tick] s = 24 ->	370	+ 0.034225 : (s' = 39)
288	0.034225 : (s' = 28)	371	+ 0.11655 : (s' = 40)
289	+ 0.11655 : (s' = 29)	372	+ 0.3969 : (s' = 41)
290	+ 0.034225 : (s' = 30)	373	+ 0.11655 : (s' = 42)
291	+ 0.11655 : (s' = 31)	374	+ 0.034225 : (s' = 43)
292	+ 0.3969 : (s' = 32)	375	+ 0.11655 : (s' = 44)
293	+ 0.11655 : (s' = 33)	376	+ 0.034225 : (s' = 45);
294	+ 0.034225 : (s' = 34)	377	[tick] s = 33 ->
295	+ 0.11655 : (s' = 35)	378	0.034225 : (s' = 37)
296	+ 0.034225 : (s' = 36);	379	+ 0.11655 : (s' = 38)
297	[tick] s = 25 ->	380	+ 0.034225 : (s' = 39)
298	0.034225 : (s' = 28)	381	+ 0.11655 : (s' = 40)
299	+ 0.11655 : (s' = 29)	382	+ 0.3969 : (s' = 41)
300	+ 0.034225 : (s' = 30)	383	+ 0.11655 : (s' = 42)
301	+ 0.11655 : (s' = 31)	384	+ 0.034225 : (s' = 43)
302	+ 0.3969 : (s' = 32)	385	+ 0.11655 : (s' = 44)
303	+ 0.11655 : (s' = 33)	386	+ 0.034225 : (s' = 45);
304	+ 0.034225 : (s' = 34)	387	[tick] s = 34 ->
305	+ 0.11655 : (s' = 35)	388	0.034225 : (s' = 37)
306	+ 0.034225 : (s' = 36);	389	+ 0.11655 : (s' = 38)
307	[tick] s = 26 ->	390	+ 0.034225 : (s' = 39)
308	0.034225 : (s' = 28)	391	+ 0.11655 : (s' = 40)
309	+ 0.11655 : (s' = 29)	392	+ 0.3969 : (s' = 41)
310	+ 0.034225 : (s' = 30)	393	+ 0.11655 : (s' = 42)
311	+ 0.11655 : (s' = 31)	394	+ 0.034225 : (s' = 43)
312	+ 0.3969 : (s' = 32)	395	+ 0.11655 : (s' = 44)
313	+ 0.11655 : (s' = 33)	396	+ 0.034225 : (s' = 45);
314	+ 0.034225 : (s' = 34)	397	[tick] s = 35 ->

```

398 0.034225 : (s' = 37)
399 + 0.11655 : (s' = 38)
400 + 0.034225 : (s' = 39)
401 + 0.11655 : (s' = 40)
402 + 0.3969 : (s' = 41)
403 + 0.11655 : (s' = 42)
404 + 0.034225 : (s' = 43)
405 + 0.11655 : (s' = 44)
406 + 0.034225 : (s' = 45);
407 [tick] s = 36 ->
408 0.034225 : (s' = 37)
409 + 0.11655 : (s' = 38)
410 + 0.034225 : (s' = 39)
411 + 0.11655 : (s' = 40)
412 + 0.3969 : (s' = 41)
413 + 0.11655 : (s' = 42)
414 + 0.034225 : (s' = 43)
415 + 0.11655 : (s' = 44)
416 + 0.034225 : (s' = 45);
417 endmodule
418
419 // environment has 2 components. statevalue for threats and statevalue1 is
    for targets
420 formula stateValue = (s = 0 ? 0 : 0) +
421 (s = 1 ? 0.00605639 : 0) +
422 (s = 2 ? 0.00605639 : 0) +
423 (s = 3 ? 0.00605639 : 0) +
424 (s = 4 ? 0.0282836 : 0) +
425 (s = 5 ? 0.0282836 : 0) +
426 (s = 6 ? 0.0282836 : 0) +
427 (s = 7 ? 0.0778979 : 0) +
428 (s = 8 ? 0.0778979 : 0) +
429 (s = 9 ? 0.0778979 : 0) +
430 (s = 10 ? 0 : 0) +
431 (s = 11 ? 0 : 0) +
432 (s = 12 ? 0 : 0) +
433 (s = 13 ? 0 : 0) +
434 (s = 14 ? 0 : 0) +
435 (s = 15 ? 0 : 0) +
436 (s = 16 ? 0 : 0) +
437 (s = 17 ? 0 : 0) +
438 (s = 18 ? 0 : 0) +
439 (s = 19 ? 0.750751 : 0) +
440 (s = 20 ? 0.750751 : 0) +
441 (s = 21 ? 0.750751 : 0) +
442 (s = 22 ? 0.885424 : 0) +
443 (s = 23 ? 0.885424 : 0) +
444 (s = 24 ? 0.885424 : 0) +
445 (s = 25 ? 0.963485 : 0) +
446 (s = 26 ? 0.963485 : 0) +
447 (s = 27 ? 0.963485 : 0) +
448 (s = 28 ? 0.435626 : 0) +
449 (s = 29 ? 0.435626 : 0) +
450 (s = 30 ? 0.435626 : 0) +
451 (s = 31 ? 0.676196 : 0) +
452 (s = 32 ? 0.676196 : 0) +
453 (s = 33 ? 0.676196 : 0) +
454 (s = 34 ? 0.864925 : 0) +
455 (s = 35 ? 0.864925 : 0) +
456 (s = 36 ? 0.864925 : 0) +
457 (s = 37 ? 0.368403 : 0) +
458 (s = 38 ? 0.368403 : 0) +
459 (s = 39 ? 0.368403 : 0) +
460 (s = 40 ? 0.793701 : 0) +
461 (s = 41 ? 0.793701 : 0) +
462 (s = 42 ? 0.793701 : 0) +
463 (s = 43 ? 0.983048 : 0) +
464 (s = 44 ? 0.983048 : 0) +
465 (s = 45 ? 0.983048 : 0);
466
467 formula stateValue1 = (s = 0 ? 0 : 0) +
468 (s = 1 ? 0.830037 : 0) +
469 (s = 2 ? 0.904439 : 0) +
470 (s = 3 ? 0.954777 : 0) +
471 (s = 4 ? 0.830037 : 0) +
472 (s = 5 ? 0.904439 : 0) +
473 (s = 6 ? 0.954777 : 0) +
474 (s = 7 ? 0.830037 : 0) +
475 (s = 8 ? 0.904439 : 0) +
476 (s = 9 ? 0.954777 : 0) +
477 (s = 10 ? 0 : 0) +
478 (s = 11 ? 0 : 0) +
479 (s = 12 ? 0 : 0) +
480 (s = 13 ? 0 : 0) +
481 (s = 14 ? 0 : 0) +
482 (s = 15 ? 0 : 0) +
483 (s = 16 ? 0 : 0) +
484 (s = 17 ? 0 : 0) +
485 (s = 18 ? 0 : 0) +
486 (s = 19 ? 0.0156741 : 0) +
487 (s = 20 ? 0.0719057 : 0) +
488 (s = 21 ? 0.190204 : 0) +
489 (s = 22 ? 0.0156741 : 0) +
490 (s = 23 ? 0.0719057 : 0) +
491 (s = 24 ? 0.190204 : 0) +
492 (s = 25 ? 0.0156741 : 0) +
493 (s = 26 ? 0.0719057 : 0) +
494 (s = 27 ? 0.190204 : 0) +
495 (s = 28 ? 0 : 0) +
496 (s = 29 ? 0 : 0) +
497 (s = 30 ? 0 : 0) +
498 (s = 31 ? 0 : 0) +
499 (s = 32 ? 0 : 0) +
500 (s = 33 ? 0 : 0) +
501 (s = 34 ? 0 : 0) +
502 (s = 35 ? 0 : 0) +
503 (s = 36 ? 0 : 0) +
504 (s = 37 ? 0 : 0) +
505 (s = 38 ? 0 : 0) +
506 (s = 39 ? 0 : 0) +
507 (s = 40 ? 0 : 0) +
508 (s = 41 ? 0 : 0) +
509 (s = 42 ? 0 : 0) +
510 (s = 43 ? 0 : 0) +
511 (s = 44 ? 0 : 0) +
512 (s = 45 ? 0 : 0);
513 // #ENV ENDS
514
515 //*****
516 // SYSTEM
517 //*****
518
519 // Variable range and initialization
520 const a_MIN=0; const a_MAX=MAX_ALT_LEVEL; const a_INIT=init_a;
521 const f_MIN=0; const f_MAX=1; const f_INIT=init_f;
522 const c_MIN=0; const c_MAX=1; const c_INIT=init_c;
523
524 module sys
525 a : [a_MIN..a_MAX] init a_INIT;
526 f : [f_MIN..f_MAX] init f_INIT;
527 c : [c_MIN..c_MAX] init c_INIT;
528
529 [EcmOn_start] c=0 & ECM_ENABLED -> 1: (c'=c_EcmOn_impact);
530 [EcmOff_start] c=1 & ECM_ENABLED -> 1: (c'=c_EcmOff_impact);
531
532 [GoTight_start] f=0 -> 1: (a'=a_GoTight_impact)
533 & (f'=f_GoTight_impact);
534 [GoLoose_start] f=1 -> 1: (a'=a_GoLoose_impact)
535 & (f'=f_GoLoose_impact);
536
537 [IncAlt_complete] a < MAX_ALT_LEVEL & ONE_LEVEL_ENABLED ->
538 1: (a'=a_IncAlt_impact)
539 & (f'=f_IncAlt_impact);
540 [IncAlt2_complete] a < MAX_ALT_LEVEL-1 &
541 TWO_LEVEL_ENABLED -> 1: (a'=a_IncAlt2_impact);
542
543 [DecAlt_complete] a > 0 & ONE_LEVEL_ENABLED -> 1:
544 (a'=a_DecAlt_impact)
545 & (f'=f_DecAlt_impact);
546 [DecAlt2_complete] a > 1 & TWO_LEVEL_ENABLED -> 1:
547 (a'=a_DecAlt2_impact);
548
549 endmodule
550
551 formula c_EcmOn_impact = c + (1) >= c_MIN ? (c+(1)<=c_MAX? c+(1) :
552 c_MAX) : c_MIN;
553 formula c_EcmOff_impact = c + (-1) >= c_MIN ? (c+(-1)<=c_MAX?
554 c+(-1) : c_MAX) : c_MIN;
555 formula a_GoTight_impact = a + (0) >= a_MIN ? (a+(0)<=a_MAX? a+(0) :
556 a_MAX) : a_MIN;
557 formula f_GoTight_impact = f + (1) >= f_MIN ? (f+(1)<=f_MAX? f+(1) :
558 f_MAX) : f_MIN;
559 formula a_GoLoose_impact = a + (0) >= a_MIN ? (a+(0)<=a_MAX? a+(0) :
560 a_MAX) : a_MIN;
561 formula f_GoLoose_impact = f + (-1) >= f_MIN ? (f+(-1)<=f_MAX? f+(-1)

```

```

: f_MAX) : f_MIN;
554 formula a_IncAlt_impact = a + (1) >= a_MIN ? ( a+(1)<=a_MAX? a+(1) :
a_MAX) : a_MIN;
555 formula f_IncAlt_impact = f + (0) >= f_MIN ? ( f+(0)<=f_MAX? f+(0) :
f_MAX) : f_MIN;
556 formula a_DecAlt_impact = a + (-1) >= a_MIN ? ( a+(-1)<=a_MAX?
a+(-1) : a_MAX) : a_MIN;
557 formula f_DecAlt_impact = f + (0) >= f_MIN ? ( f+(0)<=f_MAX? f+(0) :
f_MAX) : f_MIN;
558 formula a_IncAlt2_impact = a + (2) >= a_MIN ? ( a+(2)<=a_MAX? a+(2) :
a_MAX) : a_MIN;
559 formula a_DecAlt2_impact = a + (-2) >= a_MIN ? ( a+(-2)<=a_MAX?
a+(-2) : a_MAX) : a_MIN;
560
561 // tactic concurrency rules
562 formula IncAlt_used = IncAlt_state != 0;
563 formula DecAlt_used = DecAlt_state != 0;
564 formula IncAlt2_used = IncAlt2_state != 0;
565 formula DecAlt2_used = DecAlt2_state != 0;
566
567 formula EcmOn_compatible = !EcmOn_used;
568 formula EcmOff_compatible = !EcmOff_used;
569 formula GoTight_compatible = !GoLoose_used;
570 formula GoLoose_compatible = !GoTight_used;
571 formula IncAlt_compatible = (!DecAlt_used) & (!IncAlt2_used) &
(!DecAlt2_used);
572 formula DecAlt_compatible = (!IncAlt_used) & (!IncAlt2_used) &
(!DecAlt2_used);
573 formula IncAlt2_compatible = (!DecAlt_used) & (!IncAlt_used) &
(!DecAlt2_used);
574 formula DecAlt2_compatible = (!DecAlt_used) & (!IncAlt_used) &
(!IncAlt2_used);
575
576
577 //*****
578 // TACTIC: EcmOn
579 //*****
580
581 // Applicability conditions
582 formula EcmOn_applicable = EcmOn_compatible & c=0;
583
584 module EcmOn
585   EcmOn_used : bool init false;
586   EcmOn_go : bool init true;
587
588   // Tactic applicable, start it
589   [EcmOn_start] sys_go & EcmOn_go & EcmOn_applicable &
ECM_ENABLED -> (EcmOn_used'=true) & (EcmOn_go'=false);
590
591   // Tactic applicable, but do not start it
592   [EcmOn_pass] sys_go & EcmOn_go & EcmOn_applicable ->
(EcmOn_go'=false);
593
594   // Pass if the tactic is not applicable
595   [EcmOn_invalid] sys_go & EcmOn_go & !EcmOn_applicable -> 1 :
(EcmOn_go'=false);
596
597   [tick] !EcmOn_go -> 1: (EcmOn_go'=true) & (EcmOn_used'=false);
598 endmodule
599
600
601 //*****
602 // TACTIC: EcmOff
603 //*****
604
605 // Applicability conditions
606 formula EcmOff_applicable = EcmOff_compatible & c=1;
607
608 module EcmOff
609   EcmOff_used : bool init false;
610   EcmOff_go : bool init true;
611
612   // Tactic applicable, start it
613   [EcmOff_start] sys_go & EcmOff_go & EcmOff_applicable &
ECM_ENABLED -> (EcmOff_used'=true) & (EcmOff_go'=false);
614
615   // Tactic applicable, but do not start it
616   [EcmOff_pass] sys_go & EcmOff_go & EcmOff_applicable ->
(EcmOff_go'=false);
617
618   // Pass if the tactic is not applicable
619   [EcmOff_invalid] sys_go & EcmOff_go & !EcmOff_applicable -> 1 :
(EcmOff_go'=false);
620
621   [tick] !EcmOff_go -> 1: (EcmOff_go'=true) & (EcmOff_used'=false);
622 endmodule
623
624
625 //*****
626 // TACTIC: GoTight
627 //*****
628
629 // Applicability conditions
630 formula GoTight_applicable = GoTight_compatible & f=0;
631
632 module GoTight
633   GoTight_used : bool init false;
634   GoTight_go : bool init true;
635
636   // Tactic applicable, start it
637   [GoTight_start] sys_go & GoTight_go & GoTight_applicable ->
(GoTight_used'=true) & (GoTight_go'=false);
638
639   // Tactic applicable, but do not start it
640   [GoTight_pass] sys_go & GoTight_go & GoTight_applicable ->
(GoTight_go'=false);
641
642   // Pass if the tactic is not applicable
643   [GoTight_invalid] sys_go & GoTight_go & !GoTight_applicable -> 1 :
(GoTight_go'=false);
644
645   [tick] !GoTight_go -> 1: (GoTight_go'=true) & (GoTight_used'=false);
646 endmodule
647
648
649 //*****
650 // TACTIC: GoLoose
651 //*****
652
653 // Applicability conditions
654 formula GoLoose_applicable = GoLoose_compatible & f=1;
655
656 module GoLoose
657   GoLoose_used : bool init false;
658   GoLoose_go : bool init true;
659
660   // Tactic applicable, start it
661   [GoLoose_start] sys_go & GoLoose_go & GoLoose_applicable ->
(GoLoose_used'=true) & (GoLoose_go'=false);
662
663   // Tactic applicable, but do not start it
664   [GoLoose_pass] sys_go & GoLoose_go & GoLoose_applicable ->
(GoLoose_go'=false);
665
666   // Pass if the tactic is not applicable
667   [GoLoose_invalid] sys_go & GoLoose_go & !GoLoose_applicable -> 1 :
(GoLoose_go'=false);
668
669   [tick] !GoLoose_go -> 1: (GoLoose_go'=true) &
(GoLoose_used'=false);
670 endmodule
671
672
673 //*****
674 // TACTIC: IncAlt
675 //*****
676
677 const int IncAlt_LATENCY_PERIODS = ceil(IncAlt_LATENCY/PERIOD);
678
679 // Applicability conditions
680 formula IncAlt_applicable = IncAlt_compatible & a < MAX_ALT_LEVEL;
681
682 module IncAlt
683   IncAlt_state : [0..IncAlt_LATENCY_PERIODS] init ini_IncAlt_state;
684   IncAlt_go : bool init true;
685
686   // Tactic applicable, start it
687   [IncAlt_start] sys_go & IncAlt_go & IncAlt_state=0 & IncAlt_applicable &
ONE_LEVEL_ENABLED ->
(IncAlt_state'=IncAlt_LATENCY_PERIODS) & (IncAlt_go'=false);
688
689   // Tactic applicable, but do not start it
690   [IncAlt_pass] sys_go & IncAlt_go & IncAlt_state=0 & IncAlt_applicable
-> (IncAlt_go'=false);
691
692   // Pass if the tactic is not applicable

```



```

693 [IncAlt_invalid] sys_go & IncAlt_go & IncAlt_state=0 & !IncAlt_applicable
694     -> 1 : (IncAlt_go'=false);
695 // Progress of the tactic
696 [IncAlt_progress] sys_go & IncAlt_go & IncAlt_state > 1 -> 1:
697     (IncAlt_state'=IncAlt_state-1) & (IncAlt_go'=false);
698 // Completion of the tactic
699 [IncAlt_complete] sys_go & IncAlt_go & IncAlt_state=1 -> 1:
700     (IncAlt_state'=0) & (IncAlt_go'=true);
701 [tick] !IncAlt_go -> 1: (IncAlt_go'=true);
702 endmodule
703
704 //*****
705 // TACTIC: DecAlt
706 //*****
707
708 const int DecAlt_LATENCY_PERIODS = ceil(DecAlt_LATENCY/PERIOD);
709
710 // Applicability conditions
711 formula DecAlt_applicable = DecAlt_compatible & a > 0;
712
713 module DecAlt
714     DecAlt_state : [0..DecAlt_LATENCY_PERIODS] init ini_DecAlt_state;
715     DecAlt_go : bool init true;
716
717 // Tactic applicable, start it
718 [DecAlt_start] sys_go & DecAlt_go & DecAlt_state=0 &
719     DecAlt_applicable & ONE_LEVEL_ENABLED ->
720     (DecAlt_state'=DecAlt_LATENCY_PERIODS) &
721     (DecAlt_go'=false);
722
723 // Tactic applicable, but do not start it
724 [DecAlt_pass] sys_go & DecAlt_go & DecAlt_state=0 &
725     DecAlt_applicable -> (DecAlt_go'=false);
726
727 // Pass if the tactic is not applicable
728 [DecAlt_invalid] sys_go & DecAlt_go & DecAlt_state=0 &
729     !DecAlt_applicable -> 1 : (DecAlt_go'=false);
730
731 // Progress of the tactic
732 [DecAlt_progress] sys_go & DecAlt_go & DecAlt_state > 1 -> 1:
733     (DecAlt_state'=DecAlt_state-1) & (DecAlt_go'=false);
734
735 // Completion of the tactic
736 [DecAlt_complete] sys_go & DecAlt_go & DecAlt_state=1 -> 1:
737     (DecAlt_state'=0) & (DecAlt_go'=true);
738
739 [tick] !DecAlt_go -> 1: (DecAlt_go'=true);
740 endmodule
741
742 //*****
743 // TACTIC: IncAlt2
744 //*****
745
746 // Applicability conditions
747 formula IncAlt2_applicable = IncAlt2_compatible & a <
748     MAX_ALT_LEVEL-1;
749
750 module IncAlt2
751     IncAlt2_state : [0..IncAlt2_LATENCY_PERIODS] init ini_IncAlt2_state;
752     IncAlt2_go : bool init true;
753
754 // Tactic applicable, start it
755 [IncAlt2_start] sys_go & IncAlt2_go & IncAlt2_state=0 &
756     IncAlt2_applicable & TWO_LEVEL_ENABLED ->
757     (IncAlt2_state'=IncAlt2_LATENCY_PERIODS) &
758     (IncAlt2_go'=false);
759
760 // Tactic applicable, but do not start it
761 [IncAlt2_pass] sys_go & IncAlt2_go & IncAlt2_state=0 &
762     IncAlt2_applicable -> (IncAlt2_go'=false);
763
764 // Pass if the tactic is not applicable
765 [IncAlt2_invalid] sys_go & IncAlt2_go & IncAlt2_state=0 &
766     !IncAlt2_applicable -> 1 : (IncAlt2_go'=false);
767
768 // Progress of the tactic
769 [IncAlt2_progress] sys_go & IncAlt2_go & IncAlt2_state > 1 -> 1:
770     (IncAlt2_state'=IncAlt2_state-1) & (IncAlt2_go'=false);
771
772 // Completion of the tactic
773 [IncAlt2_complete] sys_go & IncAlt2_go & IncAlt2_state=1 -> 1:
774     (IncAlt2_state'=0) & (IncAlt2_go'=true);
775
776 [tick] !IncAlt2_go -> 1: (IncAlt2_go'=true);
777 endmodule
778
779 //*****
780 // TACTIC: DecAlt2
781 //*****
782
783 const int DecAlt2_LATENCY_PERIODS = ceil(DecAlt2_LATENCY/PERIOD);
784
785 // Applicability conditions
786 formula DecAlt2_applicable = DecAlt2_compatible & a > 1;
787
788 module DecAlt2
789     DecAlt2_state : [0..DecAlt2_LATENCY_PERIODS] init
790         ini_DecAlt2_state;
791     DecAlt2_go : bool init true;
792
793 // Tactic applicable, start it
794 [DecAlt2_start] sys_go & DecAlt2_go & DecAlt2_state=0 &
795     DecAlt2_applicable & TWO_LEVEL_ENABLED ->
796     (DecAlt2_state'=DecAlt2_LATENCY_PERIODS) &
797     (DecAlt2_go'=false);
798
799 // Tactic applicable, but do not start it
800 [DecAlt2_pass] sys_go & DecAlt2_go & DecAlt2_state=0 &
801     DecAlt2_applicable -> (DecAlt2_go'=false);
802
803 // Pass if the tactic is not applicable
804 [DecAlt2_invalid] sys_go & DecAlt2_go & DecAlt2_state=0 &
805     !DecAlt2_applicable -> 1 : (DecAlt2_go'=false);
806
807 // Progress of the tactic
808 [DecAlt2_progress] sys_go & DecAlt2_go & DecAlt2_state > 1 -> 1:
809     (DecAlt2_state'=DecAlt2_state-1) & (DecAlt2_go'=false);
810
811 // Completion of the tactic
812 [DecAlt2_complete] sys_go & DecAlt2_go & DecAlt2_state=1 -> 1:
813     (DecAlt2_state'=0) & (DecAlt2_go'=true);
814
815 [tick] !DecAlt2_go -> 1: (DecAlt2_go'=true);
816 endmodule
817
818 //*****
819 // Utility Function
820 //*****
821
822 const int LOOSE = 0;
823 const int TIGHT = 1;
824 const int EMC_ON = 1;
825
826 formula probOfThreat = stateValue;
827
828 formula probabilityOfDestruction = probOfThreat
829     * ((f = LOOSE) ? 1.0 : (1.0 / destructionFormationFactor))
830     * ((c = EMC_ON) ? ecm_threat_prob : 1.0)
831     * max(0.0, threatRange - (a + 1)) / threatRange; // +1 because
832     level 0 is one level above ground
833
834 module constraint // in this case the constraint is surviving
835     satisfied: bool init true;
836     [tick2] satisfied -> (1.0 - probabilityOfDestruction): (satisfied'=true)
837         + probabilityOfDestruction: (satisfied'=false);
838     [tick2] !satisfied -> true;
839 endmodule
840
841 formula probOfTarget= stateValue1;
842
843 formula probOfDetection = probOfTarget
844     * ((f = LOOSE) ? 1.0 : (1.0 / detectionFormationFactor))
845     * ((c = EMC_ON) ? ecm_target_prob : 1.0)
846     * max(0.0, sensorRange - (a + 1)) / sensorRange; // +1 because level 0
847     is one level above ground
848
849 module sensor
850     targetDetected: bool init false;
851     [tick2] true -> probOfDetection: (targetDetected'=true) + (1.0 -
852         probOfDetection): (targetDetected'=false);

```

```

830 endmodule
831
832 rewards "util"
833 // [tick] satisfied & targetDetected : 1;
834
835 [tick] (time < HORIZON) & satisfied & targetDetected : 1;
836 [tick] (time = HORIZON) & satisfied : (targetDetected ? 1 : 0) +
      survival_reward;
837
838 // give slight preference to not adapting
839 [tick] time = 0 & IncAlt_state=ini_IncAlt_state &
      DecAlt_state=ini_DecAlt_state & a=init_a & f=init_f :
      0.000000001;
840 endrewards

```

Listing 4. PRISM specification for long horizon MDP planning

## VI. MATERIALS FOR TRACE PATTERNS

### *Q: What Are the Load Patterns for the Cloud-based System?*

To construct a realistic environment of users accessing the cloud-based system, we used a research dataset with online traffic common in web analytics — the daily traces of user requests from the FIFA-98 WorldCup website. These traces are independent day-by-day recordings of user website activity during the championship, with rapid load changes and periods of low variation. We picked these traces because they contain the patterns for high-demand cloud systems. As shown in Figure 2.

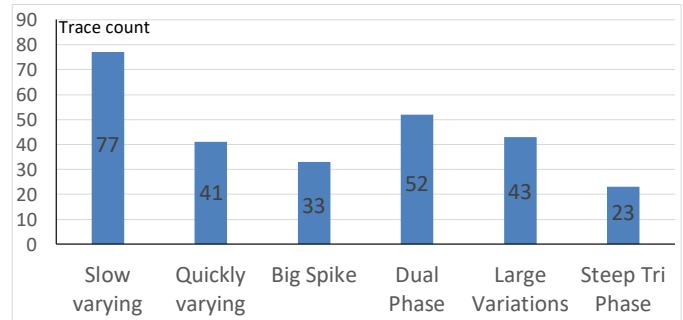
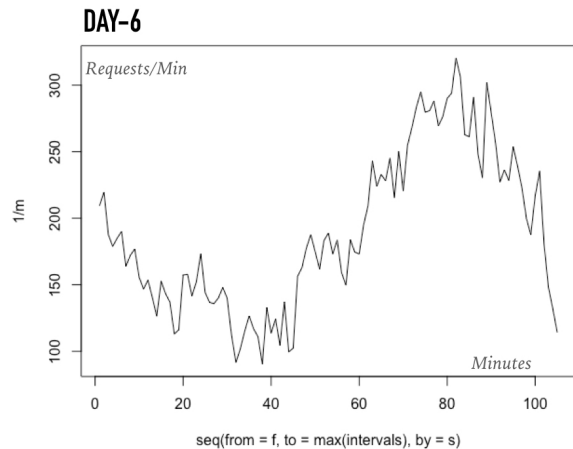


Figure 2. Patterns observed in traces. A single trace can have multiple patterns.

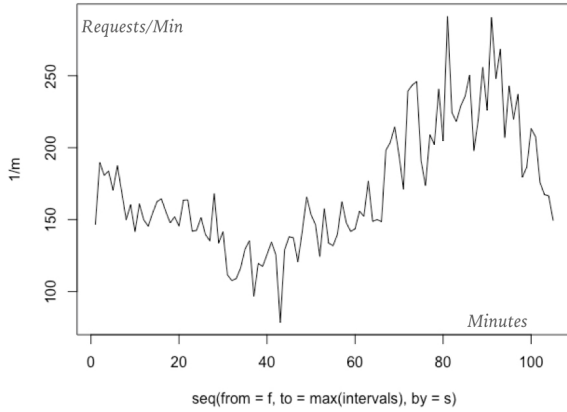
Total traces were 92 but 5 were incomplete/corrupted. Therefore, we used 87 traces for validation. Each trace is scaled such that

- duration is 105 minutes;
- starting request arrival rate is about 200 requests/min since active servers beginning of a simulation can server 200 requests/min. If workload goes beyond the capacity, the queueing model does not work;
- similarly, the highest workload is about 800, which is 90of the the total capacity (including all available servers with no optional content) of the system.

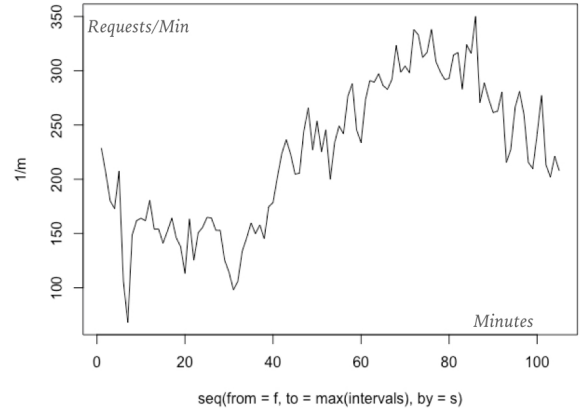
Below are the visualizations of load patterns for each day.



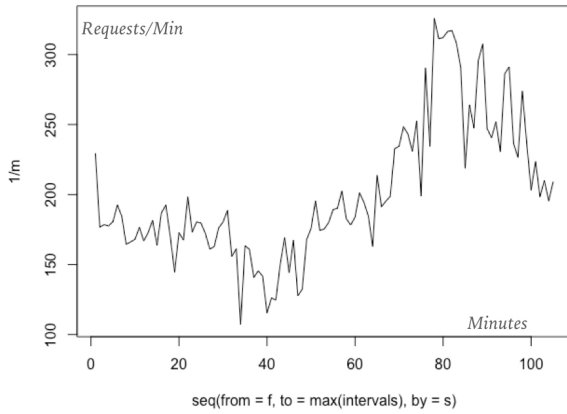
### DAY-7



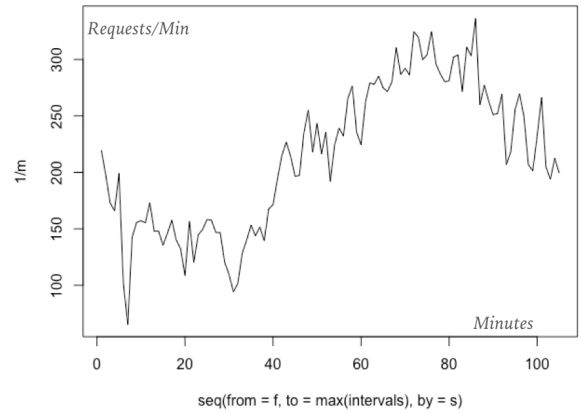
### DAY-10



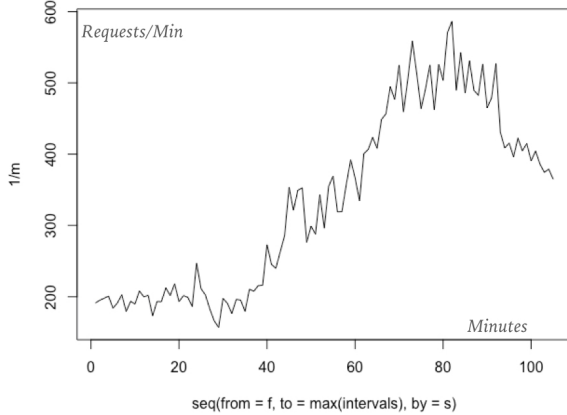
### DAY-8



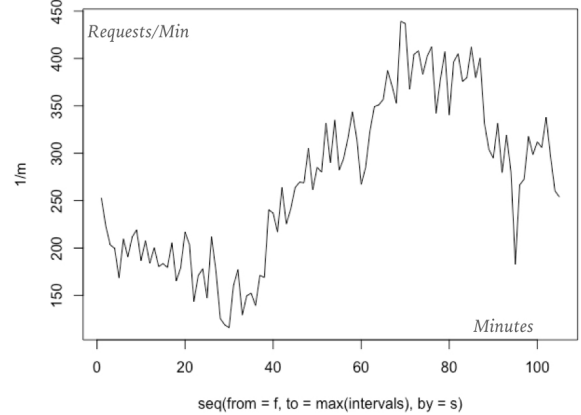
### DAY-11



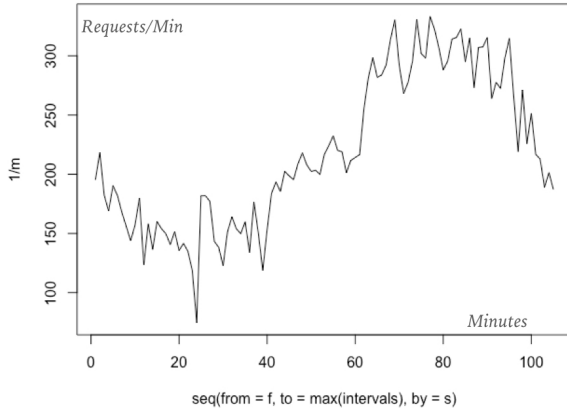
### DAY-9



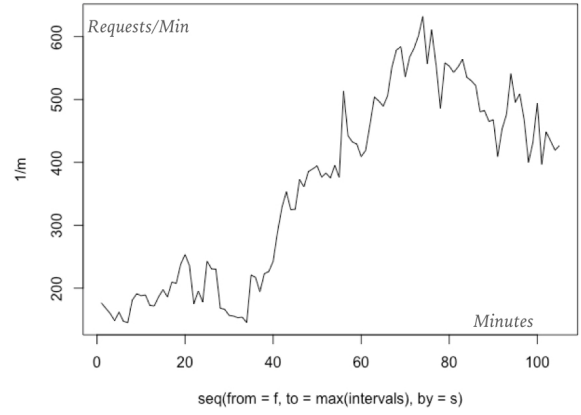
### DAY-12



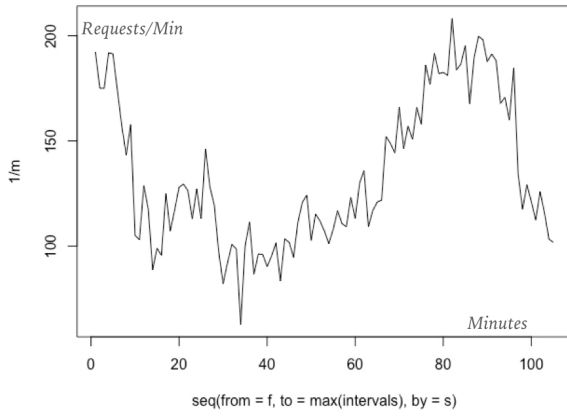
### DAY-13



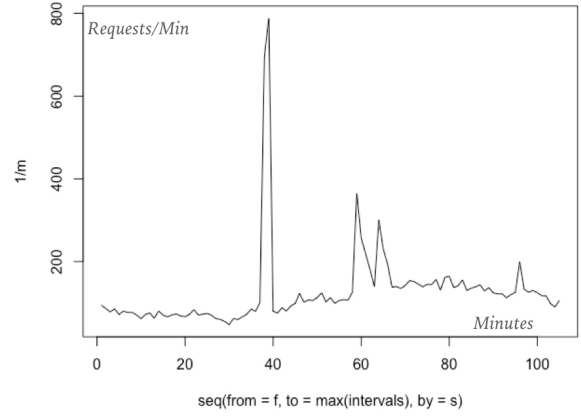
### DAY-16



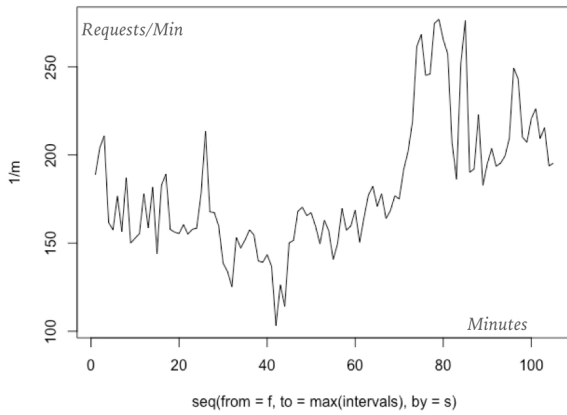
### DAY-14



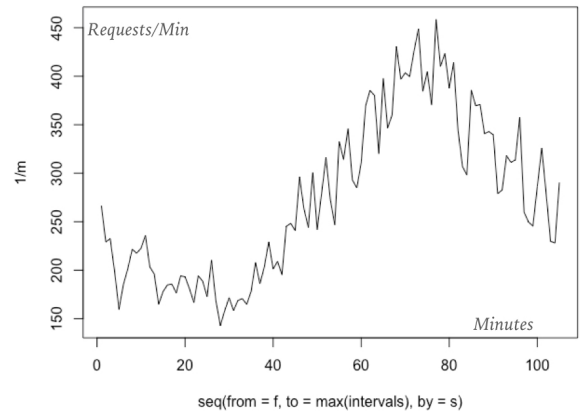
### DAY-17



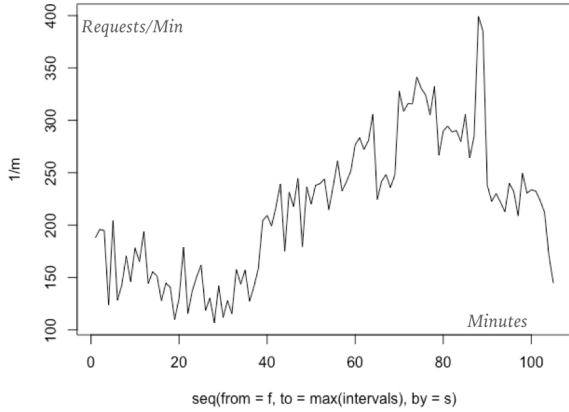
### DAY-15



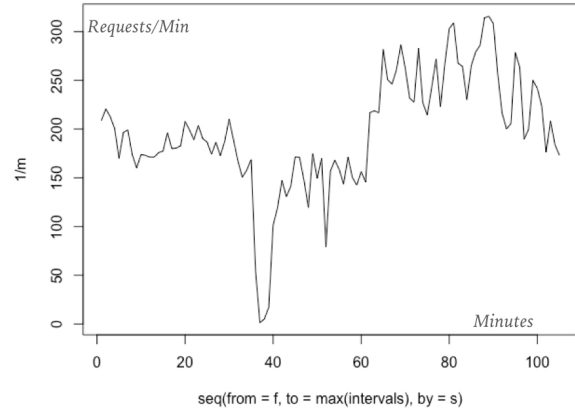
### DAY-18



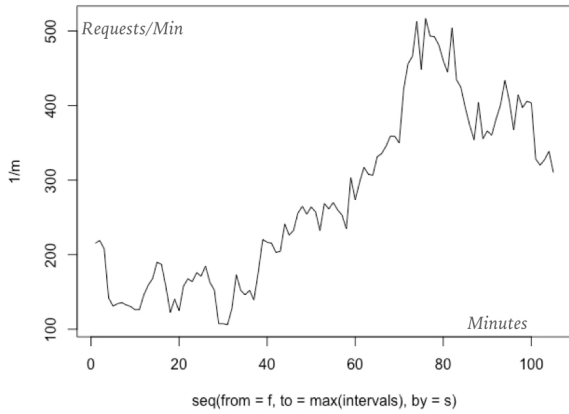
### DAY-19



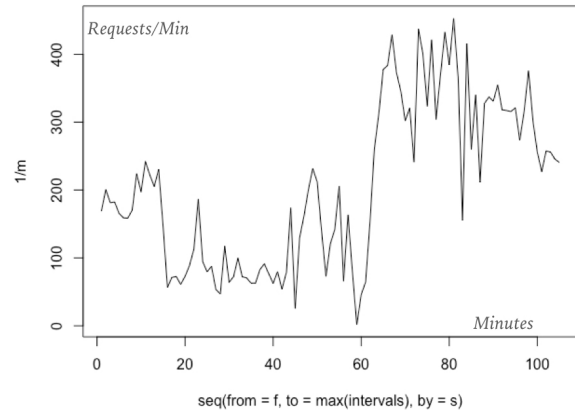
### DAY-22



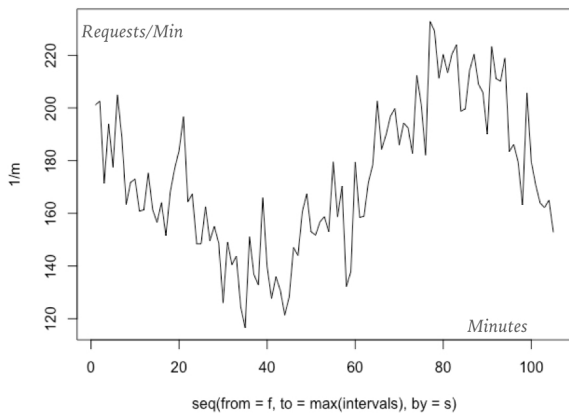
### DAY-20



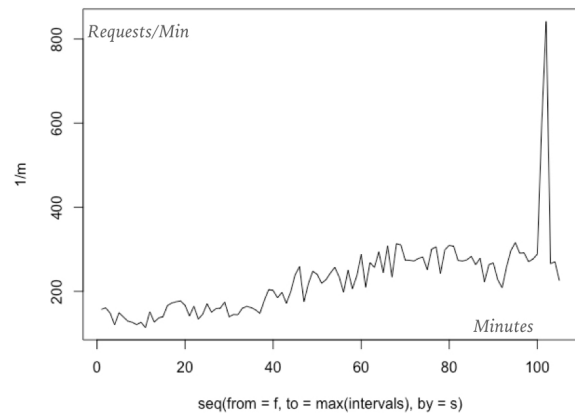
### DAY-23



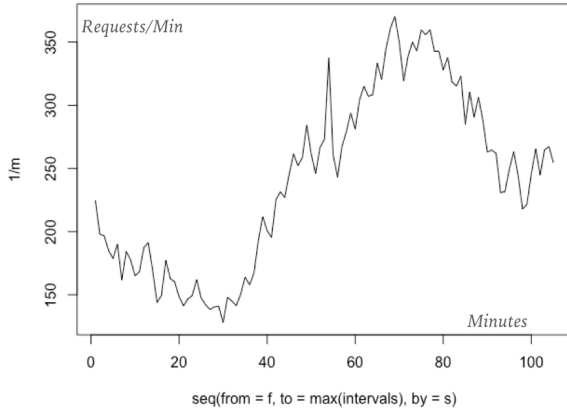
### DAY-21



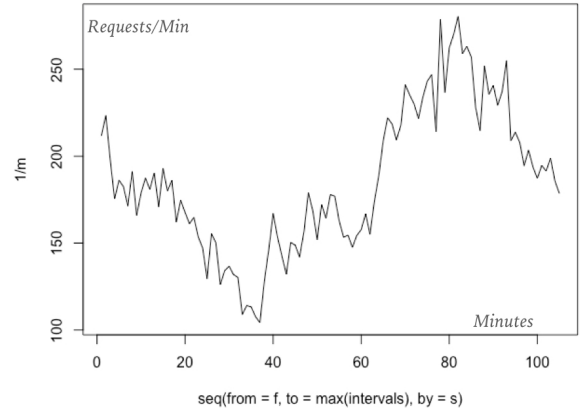
### DAY-24



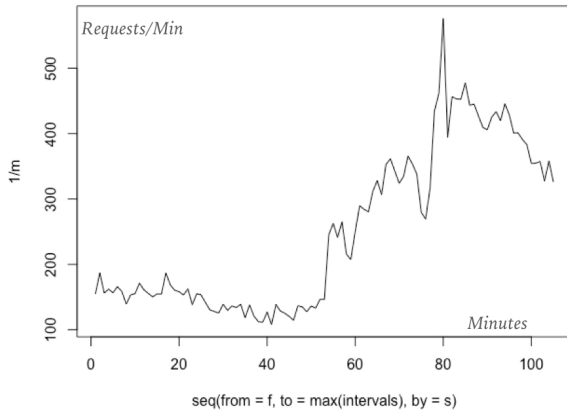
### DAY-25



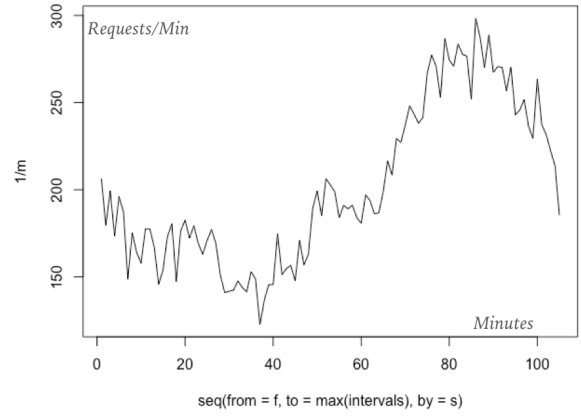
### DAY-28



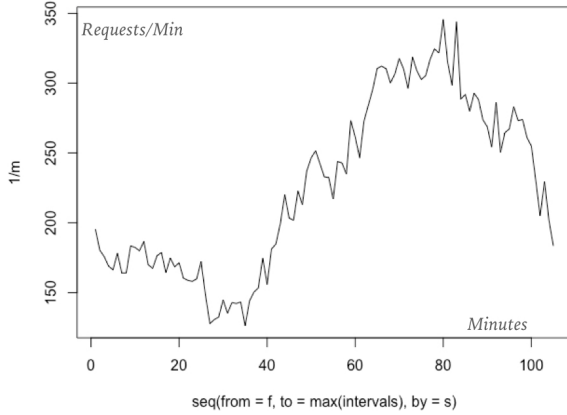
### DAY-26



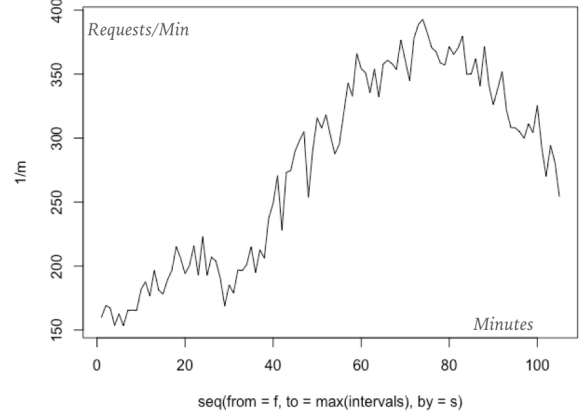
### DAY-29



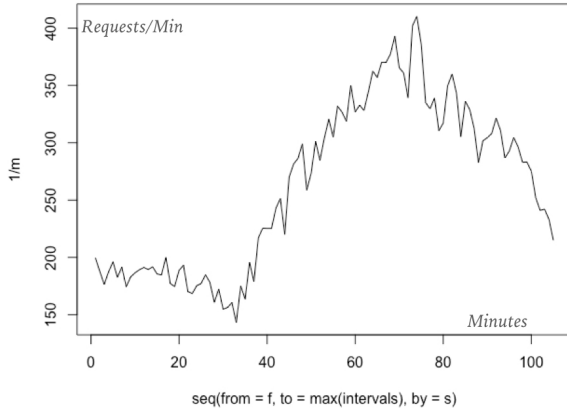
### DAY-27



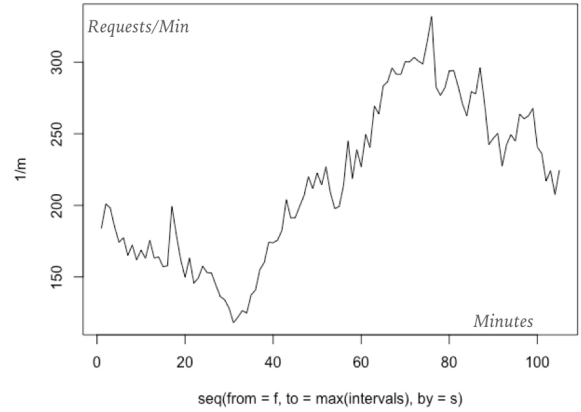
### DAY-30



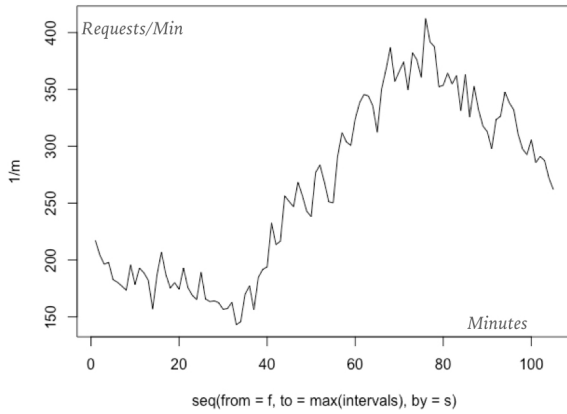
### DAY-31



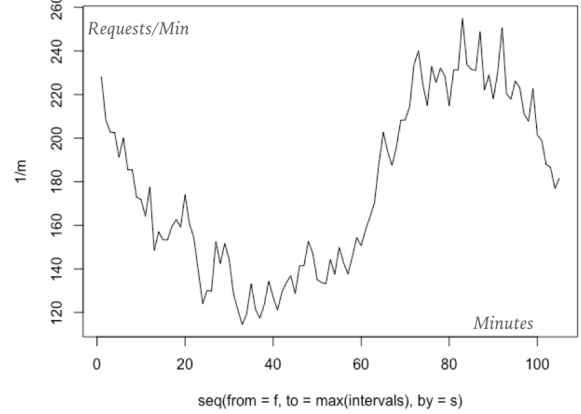
### DAY-34



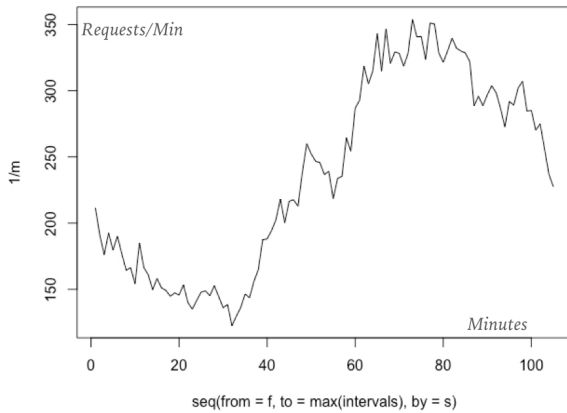
### DAY-32



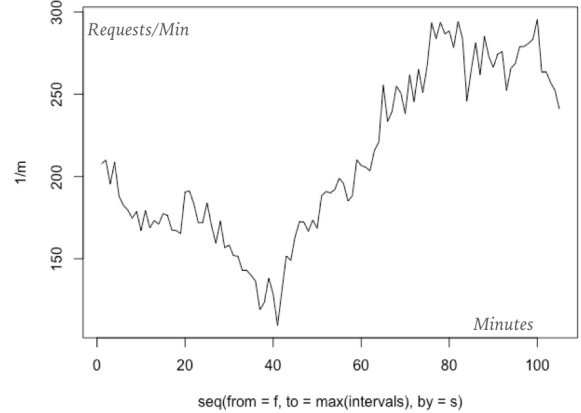
### DAY-35



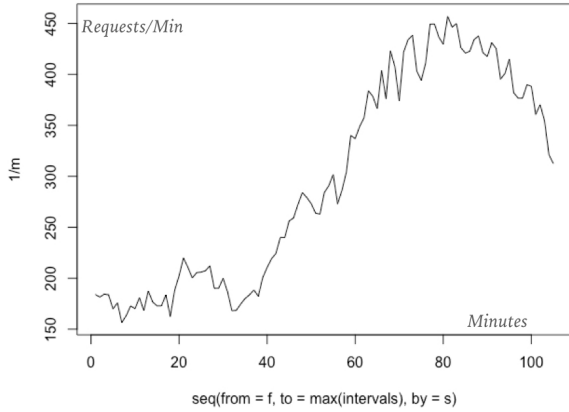
### DAY-33



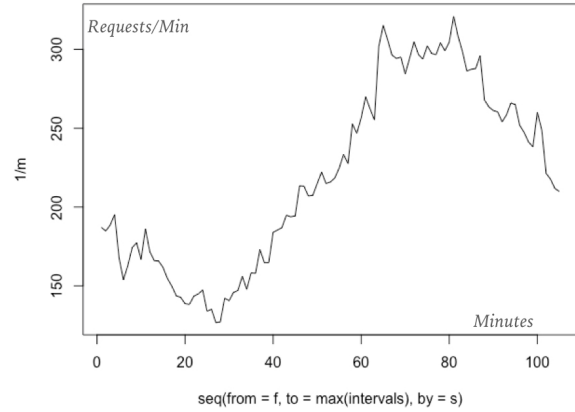
### DAY-36



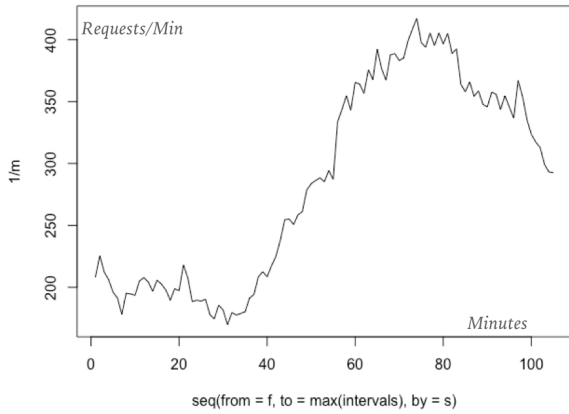
### DAY-37



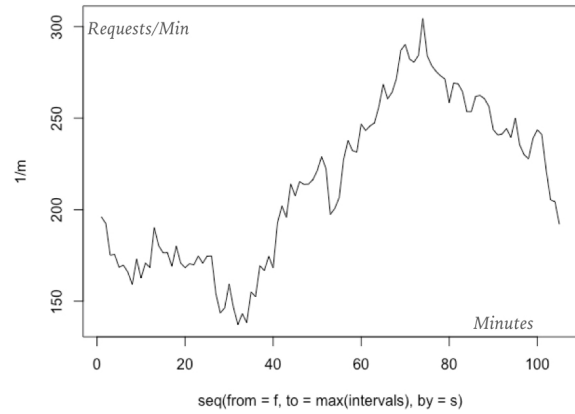
### DAY-40



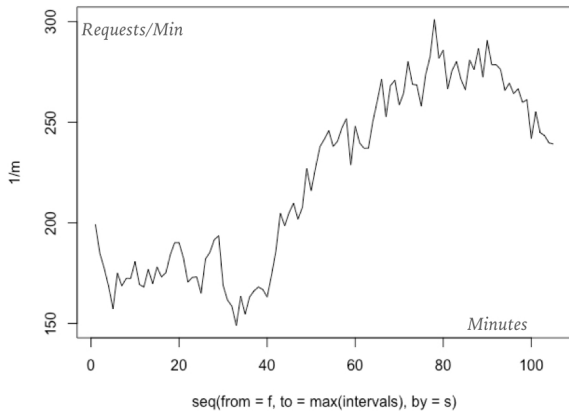
### DAY-38



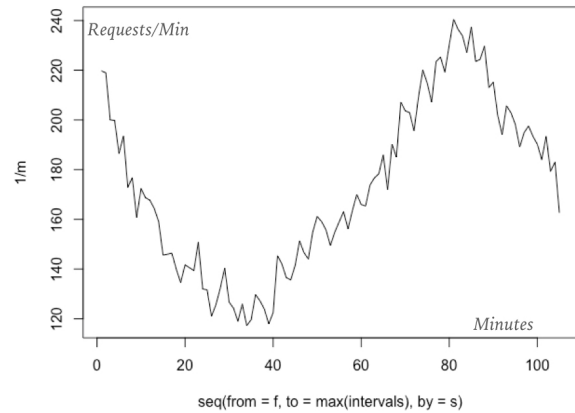
### DAY-41



### DAY-39

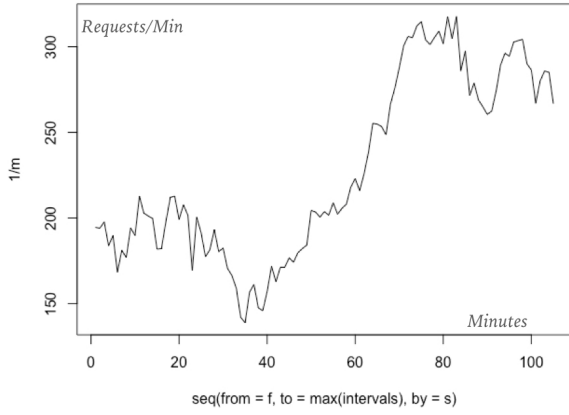


### DAY-42

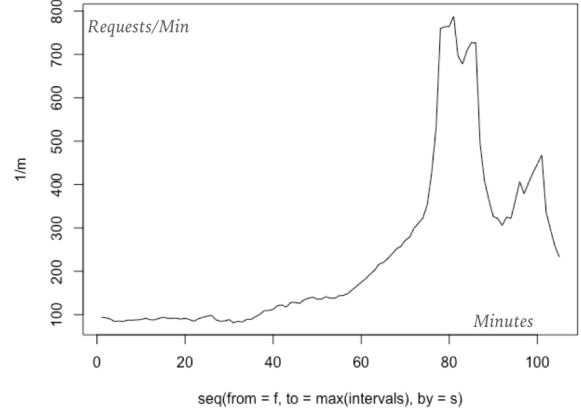




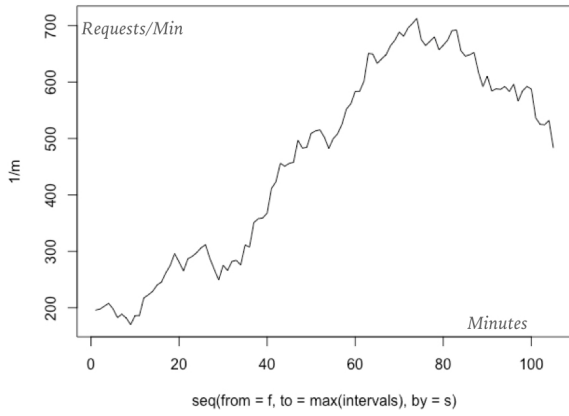
### DAY-43



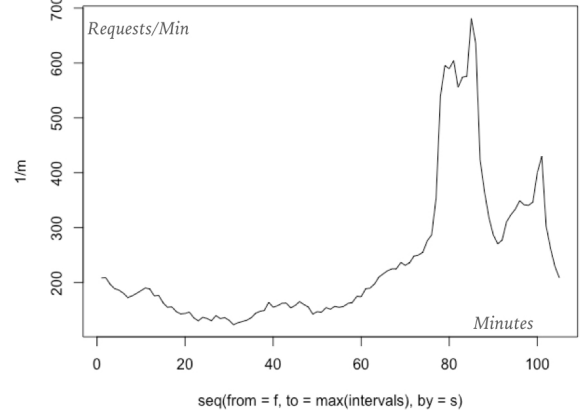
### DAY-46



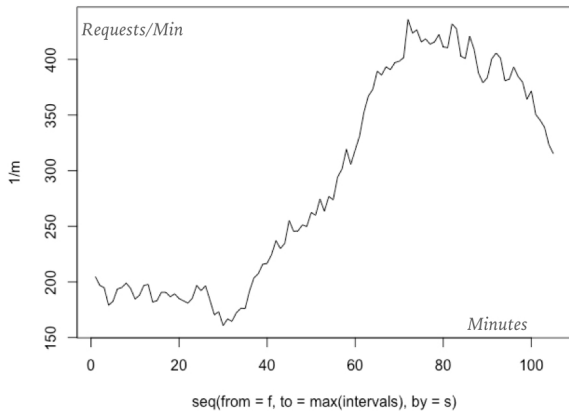
### DAY-44



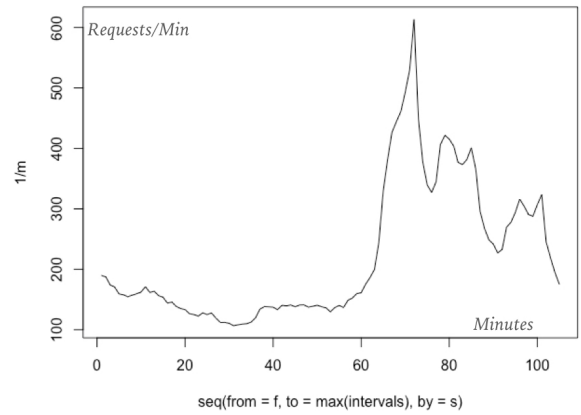
### DAY-47



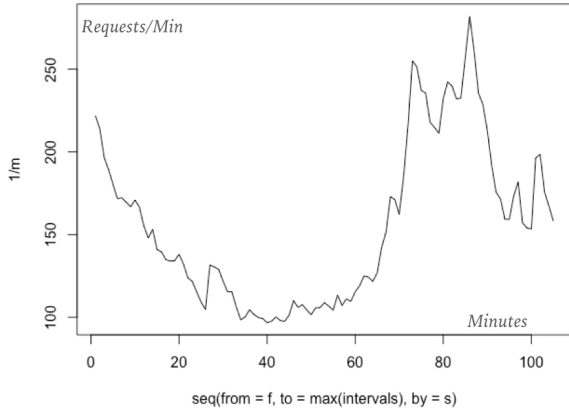
### DAY-45



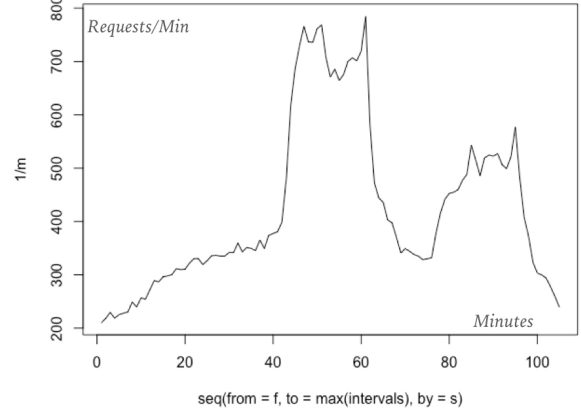
### DAY-48



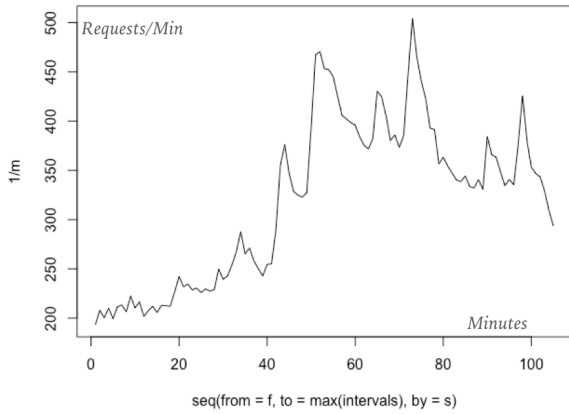
**DAY-49**



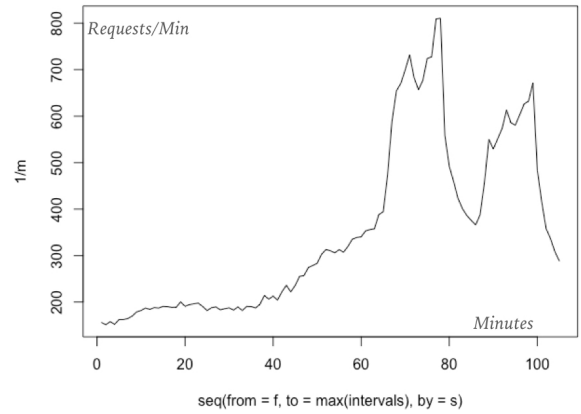
**DAY-52**



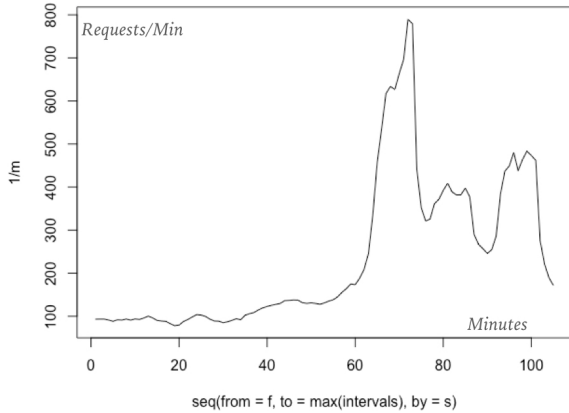
**DAY-50**



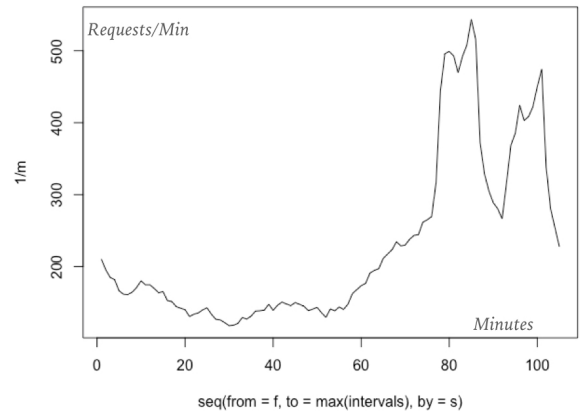
**DAY-53**



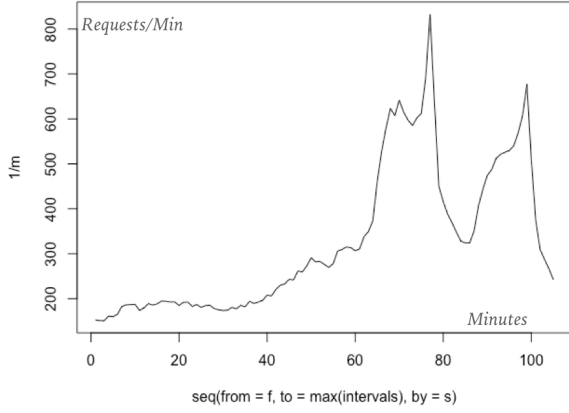
**DAY-51**



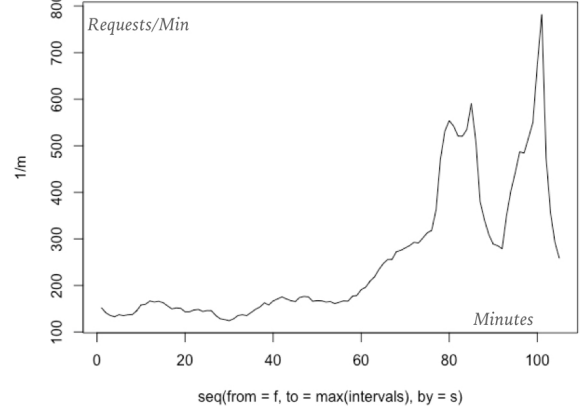
**DAY-54**



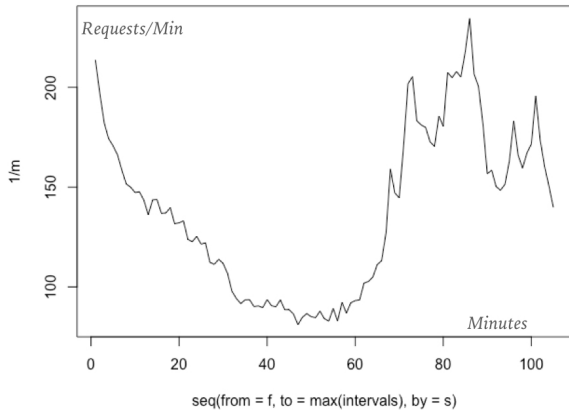
**DAY-55**



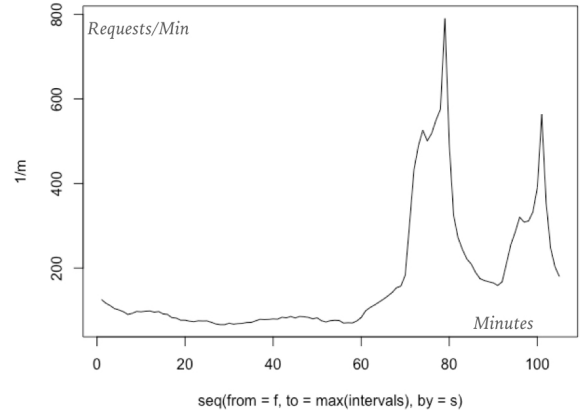
**DAY-58**



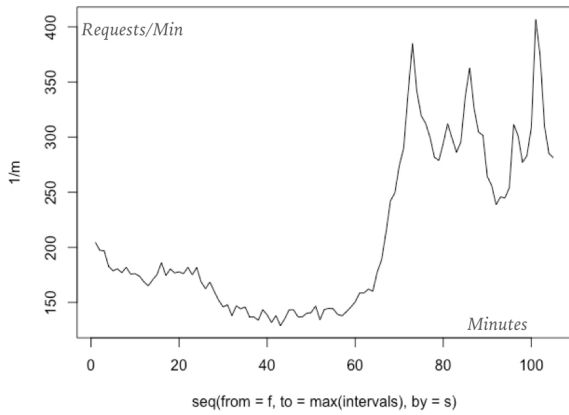
**DAY-56**



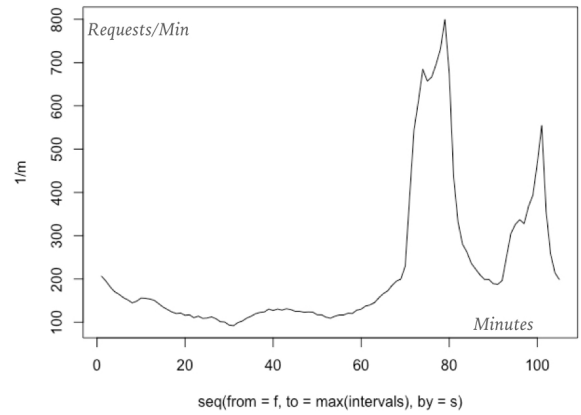
**DAY-59**



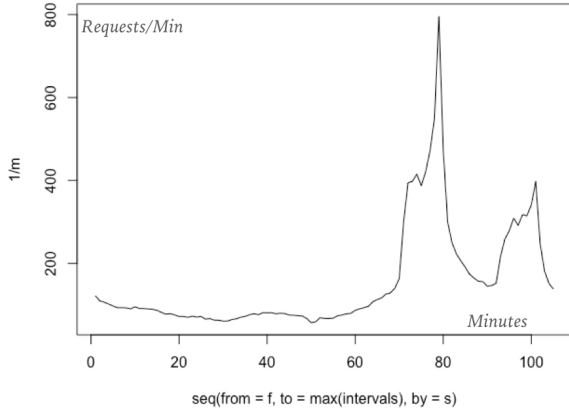
**DAY-57**



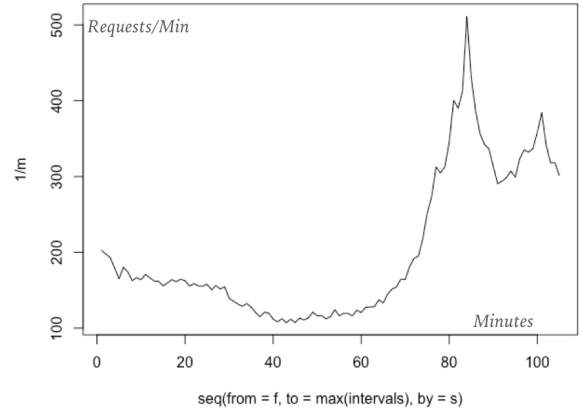
**DAY-60**



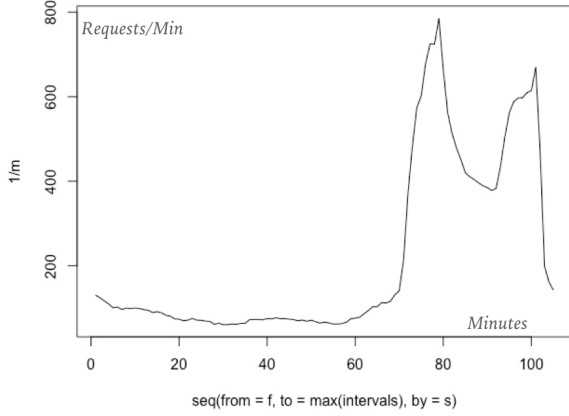
**DAY-61**



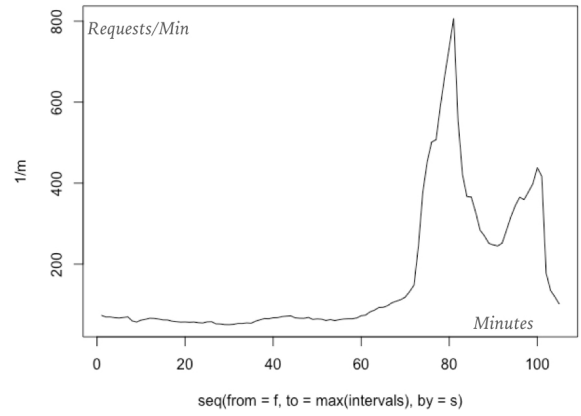
**DAY-64**



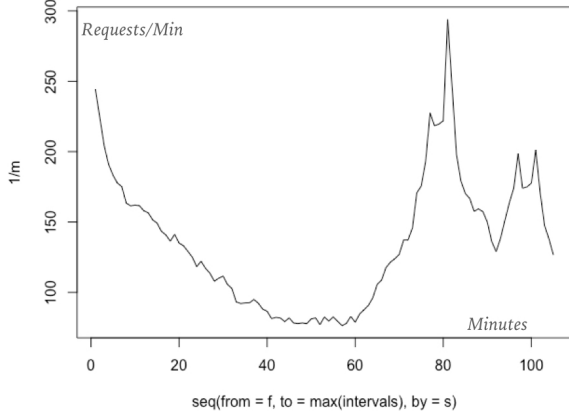
**DAY-62**



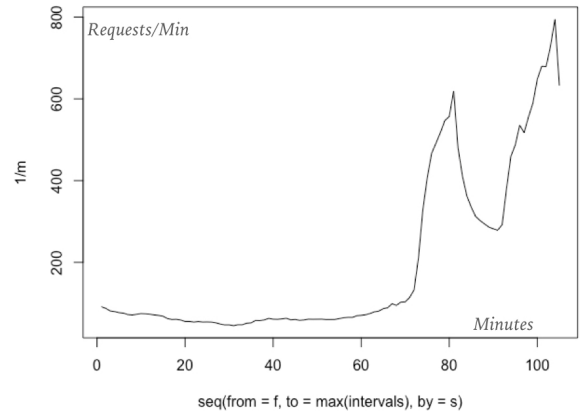
**DAY-65**



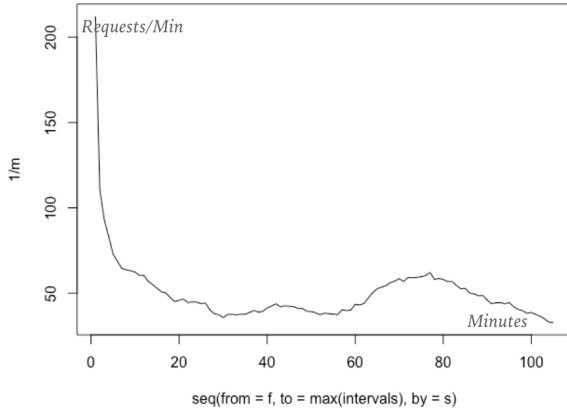
**DAY-63**



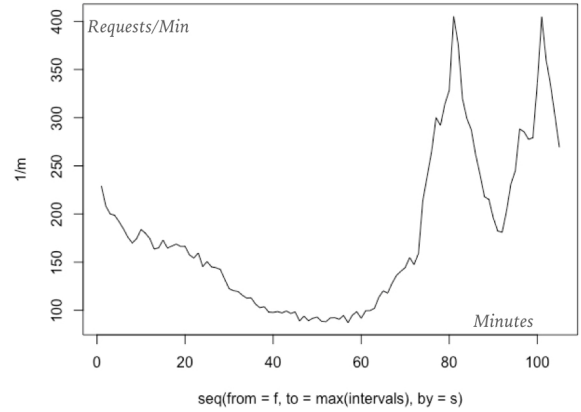
**DAY-66**



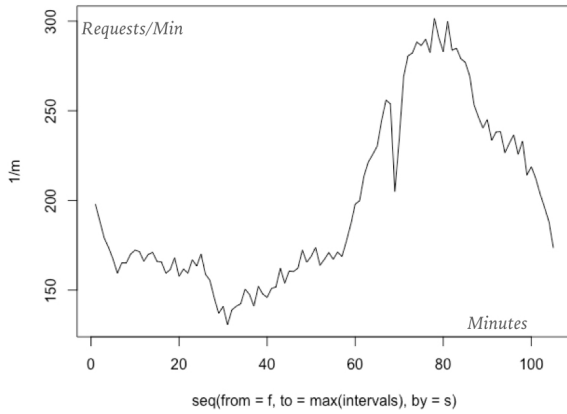
### DAY-67



### DAY-70



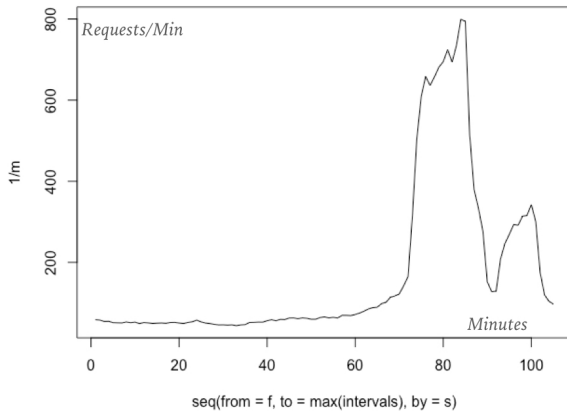
### DAY-68



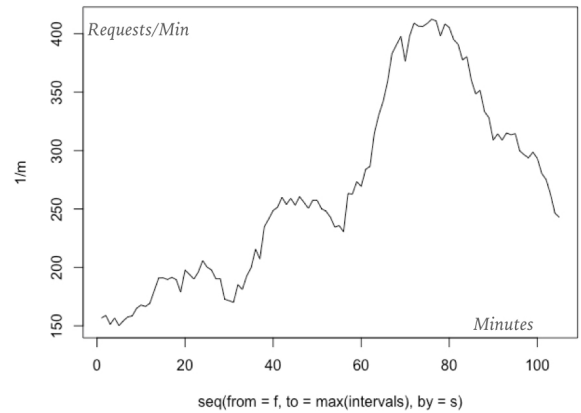
### DAY-71



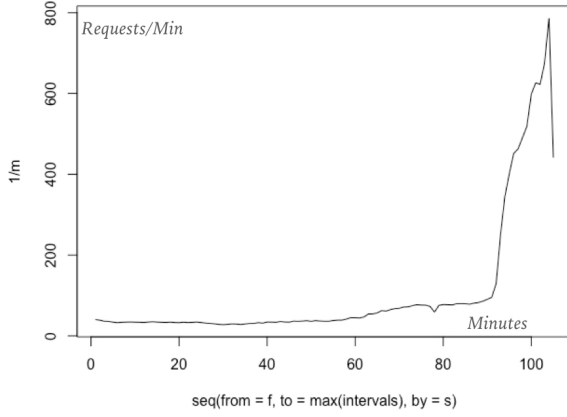
### DAY-69



### DAY-72



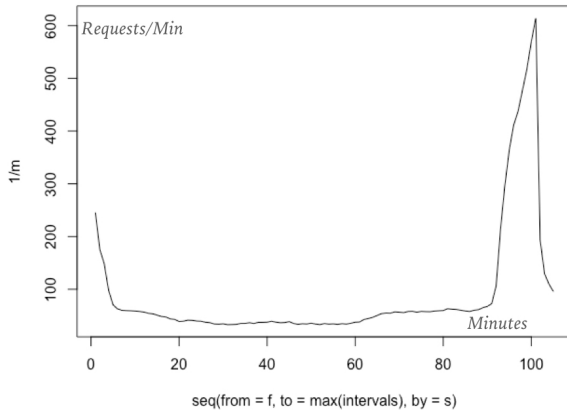
### DAY-73



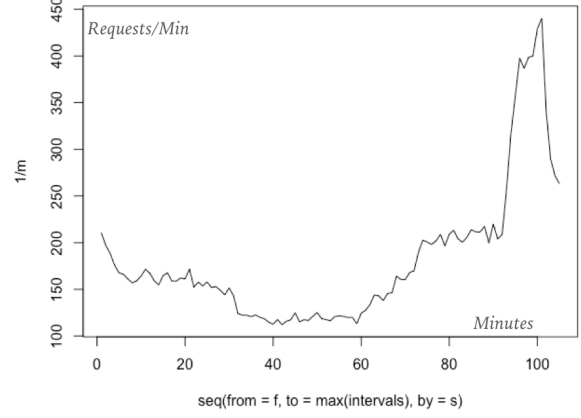
### DAY-76



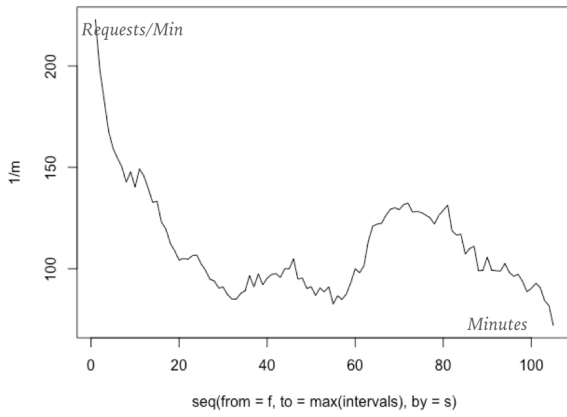
### DAY-74



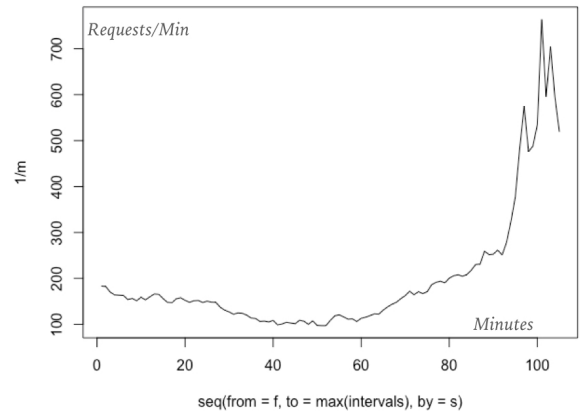
### DAY-77



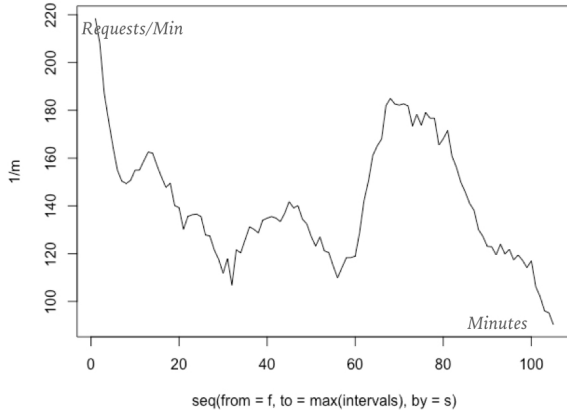
### DAY-75



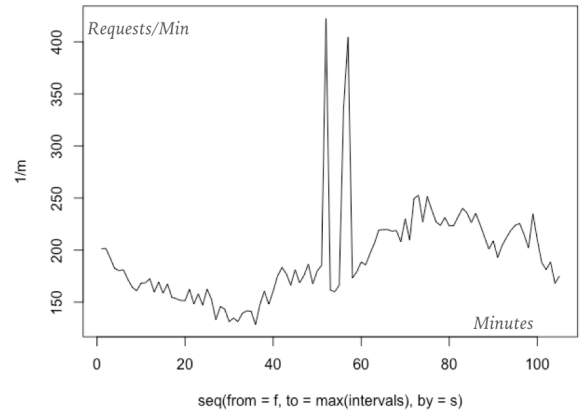
### DAY-78



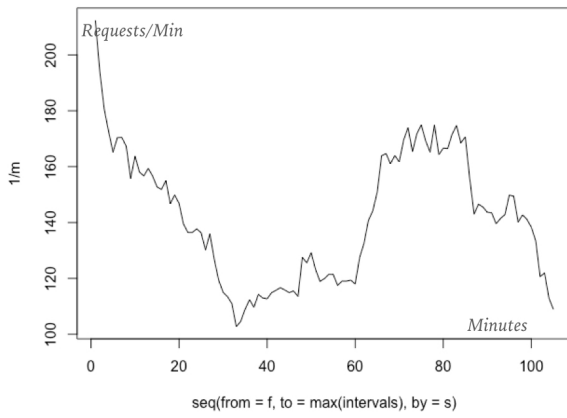
### DAY-79



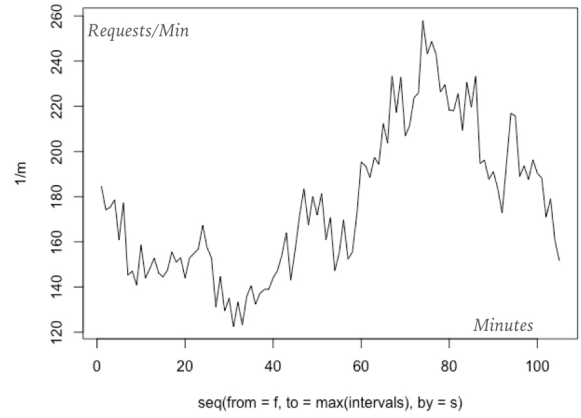
### DAY-82



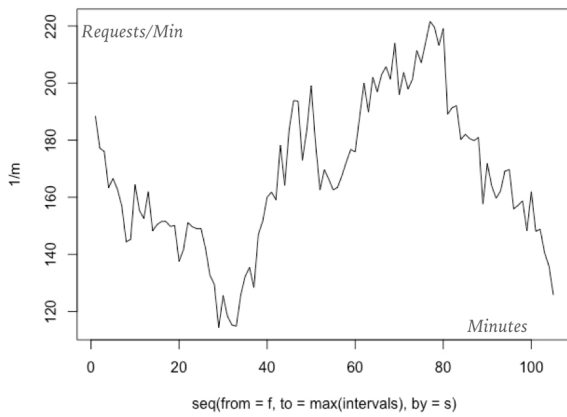
### DAY-80



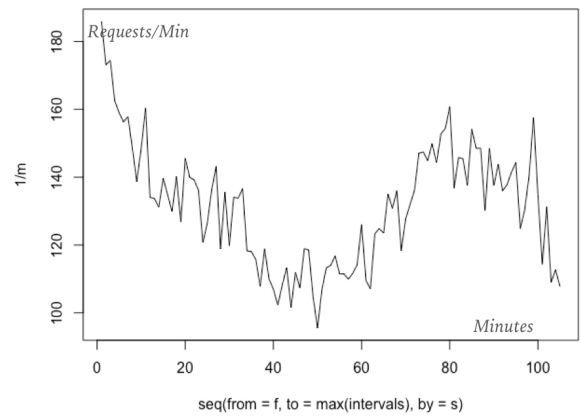
### DAY-83



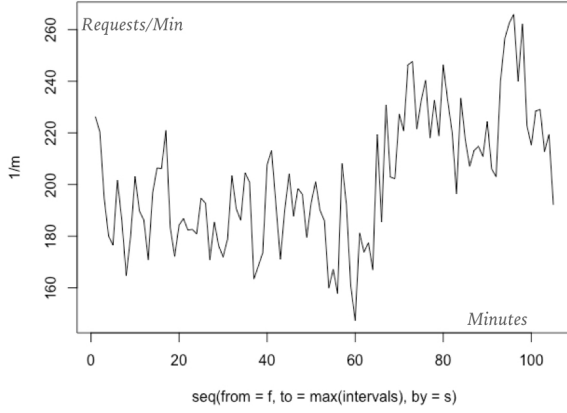
### DAY-81



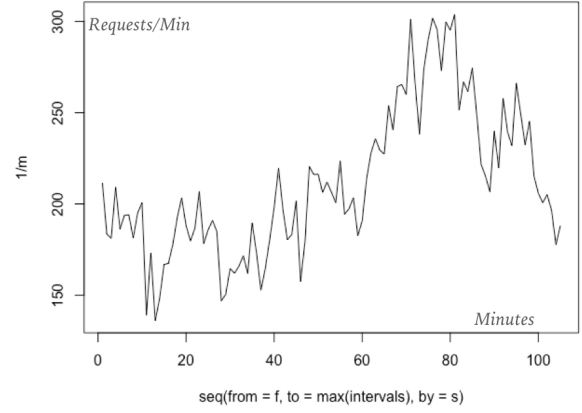
### DAY-84



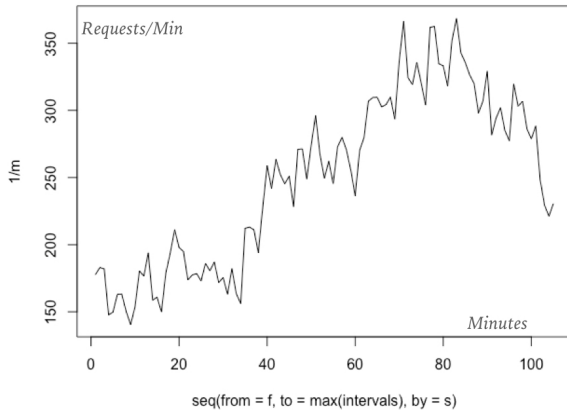
### DAY-85



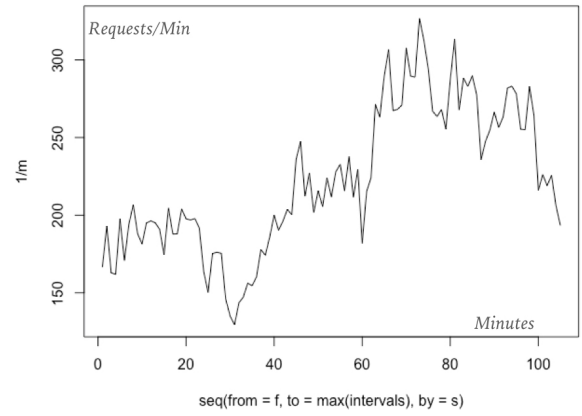
### DAY-88



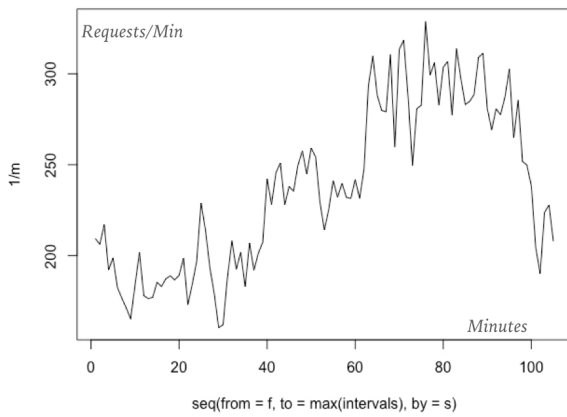
### DAY-86



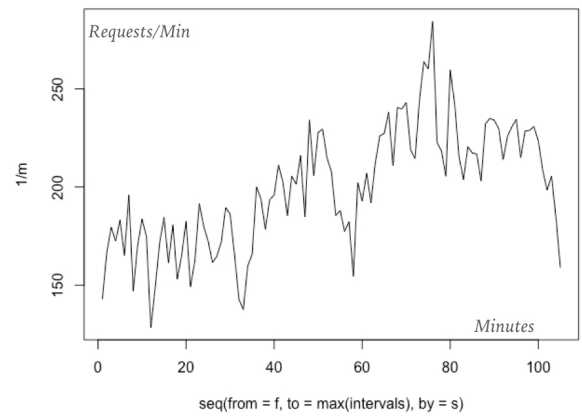
### DAY-89



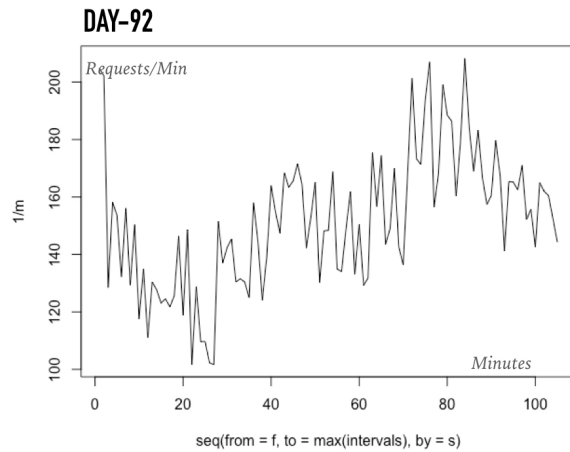
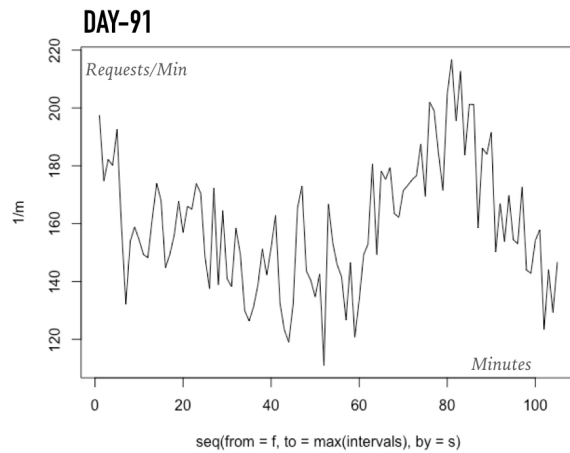
### DAY-87



### DAY-90







- [1] A. Pandey, G. A. Moreno, J. Cámara, and D. Garlan, "Hybrid planning for decision making in self-adaptive systems," in *10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*. IEEE, 2016, pp. 130–139.
- [2] M. J. Veth, "Advanced formation flight control," Air Force Institute of Technology, Tech. Rep., 1994.
- [3] A. Symington, S. Waharte, S. Julier, and N. Trigoni, "Probabilistic target detection by camera-equipped UAVs," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, May 2010, pp. 4076–4081. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5509355>