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

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 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 π, protected from race conditions by a mutex μ) using a combination of some reactive planning approach ($\rho_r \in \mathcal{F}$) and deliberative planning. A *plan* is a partial function $P : S \nrightarrow A$, 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 (ξ) 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 π
3: **global** $\mathcal{T} \leftarrow$ *new* Thread ▷ Deliberative thread
4: **function** HybridPlanning(Problem ξ, Planners $\mathcal{F}$, Planner ρ_d)
5: μ.lock()
6: **if** $\pi \neq$ *null* **and** π.has($\xi.s_{curr}$) **then**
7: μ.unlock()
8: **return** ▷ Replan only if needed
9:
10: ▷ Pick an appropriate reactive planning approach
11: Planner ρ_r = PickReactivePlanning(ξ, $\mathcal{F}$, ρ_d)
12: $\pi \leftarrow \rho_r$.Plan(ξ) ▷ Determine the reactive plan
13: μ.unlock()
14:
15: **if not** $\mathcal{T}$.isRunning() **then**
16: $\mathcal{T}$.run [▷ Deliberate in the background
17: $\pi' \leftarrow \rho_d$.Plan(ξ)
18: μ.lock()
19: $\pi \leftarrow \pi'$
20: μ.unlock()]

To find a plan corresponding to the problem ξ, the HybridPlanning 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 π. 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 PlnSel. This function can be implemented by checking predefined conditions on ξ, 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 π 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 π, μ, and $\mathcal{T}$ to configure HYBRIDPLANNING. For example, to find a reactive plan without stopping deliberative planning, the framework can lock μ, set π 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 ξ), 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 ρ_{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.,ρ_{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., ρ_{det}) varies between 25K to 100K, for deliberative planning (i.e., ρ_{mdp}) between 1.6 million and 2.8 million. The planning time for reactive planning ρ_{det} is considered negligible i.e., less than a second. The planning time for deliberative planning ρ_{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 (ϕ = 0), and tight (ϕ = 1). The latter reduces the probability of being destroyed by a factor of ψ [2]. When the team uses (E = 1) electronic countermeasures (ECM), the probability of being destroyed is reduced by a factor of α, Taking altitude, formation, and the use of ECM into account, the probability of the team being destroyed, d is given by (1).

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

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 β. The probability g of detecting a target is given by (2).

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

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 σ 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., ρ_{mdps}) and deliberative planning (i.e., ρ_{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., ψ) 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 (μ =) 0.2 and for detecting a target is (λ =) 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., ρ_{mdps}) varies between 75K to 250K, and for the long-horizon MDP planning ρ_{mdpl} between 3 million to 5 million. The planning time for reactive planning ρ_{mdps} is considered negligible i.e., less than a second. The planning time for deliberative planning ρ_{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., ρ_{det} or ρ_{mdps}) is not effective, the hybrid planner can choose another reactive approach (e.g., ρ_{wait}) in $\mathcal{F}$; therefore, performance of HP is not tightly linked to a reactive approach.

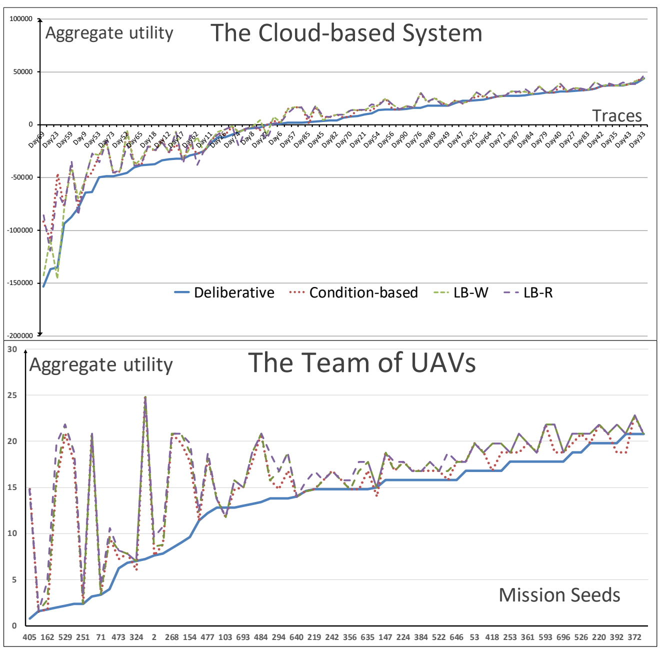


Figure 1. Performance of the hybrid planning modes improves with the performance of deliberative planning mode.

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 ρ_{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.

```
mdp

const double addServer_LATENCY = 120;
const int HORIZON = 5;
const double PERIOD = 60;
const int DIMMER_LEVELS = 3;
const int ini_dimmer = 1;
const int MAX_SERVERS_A = 1;
const int MAX_SERVERS_B = 1;
const int MAX_SERVERS_C = 1;
const int ini_servers_A = 1;
const int ini_servers_B = 0;
const int ini_servers_C = 0;
const int ini_addServerA_state = 0;
const int ini_addServerB_state = 0;
const int ini_addServerC_state = 0;
const double SERVERA_COST = 1;
const double SERVERB_COST = 0.7;
const double SERVERC_COST = 0.5;
const double MAX_ARRIVALA_CAPACITY = 200;
const double MAX_ARRIVALA_CAPACITY_LOW = 400;
const double MAX_ARRIVALB_CAPACITY = 140;
const double MAX_ARRIVALB_CAPACITY_LOW = 280;
const double MAX_ARRIVALC_CAPACITY = 100;
const double MAX_ARRIVALC_CAPACITY_LOW = 200;
const double penalty = −0.25;
const int ini_traffic_A = 4;
const int ini_traffic_B = 0;
const int ini_traffic_C = 0;
const double interArrivalScaleFactorForDecision = 1; // 1 has no effect

// The request arrival rate remains constant at the current value
formula stateValue = 0.0163126;


module clk
    time : [0..HORIZON + 1] init 0;
    readyToTick : bool init true;
    [tick] readyToTick & time < HORIZON + 1 −> 1 : (time' = time + 1) &
            (readyToTick'=false);
    [tack] !readyToTick −> 1 : (readyToTick'=true);
endmodule

label "final" = time = HORIZON + 1;
formula sys_go = readyToTick;

module controller
    active_servers_A : [0..MAX_SERVERS_A] init ini_servers_A;
    active_servers_B : [0..MAX_SERVERS_B] init ini_servers_B;
    active_servers_C : [0..MAX_SERVERS_C] init ini_servers_C;

    dimmer : [1..DIMMER_LEVELS] init ini_dimmer;

    traffic_A : [0..4] init ini_traffic_A;
    traffic_B : [0..4] init ini_traffic_B;
    traffic_C : [0..4] init ini_traffic_C;

    [addServerA_complete] active_servers_A < MAX_SERVERS_A −> 1 :
            (active_servers_A' = active_servers_A + 1);
    [addServerB_complete] active_servers_B < MAX_SERVERS_B −> 1 :
            (active_servers_B' = active_servers_B + 1);
    [addServerC_complete] active_servers_C < MAX_SERVERS_C −> 1 :
            (active_servers_C' = active_servers_C + 1);

    [removeServerA_start] active_servers_A > 0 −> 1 : (active_servers_A' =
            active_servers_A − 1);
    [removeServerB_start] active_servers_B > 0 −> 1 : (active_servers_B' =
            active_servers_B − 1);
    [removeServerC_start] active_servers_C > 0 −> 1 : (active_servers_C'
            = active_servers_C − 1);

    [increaseDimmer_start] dimmer < DIMMER_LEVELS −> 1 : (dimmer' =
            dimmer + 1);
    [decreaseDimmer_start] dimmer > 1 −> 1 : (dimmer' = dimmer − 1);

    //A−B−C
    //Possible values 0−25−50−75−100

    // 100−0−0
    [divert_100_0_0] active_servers_A > 0
        −> 1 : (traffic_A' = 4) & (traffic_B' = 0) & (traffic_C' = 0);
    // 75−25−0
    [divert_75_25_0] active_servers_A > 0 & active_servers_B > 0
        −> 1 : (traffic_A' = 3) & (traffic_B' = 1) & (traffic_C' = 0);
    // 75−0−25
    [divert_75_0_25] active_servers_A > 0 & active_servers_C > 0
        −> 1 : (traffic_A' = 3) & (traffic_B' = 0) & (traffic_C' = 1);
    // 50−50−0
    [divert_50_50_0] active_servers_A > 0 & active_servers_B > 0
        −> 1 : (traffic_A' = 2) & (traffic_B' = 2) & (traffic_C' = 0);
    // 50−0−50
    [divert_50_0_50] active_servers_A > 0 & active_servers_C > 0
        −> 1 : (traffic_A' = 2) & (traffic_B' = 0) & (traffic_C' = 2);
    // 50−25−25
    [divert_50_25_25] active_servers_A > 0 & active_servers_B > 0 &
            active_servers_C > 0
        −> 1 : (traffic_A' = 2) & (traffic_B' = 1) & (traffic_C' = 1);
    // 25−75−0
    [divert_25_75_0] active_servers_A > 0 & active_servers_B > 0
        −> 1 : (traffic_A' = 1) & (traffic_B' = 3) & (traffic_C' = 0);
    // 25−0−75
    [divert_25_0_75] active_servers_A > 0 & active_servers_C > 0
        −> 1 : (traffic_A' = 1) & (traffic_B' = 0) & (traffic_C' = 3);
    // 25−50−25
    [divert_25_50_25] active_servers_A > 0 & active_servers_B > 0 &
            active_servers_C > 0
        −> 1 : (traffic_A' = 1) & (traffic_B' = 2) & (traffic_C' = 1);
    // 25−25−50
    [divert_25_25_50] active_servers_A > 0 & active_servers_B > 0 &
            active_servers_C > 0
        −> 1 : (traffic_A' = 1) & (traffic_B' = 1) & (traffic_C' = 2);
    // 0−100−0
    [divert_0_100_0] active_servers_B > 0
        −> 1 : (traffic_A' = 0) & (traffic_B' = 4) & (traffic_C' = 0);
    // 0−0−100
    [divert_0_0_100] active_servers_C > 0
        −> 1 : (traffic_A' = 0) & (traffic_B' = 0) & (traffic_C' = 4);
    // 0−75−25
    [divert_0_75_25] active_servers_B > 0 & active_servers_C > 0
        −> 1 : (traffic_A' = 0) & (traffic_B' = 3) & (traffic_C' = 1);
    // 0−25−75
    [divert_0_25_75] active_servers_B > 0 & active_servers_C > 0
        −> 1 : (traffic_A' = 0) & (traffic_B' = 1) & (traffic_C' = 3);
    // 0−50−50
    [divert_0_50_50] active_servers_B > 0 & active_servers_C > 0
        −> 1 : (traffic_A' = 0) & (traffic_B' = 2) & (traffic_C' = 2);
endmodule


formula addServerA_applicable = active_servers_A < MAX_SERVERS_A
        & !removeServer_used
                & addServerB_state = 0 & addServerC_state = 0;
formula addServerB_applicable = active_servers_B < MAX_SERVERS_B
        & !removeServer_used
```

```
                    & addServerA_state = 0 & addServerC_state = 0;
formula addServerC_applicable = active_servers_C < MAX_SERVERS_C & !removeServer_used
                    & addServerA_state = 0 & addServerB_state = 0;

formula removeServerA_applicable = active_servers_A > 0 & addServerA_state = 0 & active_servers > 1
                    & addServerB_state = 0 & addServerC_state = 0;
formula removeServerB_applicable = active_servers_B > 0 & addServerB_state = 0 & active_servers > 1
                    & addServerA_state = 0 & addServerC_state = 0;
formula removeServerC_applicable = active_servers_C > 0 & addServerC_state = 0 & active_servers > 1
                    & addServerA_state = 0 & addServerB_state = 0;

formula increaseDimmer_compatible = !decreaseDimmer_used;
formula decreaseDimmer_compatible = !increaseDimmer_used;
formula increase_dimmer_applicable = dimmer < DIMMER_LEVELS & increaseDimmer_compatible;
formula decrease_dimmer_applicable = dimmer > 1 & decreaseDimmer_compatible;

const int addServer_LATENCY_PERIODS = ceil(addServer_LATENCY / PERIOD);

// This remove server constraints that only one server could be removed in one monitoring cycle.
module removeServer
    removeServer_go : bool init true;
    removeServer_used : bool init false;

    [removeServerA_start] sys_go & removeServer_go
        & removeServerA_applicable // applicability conditions
        −> (removeServer_go' = false) & (removeServer_used' = true);

    [removeServerB_start] sys_go & removeServer_go
        & removeServerB_applicable // applicability conditions
        −> (removeServer_go' = false) & (removeServer_used' = true);

    [removeServerC_start] sys_go & removeServer_go
        & removeServerC_applicable // applicability conditions
        −> (removeServer_go' = false) & (removeServer_used' = true);

    // Case when remove server tactic is applicable but not used
    [pass_remove_server] sys_go & removeServer_go // can go
        −> (removeServer_go' = false);

    [tick] !removeServer_go −> 1 : (removeServer_go' = true) & (removeServer_used' = false);
endmodule

module addServer
    addServerA_state : [0..addServer_LATENCY_PERIODS] init ini_addServerA_state;
    addServerB_state : [0..addServer_LATENCY_PERIODS] init ini_addServerB_state;
    addServerC_state : [0..addServer_LATENCY_PERIODS] init ini_addServerC_state;

    addServer_go : bool init true;

    // tactic applicable, start it
    [addServerA_start] sys_go & addServer_go // can go
        & addServerA_state = 0 // tactic has not been started
        & addServerA_applicable
        −> (addServerA_state' = 1) & (addServer_go' = false);

    // tactic applicable, start it
    [addServerB_start] sys_go & addServer_go // can go
        & addServerB_state = 0 // tactic has not been started
        & addServerB_applicable
        −> (addServerB_state' = 1) & (addServer_go' = false);

    // tactic applicable, start it
    [addServerC_start] sys_go & addServer_go // can go
        & addServerC_state = 0 // tactic has not been started
        & addServerC_applicable
        −> (addServerC_state' = 1) & (addServer_go' = false);

    // tactic applicable, but don't use it
    [pass_add] sys_go & addServer_go // can go
        & addServerA_state = 0 // tactic has not been started
        & addServerB_state = 0 & addServerC_state = 0
        //& addServerA_applicable
        −> (addServer_go' = false);

    // progress of the tactic
    [progressA] sys_go & addServer_go
        & addServerA_state > 0 & addServerA_state < addServer_LATENCY_PERIODS
        −> 1 : (addServerA_state' = addServerA_state + 1) & (addServer_go' = false);

    [progressB] sys_go & addServer_go
        & addServerB_state > 0 & addServerB_state < addServer_LATENCY_PERIODS
        −> 1 : (addServerB_state' = addServerB_state + 1) & (addServer_go' = false);

    [progressC] sys_go & addServer_go
        & addServerC_state > 0 & addServerC_state < addServer_LATENCY_PERIODS
        −> 1 : (addServerC_state' = addServerC_state + 1) & (addServer_go' = false);

    // completion of the tactic
    [addServerA_complete] sys_go & addServer_go
        & addServerA_state = addServer_LATENCY_PERIODS // completed
        −> 1 : (addServerA_state' = 0) & (addServer_go' = true); // so that it can start again at this time if needed

    [addServerB_complete] sys_go & addServer_go
        & addServerB_state = addServer_LATENCY_PERIODS // completed
        −> 1 : (addServerB_state' = 0) & (addServer_go' = true); // so that it can start again at this time if needed

    [addServerC_complete] sys_go & addServer_go
        & addServerC_state = addServer_LATENCY_PERIODS // completed
        −> 1 : (addServerC_state' = 0) & (addServer_go' = true); // so that it can start again at this time if needed

    [tick] !addServer_go −> 1 : (addServer_go' = true);
endmodule

// Make sure that divert traffic is executed at the end i.e.after adding or removing the servers.
formula divert_traffic_applicable = divert_go & !addServer_go & !removeServer_go & !increaseDimmer_go & !decreaseDimmer_go;

module divert_traffic
    divert_go : bool init true;

    //A−B−C
    //Possible values 0−25−50−75−100

    // 100−0−0
    [divert_100_0_0] sys_go & divert_traffic_applicable
        −> 1 : (divert_go'=false);
    // 75−25−0
    [divert_75_25_0] sys_go & divert_traffic_applicable
        −> 1 : (divert_go'=false);
    // 75−0−25
    [divert_75_0_25] sys_go & divert_traffic_applicable
        −> 1 : (divert_go'=false);
    // 50−50−0
    [divert_50_50_0] sys_go & divert_traffic_applicable
        −> 1 : (divert_go'=false);
    // 50−0−50
    [divert_50_0_50] sys_go & divert_traffic_applicable
        −> 1 : (divert_go'=false);
    // 50−25−25
    [divert_50_25_25] sys_go & divert_traffic_applicable
        −> 1 : (divert_go'=false);
    // 25−75−0
    [divert_25_75_0] sys_go & divert_traffic_applicable
        −> 1 : (divert_go'=false);
    // 25−0−75
    [divert_25_0_75] sys_go & divert_traffic_applicable
        −> 1 : (divert_go'=false);
    // 25−50−25
    [divert_25_50_25] sys_go & divert_traffic_applicable
        −> 1 : (divert_go'=false);
    // 25−25−50
```

```
    [divert_25_25_50] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 0-100-0
    [divert_0_100_0] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 0-0-100
    [divert_0_0_100] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 0-75-25
    [divert_0_75_25] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 0-25-75
    [divert_0_25_75] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 0-50-50
    [divert_0_50_50] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);

    [tick] !divert_go -> 1 : (divert_go' = true);
endmodule

module increaseDimmer
    increaseDimmer_go : bool init true;
    increaseDimmer_used : bool init false;

    [increaseDimmer_start] sys_go & increaseDimmer_go
        & increase_dimmer_applicable // applicability conditions
        -> (increaseDimmer_go' = false) & (increaseDimmer_used' = true);

    // tactic applicable but not used
    [pass_inc_dimmer] sys_go & increaseDimmer_go // can go
        -> (increaseDimmer_go' = false);

    [tick] !increaseDimmer_go -> 1 : (increaseDimmer_go' = true) &
            (increaseDimmer_used' = false);
endmodule

// tactic
module decreaseDimmer
    decreaseDimmer_go : bool init true;
    decreaseDimmer_used : bool init false;

    [decreaseDimmer_start] sys_go & decreaseDimmer_go
        & decrease_dimmer_applicable // applicability conditions
        -> (decreaseDimmer_go' = false) & (decreaseDimmer_used' =
            true);

    // tactic applicable but not used
    [pass_dec_dimmer] sys_go & decreaseDimmer_go // can go
        -> (decreaseDimmer_go' = false);

    [tick] !decreaseDimmer_go -> 1 : (decreaseDimmer_go' = true) &
            (decreaseDimmer_used' = false);
endmodule

//***************************************************************************
// Queuing network with each server having queueing model of M/G/1/PS
//***************************************************************************
formula dimmerFactor = (dimmer - 1) / (DIMMER_LEVELS - 1);
formula interarrivalMean = stateValue * interArrivalScaleFactorForDecision;

formula Pa = (traffic_A * 25)/100;
formula Pb = (traffic_B * 25)/100;
formula Pc = (traffic_C * 25)/100;

formula loaded_servers = (Pa != 0 ? 1 : 0) + (Pb != 0 ? 1 : 0) + (Pc != 0 ? 1 :
        0);

formula service_rate_A = dimmerFactor *
        (MAX_ARRIVALA_CAPACITY_LOW)
            + (1 - dimmerFactor) * (MAX_ARRIVALA_CAPACITY);
formula service_rate_B = dimmerFactor *
        (MAX_ARRIVALB_CAPACITY_LOW)
            + (1 - dimmerFactor) * (MAX_ARRIVALB_CAPACITY);
formula service_rate_C = dimmerFactor *
        (MAX_ARRIVALC_CAPACITY_LOW)
            + (1 - dimmerFactor) * (MAX_ARRIVALC_CAPACITY);

formula rhoA = Pa/(service_rate_A*interarrivalMean);
formula rhoB = Pb/(service_rate_B*interarrivalMean);
formula rhoC = Pc/(service_rate_C*interarrivalMean);
formula overloaded = (rhoA >= 1 | rhoB >= 1 | rhoC >= 1);

formula rt_A = 1/(service_rate_A - (throughput*Pa));
formula rt_B = 1/(service_rate_B - (throughput*Pb));
formula rt_C = 1/(service_rate_C - (throughput*Pc));

// Response time to clients utility function
const double RT_THRESHOLD = 1.0;

formula expected_wait_time = (Pa*rt_A + Pb*rt_B + Pc*rt_C);
formula rt = (interarrivalMean = 0 ? 0 : (overloaded ? RT_THRESHOLD + 2
        : expected_wait_time));
const double NORMAL_A_REVENUE = (SERVERA_COST /
        MAX_ARRIVALA_CAPACITY) * 10;
const double DIMMER_A_REVENUE = (SERVERA_COST /
        MAX_ARRIVALA_CAPACITY_LOW) * 3 / 2;
const double NORMAL_B_REVENUE = (SERVERB_COST /
        MAX_ARRIVALB_CAPACITY) * 10;
const double DIMMER_B_REVENUE = (SERVERB_COST /
        MAX_ARRIVALB_CAPACITY_LOW) * 3 / 2;
const double NORMAL_C_REVENUE = (SERVERC_COST /
        MAX_ARRIVALC_CAPACITY) * 10;
const double DIMMER_C_REVENUE = (SERVERC_COST /
        MAX_ARRIVALC_CAPACITY_LOW) * 3 / 2;

const double DIMMER_REVENUE = DIMMER_A_REVENUE +
        DIMMER_B_REVENUE + DIMMER_C_REVENUE;
const double NORMAL_REVENUE = NORMAL_A_REVENUE +
        NORMAL_B_REVENUE + NORMAL_C_REVENUE;

formula serverA_cost = ((addServerA_state > 0 ? 1 : 0) +
        active_servers_A) * SERVERA_COST;
formula serverB_cost = ((addServerB_state > 0 ? 1 : 0) +
        active_servers_B) * SERVERB_COST;
formula serverC_cost = ((addServerC_state > 0 ? 1 : 0) +
        active_servers_C) * SERVERC_COST;
formula cost = serverA_cost + serverB_cost + serverC_cost;
formula throughput = 1/interarrivalMean;

formula basicUtilityA = throughput * Pa * (dimmerFactor *
        DIMMER_A_REVENUE + (1 - dimmerFactor) *
        NORMAL_A_REVENUE);
formula basicUtilityB = throughput * Pb * (dimmerFactor *
        DIMMER_B_REVENUE + (1 - dimmerFactor) *
        NORMAL_B_REVENUE);
formula basicUtilityC = throughput * Pc * (dimmerFactor *
        DIMMER_C_REVENUE + (1 - dimmerFactor) *
        NORMAL_C_REVENUE);

formula basicUtility = basicUtilityA + basicUtilityB + basicUtilityC;
formula active_servers = active_servers_A + active_servers_B +
        active_servers_C;
formula poweredServers = (addServerA_state > 0 ? 1 : 0) +
        (addServerB_state > 0 ? 1 : 0) + (addServerC_state > 0 ? 1 : 0 )
       + active_servers;
formula MAX_SERVERS = MAX_SERVERS_A + MAX_SERVERS_B +
        MAX_SERVERS_C;

formula MAX_SERVER_COST = MAX_SERVERS_A * SERVERA_COST
            + MAX_SERVERS_B * SERVERB_COST
            + MAX_SERVERS_C * SERVERC_COST;

formula netPenalty = stateValue > 0 ? penalty / stateValue : 0;

formula uTotal = (overloaded & (poweredServers < MAX_SERVERS |
        dimmer < DIMMER_LEVELS | active_servers != loaded_servers))
                ? -(1000) // avoid unstable solutions
                : (((rt > RT_THRESHOLD | rt <= 0) ? netPenalty :
                    basicUtility) - cost);

rewards "util"
    // 100000000.0 is added to avoid a negative value during calculation;
        negative utility is not supported by PRISM.
    [tack] true : 100000000.0 + (PERIOD)*(uTotal);
endrewards
```

Listing 1. PRISM specification for deterministic planning

MDP Planning Specification: In contrast to deterministic planning (i.e., ρ_{det}), MDP planning (i.e., ρ_{mdp}) considers predicted (uncertain) values of request arrival rate. For ρ_{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., ρ_{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.

```
mdp

const double addServer_LATENCY = 120;
const int HORIZON = 5;
const double PERIOD = 60;
const int DIMMER_LEVELS = 3;
const int ini_dimmer = 1;
const int MAX_SERVERS_A = 1;
const int MAX_SERVERS_B = 1;
const int MAX_SERVERS_C = 1;
const int ini_servers_A = 1;
const int ini_servers_B = 0;
const int ini_servers_C = 0;
const int ini_addServerA_state = 0;
const int ini_addServerB_state = 0;
const int ini_addServerC_state = 0;
const double SERVERA_COST = 1;
const double SERVERB_COST = 0.7;
const double SERVERC_COST = 0.5;
const double MAX_ARRIVALA_CAPACITY = 200;
const double MAX_ARRIVALA_CAPACITY_LOW = 400;
const double MAX_ARRIVALB_CAPACITY = 140;
const double MAX_ARRIVALB_CAPACITY_LOW = 280;
const double MAX_ARRIVALC_CAPACITY = 100;
const double MAX_ARRIVALC_CAPACITY_LOW = 200;
const double penalty = −0.25;
const int ini_traffic_A = 4;
const int ini_traffic_B = 0;
const int ini_traffic_C = 0;
const double interArrivalScaleFactorForDecision = 1; // 1 has no effect

\\ Model of the environment as an MDP. Values from the time series predictor
\\ have been used to get the interarrival time.
module environment
s : [0..201] init 0;
[tick] s = 0 ->
    0.185 : (s' = 1)
    + 0.63 : (s' = 2)
    + 0.185 : (s' = 3);
[tick] s = 3 ->
    0.185 : (s' = 4)
    + 0.63 : (s' = 5)
    + 0.185 : (s' = 6);
[tick] s = 6 ->
    0.185 : (s' = 7)
    + 0.63 : (s' = 8)
    + 0.185 : (s' = 9);
[tick] s = 9 ->
    0.185 : (s' = 10)
    + 0.63 : (s' = 11)
    + 0.185 : (s' = 12);
[tick] s = 12 ->
    1 : (s' = 13);
[tick] s = 11 ->
    1 : (s' = 14);
[tick] s = 10 ->
    1 : (s' = 15);
[tick] s = 8 ->
    0.185 : (s' = 16)
    + 0.63 : (s' = 17)
    + 0.185 : (s' = 18);
[tick] s = 18 ->
    1 : (s' = 19);
[tick] s = 17 ->
    1 : (s' = 20);
[tick] s = 16 ->
    1 : (s' = 21);
[tick] s = 7 ->
    0.185 : (s' = 22)
    + 0.63 : (s' = 23)
    + 0.185 : (s' = 24);
[tick] s = 24 ->
    1 : (s' = 25);
[tick] s = 23 ->
    1 : (s' = 26);
[tick] s = 22 ->
    1 : (s' = 27);
[tick] s = 5 ->
    0.185 : (s' = 28)
    + 0.63 : (s' = 29)
    + 0.185 : (s' = 30);
[tick] s = 30 ->
    0.185 : (s' = 31)
    + 0.63 : (s' = 32)
    + 0.185 : (s' = 33);
[tick] s = 33 ->
    1 : (s' = 34);
[tick] s = 32 ->
    1 : (s' = 35);
[tick] s = 31 ->
    1 : (s' = 36);
[tick] s = 29 ->
    0.185 : (s' = 37)
    + 0.63 : (s' = 38)
    + 0.185 : (s' = 39);
[tick] s = 39 ->
    1 : (s' = 40);
[tick] s = 38 ->
    1 : (s' = 41);
[tick] s = 37 ->
    1 : (s' = 42);
[tick] s = 28 ->
    0.185 : (s' = 43)
    + 0.63 : (s' = 44)
    + 0.185 : (s' = 45);
[tick] s = 45 ->
    1 : (s' = 46);
[tick] s = 44 ->
    1 : (s' = 47);
[tick] s = 43 ->
    1 : (s' = 48);
[tick] s = 4 ->
    0.185 : (s' = 49)
    + 0.63 : (s' = 50)
    + 0.185 : (s' = 51);
[tick] s = 51 ->
    0.185 : (s' = 52)
    + 0.63 : (s' = 53)
    + 0.185 : (s' = 54);
[tick] s = 54 ->
    1 : (s' = 55);
[tick] s = 53 ->
    1 : (s' = 56);
[tick] s = 52 ->
    1 : (s' = 57);
[tick] s = 50 ->
    0.185 : (s' = 58)
    + 0.63 : (s' = 59)
    + 0.185 : (s' = 60);
[tick] s = 60 ->
    1 : (s' = 61);
[tick] s = 59 ->
    1 : (s' = 62);
[tick] s = 58 ->
    1 : (s' = 63);
[tick] s = 49 ->
    0.185 : (s' = 64)
    + 0.63 : (s' = 65)
    + 0.185 : (s' = 66);
[tick] s = 66 ->
    1 : (s' = 67);
[tick] s = 65 ->
    1 : (s' = 68);
[tick] s = 64 ->
    1 : (s' = 69);
[tick] s = 2 ->
    0.185 : (s' = 70)
    + 0.63 : (s' = 71)
```

```
        + 0.185 : (s' = 72);
[tick] s = 72 ->
        0.185 : (s' = 73)
        + 0.63 : (s' = 74)
        + 0.185 : (s' = 75);
[tick] s = 75 ->
        0.185 : (s' = 76)
        + 0.63 : (s' = 77)
        + 0.185 : (s' = 78);
[tick] s = 78 ->
        1 : (s' = 79);
[tick] s = 77 ->
        1 : (s' = 80);
[tick] s = 76 ->
        1 : (s' = 81);
[tick] s = 74 ->
        0.185 : (s' = 82)
        + 0.63 : (s' = 83)
        + 0.185 : (s' = 84);
[tick] s = 84 ->
        1 : (s' = 85);
[tick] s = 83 ->
        1 : (s' = 86);
[tick] s = 82 ->
        1 : (s' = 87);
[tick] s = 73 ->
        0.185 : (s' = 88)
        + 0.63 : (s' = 89)
        + 0.185 : (s' = 90);
[tick] s = 90 ->
        1 : (s' = 91);
[tick] s = 89 ->
        1 : (s' = 92);
[tick] s = 88 ->
        1 : (s' = 93);
[tick] s = 71 ->
        0.185 : (s' = 94)
        + 0.63 : (s' = 95)
        + 0.185 : (s' = 96);
[tick] s = 96 ->
        0.185 : (s' = 97)
        + 0.63 : (s' = 98)
        + 0.185 : (s' = 99);
[tick] s = 99 ->
        1 : (s' = 100);
[tick] s = 98 ->
        1 : (s' = 101);
[tick] s = 97 ->
        1 : (s' = 102);
[tick] s = 95 ->
        0.185 : (s' = 103)
        + 0.63 : (s' = 104)
        + 0.185 : (s' = 105);
[tick] s = 105 ->
        1 : (s' = 106);
[tick] s = 104 ->
        1 : (s' = 107);
[tick] s = 103 ->
        1 : (s' = 108);
[tick] s = 94 ->
        0.185 : (s' = 109)
        + 0.63 : (s' = 110)
        + 0.185 : (s' = 111);
[tick] s = 111 ->
        1 : (s' = 112);
[tick] s = 110 ->
        1 : (s' = 113);
[tick] s = 109 ->
        1 : (s' = 114);
[tick] s = 70 ->
        0.185 : (s' = 115)
        + 0.63 : (s' = 116)
        + 0.185 : (s' = 117);
[tick] s = 117 ->
        0.185 : (s' = 118)
        + 0.63 : (s' = 119)
        + 0.185 : (s' = 120);
[tick] s = 120 ->
        1 : (s' = 121);
[tick] s = 119 ->
        1 : (s' = 122);
[tick] s = 118 ->
        1 : (s' = 123);
[tick] s = 116 ->
        0.185 : (s' = 124)
        + 0.63 : (s' = 125)
        + 0.185 : (s' = 126);
[tick] s = 126 ->
        1 : (s' = 127);
[tick] s = 125 ->
        1 : (s' = 128);
[tick] s = 124 ->
        1 : (s' = 129);
[tick] s = 115 ->
        0.185 : (s' = 130)
        + 0.63 : (s' = 131)
        + 0.185 : (s' = 132);
[tick] s = 132 ->
        1 : (s' = 133);
[tick] s = 131 ->
        1 : (s' = 134);
[tick] s = 130 ->
        1 : (s' = 135);
[tick] s = 1 ->
        0.185 : (s' = 136)
        + 0.63 : (s' = 137)
        + 0.185 : (s' = 138);
[tick] s = 138 ->
        0.185 : (s' = 139)
        + 0.63 : (s' = 140)
        + 0.185 : (s' = 141);
[tick] s = 141 ->
        0.185 : (s' = 142)
        + 0.63 : (s' = 143)
        + 0.185 : (s' = 144);
[tick] s = 144 ->
        1 : (s' = 145);
[tick] s = 143 ->
        1 : (s' = 146);
[tick] s = 142 ->
        1 : (s' = 147);
[tick] s = 140 ->
        0.185 : (s' = 148)
        + 0.63 : (s' = 149)
        + 0.185 : (s' = 150);
[tick] s = 150 ->
        1 : (s' = 151);
[tick] s = 149 ->
        1 : (s' = 152);
[tick] s = 148 ->
        1 : (s' = 153);
[tick] s = 139 ->
        0.185 : (s' = 154)
        + 0.63 : (s' = 155)
        + 0.185 : (s' = 156);
[tick] s = 156 ->
        1 : (s' = 157);
[tick] s = 155 ->
        1 : (s' = 158);
[tick] s = 154 ->
        1 : (s' = 159);
[tick] s = 137 ->
        0.185 : (s' = 160)
        + 0.63 : (s' = 161)
        + 0.185 : (s' = 162);
[tick] s = 162 ->
        0.185 : (s' = 163)
        + 0.63 : (s' = 164)
        + 0.185 : (s' = 165);
[tick] s = 165 ->
        1 : (s' = 166);
[tick] s = 164 ->
        1 : (s' = 167);
[tick] s = 163 ->
        1 : (s' = 168);
[tick] s = 161 ->
        0.185 : (s' = 169)
        + 0.63 : (s' = 170)
        + 0.185 : (s' = 171);
[tick] s = 171 ->
        1 : (s' = 172);
[tick] s = 170 ->
        1 : (s' = 173);
[tick] s = 169 ->
        1 : (s' = 174);
[tick] s = 160 ->
```

```
    0.185 : (s' = 175)
    + 0.63 : (s' = 176)
    + 0.185 : (s' = 177);
[tick] s = 177 −>
    1 : (s' = 178);
[tick] s = 176 −>
    1 : (s' = 179);
[tick] s = 175 −>
    1 : (s' = 180);
[tick] s = 136 −>
    0.185 : (s' = 181)
    + 0.63 : (s' = 182)
    + 0.185 : (s' = 183);
[tick] s = 183 −>
    0.185 : (s' = 184)
    + 0.63 : (s' = 185)
    + 0.185 : (s' = 186);
[tick] s = 186 −>
    1 : (s' = 187);
[tick] s = 185 −>
    1 : (s' = 188);
[tick] s = 184 −>
    1 : (s' = 189);
[tick] s = 182 −>
    0.185 : (s' = 190)
    + 0.63 : (s' = 191)
    + 0.185 : (s' = 192);
[tick] s = 192 −>
    1 : (s' = 193);
[tick] s = 191 −>
    1 : (s' = 194);
[tick] s = 190 −>
    1 : (s' = 195);
[tick] s = 181 −>
    0.185 : (s' = 196)
    + 0.63 : (s' = 197)
    + 0.185 : (s' = 198);
[tick] s = 198 −>
    1 : (s' = 199);
[tick] s = 197 −>
    1 : (s' = 200);
[tick] s = 196 −>
    1 : (s' = 201);
[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;
endmodule
formula stateValue = (s = 0 ? 0.0176932 : 0) +
                     (s = 3 ? 0.0214149 : 0) +
                     (s = 6 ? 0.0270787 : 0) +
                     (s = 9 ? 0.0333291 : 0) +
                     (s = 12 ? 0.0407615 : 0) +
                     (s = 13 ? 0.0387364 : 0) +
                     (s = 11 ? 0.0321959 : 0) +
                     (s = 14 ? 0.0311986 : 0) +
                     (s = 10 ? 0.0236303 : 0) +
                     (s = 15 ? 0.0236609 : 0) +
                     (s = 8 ? 0.0266955 : 0) +
                     (s = 18 ? 0.0328878 : 0) +
                     (s = 19 ? 0.0318075 : 0) +
                     (s = 17 ? 0.0263583 : 0) +
                     (s = 20 ? 0.0260615 : 0) +
                     (s = 16 ? 0.0198287 : 0) +
                     (s = 21 ? 0.0203155 : 0) +
                     (s = 7 ? 0.0200618 : 0) +
                     (s = 24 ? 0.0256958 : 0) +
                     (s = 25 ? 0.0254786 : 0) +
                     (s = 23 ? 0.0205206 : 0) +
                     (s = 26 ? 0.0209244 : 0) +
                     (s = 22 ? 0.0153454 : 0) +
                     (s = 27 ? 0.0163702 : 0) +
                     (s = 5 ? 0.0217113 : 0) +
                     (s = 30 ? 0.0273884 : 0) +
                     (s = 33 ? 0.0336873 : 0) +
                     (s = 34 ? 0.0325111 : 0) +
                     (s = 32 ? 0.026968 : 0) +
                     (s = 35 ? 0.0265981 : 0) +
                     (s = 31 ? 0.0202487 : 0) +
                     (s = 36 ? 0.0206851 : 0) +
                     (s = 29 ? 0.0219722 : 0) +
                     (s = 39 ? 0.0276628 : 0) +
                     (s = 40 ? 0.0272095 : 0) +
                     (s = 38 ? 0.0222018 : 0) +
                     (s = 41 ? 0.0224038 : 0) +
                     (s = 37 ? 0.0167408 : 0) +
                     (s = 42 ? 0.0175982 : 0) +
                     (s = 28 ? 0.0165561 : 0) +
                     (s = 45 ? 0.0223769 : 0) +
                     (s = 46 ? 0.0225579 : 0) +
                     (s = 44 ? 0.0174356 : 0) +
                     (s = 47 ? 0.0182096 : 0) +
                     (s = 43 ? 0.0124943 : 0) +
                     (s = 48 ? 0.0138613 : 0) +
                     (s = 4 ? 0.016344 : 0) +
                     (s = 51 ? 0.0221892 : 0) +
                     (s = 54 ? 0.0278924 : 0) +
                     (s = 55 ? 0.0274116 : 0) +
                     (s = 53 ? 0.0223928 : 0) +
                     (s = 56 ? 0.0225719 : 0) +
                     (s = 52 ? 0.0168932 : 0) +
                     (s = 57 ? 0.0177322 : 0) +
                     (s = 50 ? 0.017249 : 0) +
                     (s = 60 ? 0.0230006 : 0) +
                     (s = 61 ? 0.0231068 : 0) +
                     (s = 59 ? 0.0180454 : 0) +
                     (s = 62 ? 0.0187462 : 0) +
                     (s = 58 ? 0.0130901 : 0) +
                     (s = 63 ? 0.0143856 : 0) +
                     (s = 49 ? 0.0123087 : 0) +
                     (s = 66 ? 0.0189053 : 0) +
                     (s = 67 ? 0.0195029 : 0) +
                     (s = 65 ? 0.0136979 : 0) +
                     (s = 68 ? 0.0149204 : 0) +
                     (s = 64 ? 0.00849053 : 0) +
                     (s = 69 ? 0.0103379 : 0) +
                     (s = 2 ? 0.0163126 : 0) +
                     (s = 72 ? 0.0221616 : 0) +
                     (s = 75 ? 0.0278631 : 0) +
                     (s = 78 ? 0.0342389 : 0) +
                     (s = 79 ? 0.0329965 : 0) +
                     (s = 77 ? 0.0273858 : 0) +
                     (s = 80 ? 0.0269657 : 0) +
                     (s = 76 ? 0.0205326 : 0) +
                     (s = 81 ? 0.0209349 : 0) +
                     (s = 74 ? 0.0223684 : 0) +
                     (s = 84 ? 0.0280829 : 0) +
                     (s = 85 ? 0.0275792 : 0) +
                     (s = 83 ? 0.0225505 : 0) +
                     (s = 86 ? 0.0227107 : 0) +
                     (s = 82 ? 0.0170181 : 0) +
                     (s = 87 ? 0.0178422 : 0) +
                     (s = 73 ? 0.0168738 : 0) +
                     (s = 90 ? 0.0226608 : 0) +
                     (s = 91 ? 0.0228078 : 0) +
                     (s = 89 ? 0.0177152 : 0) +
                     (s = 92 ? 0.0184556 : 0) +
                     (s = 88 ? 0.0127696 : 0) +
                     (s = 93 ? 0.0141035 : 0) +
                     (s = 71 ? 0.0172213 : 0) +
                     (s = 96 ? 0.0229754 : 0) +
                     (s = 99 ? 0.0287337 : 0) +
                     (s = 100 ? 0.0281519 : 0) +
                     (s = 98 ? 0.0230846 : 0) +
                     (s = 101 ? 0.0231807 : 0) +
                     (s = 97 ? 0.0174355 : 0) +
                     (s = 102 ? 0.0182095 : 0) +
                     (s = 95 ? 0.018021 : 0) +
                     (s = 105 ? 0.0237147 : 0) +
                     (s = 106 ? 0.0237352 : 0) +
                     (s = 104 ? 0.0187248 : 0) +
                     (s = 107 ? 0.019344 : 0) +
                     (s = 103 ? 0.0137348 : 0) +
                     (s = 108 ? 0.0149529 : 0) +
                     (s = 94 ? 0.0130667 : 0) +
                     (s = 111 ? 0.0194823 : 0) +
                     (s = 112 ? 0.0200107 : 0) +
                     (s = 110 ? 0.0143649 : 0) +
                     (s = 113 ? 0.0155074 : 0) +
```

```
                    (s = 109 ? 0.00924753 : 0) +
                    (s = 114 ? 0.0110041 : 0) +
                    (s = 70 ? 0.0122811 : 0) +
                    (s = 117 ? 0.0188846 : 0) +
                    (s = 120 ? 0.0245369 : 0) +
                    (s = 121 ? 0.0244587 : 0) +
                    (s = 119 ? 0.0194847 : 0) +
                    (s = 122 ? 0.0200128 : 0) +
                    (s = 118 ? 0.0144325 : 0) +
                    (s = 123 ? 0.0155668 : 0) +
                    (s = 116 ? 0.0136736 : 0) +
                    (s = 126 ? 0.0199571 : 0) +
                    (s = 127 ? 0.0204285 : 0) +
                    (s = 125 ? 0.014899 : 0) +
                    (s = 128 ? 0.0159774 : 0) +
                    (s = 124 ? 0.009841 : 0) +
                    (s = 129 ? 0.0115263 : 0) +
                    (s = 115 ? 0.00846263 : 0) +
                    (s = 132 ? 0.0162147 : 0) +
                    (s = 133 ? 0.0171352 : 0) +
                    (s = 131 ? 0.0103134 : 0) +
                    (s = 134 ? 0.011942 : 0) +
                    (s = 130 ? 0.00441199 : 0) +
                    (s = 135 ? 0.0067488 : 0) +
                    (s = 1 ? 0.0112104 : 0) +
                    (s = 138 ? 0.0180987 : 0) +
                    (s = 141 ? 0.0237876 : 0) +
                    (s = 144 ? 0.0296178 : 0) +
                    (s = 145 ? 0.0289299 : 0) +
                    (s = 143 ? 0.0237994 : 0) +
                    (s = 146 ? 0.0238097 : 0) +
                    (s = 142 ? 0.0179809 : 0) +
                    (s = 147 ? 0.0186894 : 0) +
                    (s = 140 ? 0.0187931 : 0) +
                    (s = 150 ? 0.0244486 : 0) +
                    (s = 151 ? 0.024381 : 0) +
                    (s = 149 ? 0.0194042 : 0) +
                    (s = 152 ? 0.0199419 : 0) +
                    (s = 148 ? 0.0143597 : 0) +
                    (s = 153 ? 0.0155028 : 0) +
                    (s = 139 ? 0.0137985 : 0) +
                    (s = 156 ? 0.0200562 : 0) +
                    (s = 157 ? 0.0205157 : 0) +
                    (s = 155 ? 0.015009 : 0) +
                    (s = 158 ? 0.0160741 : 0) +
                    (s = 154 ? 0.0099617 : 0) +
                    (s = 159 ? 0.0116325 : 0) +
                    (s = 137 ? 0.0127314 : 0) +
                    (s = 162 ? 0.0192249 : 0) +
                    (s = 165 ? 0.0248675 : 0) +
                    (s = 166 ? 0.0247496 : 0) +
                    (s = 164 ? 0.0197842 : 0) +
                    (s = 167 ? 0.0202763 : 0) +
                    (s = 163 ? 0.0147008 : 0) +
                    (s = 168 ? 0.015803 : 0) +
                    (s = 161 ? 0.0140698 : 0) +
                    (s = 171 ? 0.0202733 : 0) +
                    (s = 172 ? 0.0207067 : 0) +
                    (s = 170 ? 0.0152477 : 0) +
                    (s = 173 ? 0.0162842 : 0) +
                    (s = 169 ? 0.0102221 : 0) +
                    (s = 174 ? 0.0118617 : 0) +
                    (s = 160 ? 0.00891477 : 0) +
                    (s = 177 ? 0.016513 : 0) +
                    (s = 178 ? 0.0173977 : 0) +
                    (s = 176 ? 0.0107112 : 0) +
                    (s = 179 ? 0.0122921 : 0) +
                    (s = 175 ? 0.00490948 : 0) +
                    (s = 180 ? 0.00718659 : 0) +
                    (s = 136 ? 0.00736404 : 0) +
                    (s = 183 ? 0.0155067 : 0) +
                    (s = 186 ? 0.0214635 : 0) +
                    (s = 187 ? 0.0217541 : 0) +
                    (s = 185 ? 0.0165121 : 0) +
                    (s = 188 ? 0.0173969 : 0) +
                    (s = 184 ? 0.0115608 : 0) +
                    (s = 189 ? 0.0130397 : 0) +
                    (s = 182 ? 0.0093466 : 0) +
                    (s = 192 ? 0.0168019 : 0) +
                    (s = 193 ? 0.0176519 : 0) +
                    (s = 191 ? 0.0110913 : 0) +
                    (s = 194 ? 0.0126265 : 0) +
                    (s = 190 ? 0.0053806 : 0) +
                    (s = 195 ? 0.00760117 : 0) +
                    (s = 181 ? 0.0031865 : 0) +
                    (s = 198 ? 0.0129875 : 0) +
                    (s = 199 ? 0.0142952 : 0) +
                    (s = 197 ? 0.00567036 : 0) +
                    (s = 200 ? 0.00785617 : 0) +
                    (s = 196 ? 0 : 0) +
                    (s = 201 ? 0.00286625 : 0);


module clk
    time : [0..HORIZON + 1] init 0;
    readyToTick : bool init true;
    [tick] readyToTick & time < HORIZON + 1 -> 1 : (time' = time + 1) &
            (readyToTick'=false);
    [tack] !readyToTick -> 1 : (readyToTick'=true);
endmodule

label "final" = time = HORIZON + 1;
formula sys_go = readyToTick;

module controller
    active_servers_A : [0..MAX_SERVERS_A] init ini_servers_A;
    active_servers_B : [0..MAX_SERVERS_B] init ini_servers_B;
    active_servers_C : [0..MAX_SERVERS_C] init ini_servers_C;

    dimmer : [1..DIMMER_LEVELS] init ini_dimmer;

    traffic_A : [0..4] init ini_traffic_A;
    traffic_B : [0..4] init ini_traffic_B;
    traffic_C : [0..4] init ini_traffic_C;

    [addServerA_complete] active_servers_A < MAX_SERVERS_A -> 1 :
            (active_servers_A' = active_servers_A + 1);
    [addServerB_complete] active_servers_B < MAX_SERVERS_B -> 1 :
            (active_servers_B' = active_servers_B + 1);
    [addServerC_complete] active_servers_C < MAX_SERVERS_C -> 1 :
            (active_servers_C' = active_servers_C + 1);

    [removeServerA_start] active_servers_A > 0 -> 1 : (active_servers_A' =
            active_servers_A - 1);
    [removeServerB_start] active_servers_B > 0 -> 1 : (active_servers_B' =
            active_servers_B - 1);
    [removeServerC_start] active_servers_C > 0 -> 1 : (active_servers_C'
            = active_servers_C - 1);

    [increaseDimmer_start] dimmer < DIMMER_LEVELS -> 1 : (dimmer' =
            dimmer + 1);
    [decreaseDimmer_start] dimmer > 1 -> 1 : (dimmer' = dimmer - 1);

    //A-B-C
    //Possible values 0-25-50-75-100

    // 100-0-0
    [divert_100_0_0] active_servers_A > 0
         -> 1 : (traffic_A' = 4) & (traffic_B' = 0) & (traffic_C' = 0);
    // 75-25-0
    [divert_75_25_0] active_servers_A > 0 & active_servers_B > 0
         -> 1 : (traffic_A' = 3) & (traffic_B' = 1) & (traffic_C' = 0);
    // 75-0-25
    [divert_75_0_25] active_servers_A > 0 & active_servers_C > 0
         -> 1 : (traffic_A' = 3) & (traffic_B' = 0) & (traffic_C' = 1);
    // 50-50-0
    [divert_50_50_0] active_servers_A > 0 & active_servers_B > 0
         -> 1 : (traffic_A' = 2) & (traffic_B' = 2) & (traffic_C' = 0);
    // 50-0-50
    [divert_50_0_50] active_servers_A > 0 & active_servers_C > 0
         -> 1 : (traffic_A' = 2) & (traffic_B' = 0) & (traffic_C' = 2);
    // 50-25-25
    [divert_50_25_25] active_servers_A > 0 & active_servers_B > 0 &
            active_servers_C > 0
         -> 1 : (traffic_A' = 2) & (traffic_B' = 1) & (traffic_C' = 1);
    // 25-75-0
    [divert_25_75_0] active_servers_A > 0 & active_servers_B > 0
         -> 1 : (traffic_A' = 1) & (traffic_B' = 3) & (traffic_C' = 0);
    // 25-0-75
    [divert_25_0_75] active_servers_A > 0 & active_servers_C > 0
         -> 1 : (traffic_A' = 1) & (traffic_B' = 0) & (traffic_C' = 3);
    // 25-50-25
    [divert_25_50_25] active_servers_A > 0 & active_servers_B > 0 &
            active_servers_C > 0
         -> 1 : (traffic_A' = 1) & (traffic_B' = 2) & (traffic_C' = 1);
    // 25-25-50
```

```
    [divert_25_25_50] active_servers_A > 0 & active_servers_B > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 1) & (traffic_B' = 1) & (traffic_C' = 2);
    // 0-100-0
    [divert_0_100_0] active_servers_B > 0
        -> 1 : (traffic_A' = 0) & (traffic_B' = 4) & (traffic_C' = 0);
    // 0-0-100
    [divert_0_0_100] active_servers_C > 0
        -> 1 : (traffic_A' = 0) & (traffic_B' = 0) & (traffic_C' = 4);
    // 0-75-25
    [divert_0_75_25] active_servers_B > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 0) & (traffic_B' = 3) & (traffic_C' = 1);
    // 0-25-75
    [divert_0_25_75] active_servers_B > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 0) & (traffic_B' = 1) & (traffic_C' = 3);
    // 0-50-50
    [divert_0_50_50] active_servers_B > 0 & active_servers_C > 0
        -> 1 : (traffic_A' = 0) & (traffic_B' = 2) & (traffic_C' = 2);
endmodule


formula addServerA_applicable = active_servers_A < MAX_SERVERS_A & !removeServer_used
                & addServerB_state = 0 & addServerC_state = 0;
formula addServerB_applicable = active_servers_B < MAX_SERVERS_B & !removeServer_used
                & addServerA_state = 0 & addServerC_state = 0;
formula addServerC_applicable = active_servers_C < MAX_SERVERS_C & !removeServer_used
                & addServerA_state = 0 & addServerB_state = 0;

formula removeServerA_applicable = active_servers_A > 0 & addServerA_state = 0 & active_servers > 1
                & addServerB_state = 0 & addServerC_state = 0;
formula removeServerB_applicable = active_servers_B > 0 & addServerB_state = 0 & active_servers > 1
                & addServerA_state = 0 & addServerC_state = 0;
formula removeServerC_applicable = active_servers_C > 0 & addServerC_state = 0 & active_servers > 1
                & addServerA_state = 0 & addServerB_state = 0;

formula increaseDimmer_compatible = !decreaseDimmer_used;
formula decreaseDimmer_compatible = !increaseDimmer_used;
formula increase_dimmer_applicable = dimmer < DIMMER_LEVELS & increaseDimmer_compatible;
formula decrease_dimmer_applicable = dimmer > 1 & decreaseDimmer_compatible;

const int addServer_LATENCY_PERIODS = ceil(addServer_LATENCY / PERIOD);

// This remove server constraints that only one server could be removed in one monitoring cycle.
module removeServer
    removeServer_go : bool init true;
    removeServer_used : bool init false;

    [removeServerA_start] sys_go & removeServer_go
        & removeServerA_applicable // applicability conditions
        -> (removeServer_go' = false) & (removeServer_used' = true);

    [removeServerB_start] sys_go & removeServer_go
        & removeServerB_applicable // applicability conditions
        -> (removeServer_go' = false) & (removeServer_used' = true);

    [removeServerC_start] sys_go & removeServer_go
        & removeServerC_applicable // applicability conditions
        -> (removeServer_go' = false) & (removeServer_used' = true);

    // Case when remove server tactic is applicable but not used
    [pass_remove_server] sys_go & removeServer_go // can go
        -> (removeServer_go' = false);

    [tick] !removeServer_go -> 1 : (removeServer_go' = true) & (removeServer_used' = false);
endmodule

module addServer
    addServerA_state : [0..addServer_LATENCY_PERIODS] init ini_addServerA_state;
    addServerB_state : [0..addServer_LATENCY_PERIODS] init ini_addServerB_state;
    addServerC_state : [0..addServer_LATENCY_PERIODS] init ini_addServerC_state;

    addServer_go : bool init true;

    // tactic applicable, start it
    [addServerA_start] sys_go & addServer_go // can go
        & addServerA_state = 0 // tactic has not been started
        & addServerA_applicable
        -> (addServerA_state' = 1) & (addServer_go' = false);

    // tactic applicable, start it
    [addServerB_start] sys_go & addServer_go // can go
        & addServerB_state = 0 // tactic has not been started
        & addServerB_applicable
        -> (addServerB_state' = 1) & (addServer_go' = false);

    // tactic applicable, start it
    [addServerC_start] sys_go & addServer_go // can go
        & addServerC_state = 0 // tactic has not been started
        & addServerC_applicable
        -> (addServerC_state' = 1) & (addServer_go' = false);

    // tactic applicable, but don't use it
    [pass_add] sys_go & addServer_go // can go
        & addServerA_state = 0 // tactic has not been started
        & addServerB_state = 0 & addServerC_state = 0
        //& addServerA_applicable
        -> (addServer_go' = false);

    // progress of the tactic
    [progressA] sys_go & addServer_go
        & addServerA_state > 0 & addServerA_state < addServer_LATENCY_PERIODS
        -> 1 : (addServerA_state' = addServerA_state + 1) & (addServer_go' = false);

    [progressB] sys_go & addServer_go
        & addServerB_state > 0 & addServerB_state < addServer_LATENCY_PERIODS
        -> 1 : (addServerB_state' = addServerB_state + 1) & (addServer_go' = false);

    [progressC] sys_go & addServer_go
        & addServerC_state > 0 & addServerC_state < addServer_LATENCY_PERIODS
        -> 1 : (addServerC_state' = addServerC_state + 1) & (addServer_go' = false);

    // completion of the tactic
    [addServerA_complete] sys_go & addServer_go
        & addServerA_state = addServer_LATENCY_PERIODS // completed
        -> 1 : (addServerA_state' = 0) & (addServer_go' = true); // so that it can start again at this time if needed

    [addServerB_complete] sys_go & addServer_go
        & addServerB_state = addServer_LATENCY_PERIODS // completed
        -> 1 : (addServerB_state' = 0) & (addServer_go' = true); // so that it can start again at this time if needed

    [addServerC_complete] sys_go & addServer_go
        & addServerC_state = addServer_LATENCY_PERIODS // completed
        -> 1 : (addServerC_state' = 0) & (addServer_go' = true); // so that it can start again at this time if needed

    [tick] !addServer_go -> 1 : (addServer_go' = true);
endmodule

// Make sure that divert traffic is executed at the end i.e.after adding or removing the servers.
formula divert_traffic_applicable = divert_go & !addServer_go & !removeServer_go & !increaseDimmer_go & !decreaseDimmer_go;

module divert_traffic
    divert_go : bool init true;

    //A-B-C
    //Possible values 0-25-50-75-100

    // 100-0-0
    [divert_100_0_0] sys_go & divert_traffic_applicable
```

```
        -> 1 : (divert_go'=false);
    // 75-25-0
    [divert_75_25_0] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 75-0-25
    [divert_75_0_25] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 50-50-0
    [divert_50_50_0] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 50-0-50
    [divert_50_0_50] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 50-25-25
    [divert_50_25_25] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 25-75-0
    [divert_25_75_0] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 25-0-75
    [divert_25_0_75] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 25-50-25
    [divert_25_50_25] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 25-25-50
    [divert_25_25_50] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 0-100-0
    [divert_0_100_0] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 0-0-100
    [divert_0_0_100] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 0-75-25
    [divert_0_75_25] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 0-25-75
    [divert_0_25_75] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);
    // 0-50-50
    [divert_0_50_50] sys_go & divert_traffic_applicable
        -> 1 : (divert_go'=false);

    [tick] !divert_go -> 1 : (divert_go' = true);
endmodule

module increaseDimmer
    increaseDimmer_go : bool init true;
    increaseDimmer_used : bool init false;

    [increaseDimmer_start] sys_go & increaseDimmer_go
        & increase_dimmer_applicable // applicability conditions
        -> (increaseDimmer_go' = false) & (increaseDimmer_used' = true);

    // tactic applicable but not used
    [pass_inc_dimmer] sys_go & increaseDimmer_go // can go
        -> (increaseDimmer_go' = false);

    [tick] !increaseDimmer_go -> 1 : (increaseDimmer_go' = true) &
            (increaseDimmer_used' = false);
endmodule

// tactic
module decreaseDimmer
    decreaseDimmer_go : bool init true;
    decreaseDimmer_used : bool init false;

    [decreaseDimmer_start] sys_go & decreaseDimmer_go
        & decrease_dimmer_applicable // applicability conditions
        -> (decreaseDimmer_go' = false) & (decreaseDimmer_used' =
            true);

    // tactic applicable but not used
    [pass_dec_dimmer] sys_go & decreaseDimmer_go // can go
        -> (decreaseDimmer_go' = false);

    [tick] !decreaseDimmer_go -> 1 : (decreaseDimmer_go' = true) &
            (decreaseDimmer_used' = false);
endmodule

//*****************************************************************************
// Queuing network with each server having queueing model of M/G/1/PS
//*****************************************************************************
formula dimmerFactor = (dimmer - 1) / (DIMMER_LEVELS - 1);
formula interarrivalMean = stateValue * interArrivalScaleFactorForDecision;

formula Pa = (traffic_A * 25)/100;
formula Pb = (traffic_B * 25)/100;
formula Pc = (traffic_C * 25)/100;

formula loaded_servers = (Pa != 0 ? 1 : 0) + (Pb != 0 ? 1 : 0) + (Pc != 0 ? 1 :
        0);

formula service_rate_A = dimmerFactor *
        (MAX_ARRIVALA_CAPACITY_LOW)
            + (1 - dimmerFactor) * (MAX_ARRIVALA_CAPACITY);
formula service_rate_B = dimmerFactor *
        (MAX_ARRIVALB_CAPACITY_LOW)
            + (1 - dimmerFactor) * (MAX_ARRIVALB_CAPACITY);
formula service_rate_C = dimmerFactor *
        (MAX_ARRIVALC_CAPACITY_LOW)
            + (1 - dimmerFactor) * (MAX_ARRIVALC_CAPACITY);

formula rhoA = Pa/(service_rate_A*interarrivalMean);
formula rhoB = Pb/(service_rate_B*interarrivalMean);
formula rhoC = Pc/(service_rate_C*interarrivalMean);

formula overloaded = (rhoA >= 1 | rhoB >= 1 | rhoC >= 1);

formula rt_A = 1/(service_rate_A - (throughput*Pa));
formula rt_B = 1/(service_rate_B - (throughput*Pb));
formula rt_C = 1/(service_rate_C - (throughput*Pc));

// Response time to clients utility function
const double RT_THRESHOLD = 1.0;

formula expected_wait_time = (Pa*rt_A + Pb*rt_B + Pc*rt_C);
formula rt = (interarrivalMean = 0 ? 0 : (overloaded ? RT_THRESHOLD + 2
        : expected_wait_time));

const double NORMAL_A_REVENUE = (SERVERA_COST /
        MAX_ARRIVALA_CAPACITY) * 10;
const double DIMMER_A_REVENUE = (SERVERA_COST /
        MAX_ARRIVALA_CAPACITY_LOW) * 3 / 2;
const double NORMAL_B_REVENUE = (SERVERB_COST /
        MAX_ARRIVALB_CAPACITY) * 10;
const double DIMMER_B_REVENUE = (SERVERB_COST /
        MAX_ARRIVALB_CAPACITY_LOW) * 3 / 2;
const double NORMAL_C_REVENUE = (SERVERC_COST /
        MAX_ARRIVALC_CAPACITY) * 10;
const double DIMMER_C_REVENUE = (SERVERC_COST /
        MAX_ARRIVALC_CAPACITY_LOW) * 3 / 2;

const double DIMMER_REVENUE = DIMMER_A_REVENUE +
        DIMMER_B_REVENUE + DIMMER_C_REVENUE;
const double NORMAL_REVENUE = NORMAL_A_REVENUE +
        NORMAL_B_REVENUE + NORMAL_C_REVENUE;

formula serverA_cost = ((addServerA_state > 0 ? 1 : 0) +
        active_servers_A) * SERVERA_COST;
formula serverB_cost = ((addServerB_state > 0 ? 1 : 0) +
        active_servers_B) * SERVERB_COST;
formula serverC_cost = ((addServerC_state > 0 ? 1 : 0) +
        active_servers_C) * SERVERC_COST;
formula cost = serverA_cost + serverB_cost + serverC_cost;

formula throughput = 1/interarrivalMean;

formula basicUtilityA = throughput * Pa * (dimmerFactor *
        DIMMER_A_REVENUE + (1 - dimmerFactor) *
        NORMAL_A_REVENUE);
formula basicUtilityB = throughput * Pb * (dimmerFactor *
        DIMMER_B_REVENUE + (1 - dimmerFactor) *
        NORMAL_B_REVENUE);
formula basicUtilityC = throughput * Pc * (dimmerFactor *
        DIMMER_C_REVENUE + (1 - dimmerFactor) *
        NORMAL_C_REVENUE);

formula basicUtility = basicUtilityA + basicUtilityB + basicUtilityC;
formula active_servers = active_servers_A + active_servers_B +
        active_servers_C;
formula poweredServers = (addServerA_state > 0 ? 1 : 0) +
        (addServerB_state > 0 ? 1 : 0) + (addServerC_state > 0 ? 1 : 0 )
        + active_servers;
formula MAX_SERVERS = MAX_SERVERS_A + MAX_SERVERS_B +
```

```
        MAX_SERVERS_C;

formula MAX_SERVER_COST = MAX_SERVERS_A * SERVERA_COST
        + MAX_SERVERS_B * SERVERB_COST
        + MAX_SERVERS_C * SERVERC_COST;

formula netPenalty = stateValue > 0 ? penalty / stateValue : 0;

formula uTotal = (overloaded & (poweredServers < MAX_SERVERS |
        dimmer < DIMMER_LEVELS | active_servers != loaded_servers))
                ? −(1000) // avoid unstable solutions
                : (((rt > RT_THRESHOLD | rt <= 0) ? netPenalty :
                        basicUtility) − cost);


rewards "util"
    // 100000000.0 is added to avoid a negative value during calculation;
            negative utility is not supported by PRISM.
    [tack] true : 100000000.0 + (PERIOD)*(uTotal);
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 ρ_{mdps} plans with a shorter horizon compared to deliberative planning ρ_{mdpl}. Moreover, while planning, ρ_{mdps} do not consider adaptation actions `IncAlt`, `DecAlt`, and `EcmOn`, and `EcmOff`.

```
mdp
const double PERIOD = 60;
const int HORIZON = 2; // Planning horizon for reactive planning
const double IncAlt_LATENCY = 60;
const double DecAlt_LATENCY = 60;
const int MAX_ALT_LEVEL = 3;
const double destructionFormationFactor = 1.5;
const double threatRange = 3;
const double detectionFormationFactor = 1.2;
const double sensorRange = 4;
const init_a = 0;
const init_c = 0;
const init_f = 0;
const bool ECM_ENABLED = false; // ECM is not enabled for reactive
        planning
const bool ONE_LEVEL_ENABLED = false; // This is not enabled for
        reactive planning
const bool TWO_LEVEL_ENABLED = true; // Two level increase/decrease
        altitude enabled
const int ini_IncAlt_state = 0;
const int ini_DecAlt_state = 0;
const int ini_IncAlt2_state = 0;
const int ini_DecAlt2_state = 0;
const double ecm_threat_prob = 0.15;
const double ecm_target_prob = 0.3;
const double survival_reward = 1;


//**********************************************************************
// CLOCK
//**********************************************************************
const int TO_TICK = 0;
const int TO_TICK2 = 1; // intermediate tick for constraint satisf. update
const int TO_TACK = 2;

label "final" = time = HORIZON & clockstep=TO_TICK;
formula sys_go = clockstep=TO_TICK;

module clk
    time : [0..HORIZON] init 0;
    clockstep : [0..2] init TO_TICK;

    [tick] clockstep=TO_TICK & time < HORIZON −> 1: (time'=time+1) &
            (clockstep'=TO_TICK2);
    [tick2] clockstep=TO_TICK2 −> 1 : (clockstep'=TO_TACK);
    [tack] clockstep=TO_TACK −> 1: (clockstep'=TO_TICK);
endmodule

module env
s : [0..45] init 0;
[tick] s = 0 −>
    0.034225 : (s' = 1)
    + 0.11655 : (s' = 2)
    + 0.034225 : (s' = 3)
    + 0.11655 : (s' = 4)
    + 0.3969 : (s' = 5)
    + 0.11655 : (s' = 6)
```

```
      + 0.034225 : (s’ = 7)
      + 0.11655 : (s’ = 8)
      + 0.034225 : (s’ = 9);
[tick] s = 1 ->
      0.034225 : (s’ = 10)
      + 0.11655 : (s’ = 11)
      + 0.034225 : (s’ = 12)
      + 0.11655 : (s’ = 13)
      + 0.3969 : (s’ = 14)
      + 0.11655 : (s’ = 15)
      + 0.034225 : (s’ = 16)
      + 0.11655 : (s’ = 17)
      + 0.034225 : (s’ = 18);
[tick] s = 2 ->
      0.034225 : (s’ = 10)
      + 0.11655 : (s’ = 11)
      + 0.034225 : (s’ = 12)
      + 0.11655 : (s’ = 13)
      + 0.3969 : (s’ = 14)
      + 0.11655 : (s’ = 15)
      + 0.034225 : (s’ = 16)
      + 0.11655 : (s’ = 17)
      + 0.034225 : (s’ = 18);
[tick] s = 3 ->
      0.034225 : (s’ = 10)
      + 0.11655 : (s’ = 11)
      + 0.034225 : (s’ = 12)
      + 0.11655 : (s’ = 13)
      + 0.3969 : (s’ = 14)
      + 0.11655 : (s’ = 15)
      + 0.034225 : (s’ = 16)
      + 0.11655 : (s’ = 17)
      + 0.034225 : (s’ = 18);
[tick] s = 4 ->
      0.034225 : (s’ = 10)
      + 0.11655 : (s’ = 11)
      + 0.034225 : (s’ = 12)
      + 0.11655 : (s’ = 13)
      + 0.3969 : (s’ = 14)
      + 0.11655 : (s’ = 15)
      + 0.034225 : (s’ = 16)
      + 0.11655 : (s’ = 17)
      + 0.034225 : (s’ = 18);
[tick] s = 5 ->
      0.034225 : (s’ = 10)
      + 0.11655 : (s’ = 11)
      + 0.034225 : (s’ = 12)
      + 0.11655 : (s’ = 13)
      + 0.3969 : (s’ = 14)
      + 0.11655 : (s’ = 15)
      + 0.034225 : (s’ = 16)
      + 0.11655 : (s’ = 17)
      + 0.034225 : (s’ = 18);
[tick] s = 6 ->
      0.034225 : (s’ = 10)
      + 0.11655 : (s’ = 11)
      + 0.034225 : (s’ = 12)
      + 0.11655 : (s’ = 13)
      + 0.3969 : (s’ = 14)
      + 0.11655 : (s’ = 15)
      + 0.034225 : (s’ = 16)
      + 0.11655 : (s’ = 17)
      + 0.034225 : (s’ = 18);
[tick] s = 7 ->
      0.034225 : (s’ = 10)
      + 0.11655 : (s’ = 11)
      + 0.034225 : (s’ = 12)
      + 0.11655 : (s’ = 13)
      + 0.3969 : (s’ = 14)
      + 0.11655 : (s’ = 15)
      + 0.034225 : (s’ = 16)
      + 0.11655 : (s’ = 17)
      + 0.034225 : (s’ = 18);
[tick] s = 8 ->
      0.034225 : (s’ = 10)
      + 0.11655 : (s’ = 11)
      + 0.034225 : (s’ = 12)
      + 0.11655 : (s’ = 13)
      + 0.3969 : (s’ = 14)
      + 0.11655 : (s’ = 15)
      + 0.034225 : (s’ = 16)
      + 0.11655 : (s’ = 17)
      + 0.034225 : (s’ = 18);
[tick] s = 9 ->
      0.034225 : (s’ = 10)
      + 0.11655 : (s’ = 11)
      + 0.034225 : (s’ = 12)
      + 0.11655 : (s’ = 13)
      + 0.3969 : (s’ = 14)
      + 0.11655 : (s’ = 15)
      + 0.034225 : (s’ = 16)
      + 0.11655 : (s’ = 17)
      + 0.034225 : (s’ = 18);
[tick] s = 10 ->
      0.034225 : (s’ = 19)
      + 0.11655 : (s’ = 20)
      + 0.034225 : (s’ = 21)
      + 0.11655 : (s’ = 22)
      + 0.3969 : (s’ = 23)
      + 0.11655 : (s’ = 24)
      + 0.034225 : (s’ = 25)
      + 0.11655 : (s’ = 26)
      + 0.034225 : (s’ = 27);
[tick] s = 11 ->
      0.034225 : (s’ = 19)
      + 0.11655 : (s’ = 20)
      + 0.034225 : (s’ = 21)
      + 0.11655 : (s’ = 22)
      + 0.3969 : (s’ = 23)
      + 0.11655 : (s’ = 24)
      + 0.034225 : (s’ = 25)
      + 0.11655 : (s’ = 26)
      + 0.034225 : (s’ = 27);
[tick] s = 12 ->
      0.034225 : (s’ = 19)
      + 0.11655 : (s’ = 20)
      + 0.034225 : (s’ = 21)
      + 0.11655 : (s’ = 22)
      + 0.3969 : (s’ = 23)
      + 0.11655 : (s’ = 24)
      + 0.034225 : (s’ = 25)
      + 0.11655 : (s’ = 26)
      + 0.034225 : (s’ = 27);
[tick] s = 13 ->
      0.034225 : (s’ = 19)
      + 0.11655 : (s’ = 20)
      + 0.034225 : (s’ = 21)
      + 0.11655 : (s’ = 22)
      + 0.3969 : (s’ = 23)
      + 0.11655 : (s’ = 24)
      + 0.034225 : (s’ = 25)
      + 0.11655 : (s’ = 26)
      + 0.034225 : (s’ = 27);
[tick] s = 14 ->
      0.034225 : (s’ = 19)
      + 0.11655 : (s’ = 20)
      + 0.034225 : (s’ = 21)
      + 0.11655 : (s’ = 22)
      + 0.3969 : (s’ = 23)
      + 0.11655 : (s’ = 24)
      + 0.034225 : (s’ = 25)
      + 0.11655 : (s’ = 26)
      + 0.034225 : (s’ = 27);
[tick] s = 15 ->
      0.034225 : (s’ = 19)
      + 0.11655 : (s’ = 20)
      + 0.034225 : (s’ = 21)
      + 0.11655 : (s’ = 22)
      + 0.3969 : (s’ = 23)
      + 0.11655 : (s’ = 24)
      + 0.034225 : (s’ = 25)
      + 0.11655 : (s’ = 26)
      + 0.034225 : (s’ = 27);
[tick] s = 16 ->
      0.034225 : (s’ = 19)
      + 0.11655 : (s’ = 20)
      + 0.034225 : (s’ = 21)
      + 0.11655 : (s’ = 22)
      + 0.3969 : (s’ = 23)
      + 0.11655 : (s’ = 24)
      + 0.034225 : (s’ = 25)
      + 0.11655 : (s’ = 26)
      + 0.034225 : (s’ = 27);
[tick] s = 17 ->
      0.034225 : (s’ = 19)
      + 0.11655 : (s’ = 20)
```

```
        + 0.034225 : (s' = 21)
        + 0.11655 : (s' = 22)
        + 0.3969 : (s' = 23)
        + 0.11655 : (s' = 24)
        + 0.034225 : (s' = 25)
        + 0.11655 : (s' = 26)
        + 0.034225 : (s' = 27);
    [tick] s = 18 −>
        0.034225 : (s' = 19)
        + 0.11655 : (s' = 20)
        + 0.034225 : (s' = 21)
        + 0.11655 : (s' = 22)
        + 0.3969 : (s' = 23)
        + 0.11655 : (s' = 24)
        + 0.034225 : (s' = 25)
        + 0.11655 : (s' = 26)
        + 0.034225 : (s' = 27);
    [tick] s = 19 −>
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 20 −>
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 21 −>
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 22 −>
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 23 −>
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 24 −>
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 25 −>
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 26 −>
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 27 −>
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 28 −>
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 29 −>
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 30 −>
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 31 −>
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 32 −>
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 33 −>
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
```

```
      + 0.034225 : (s' = 45);
[tick] s = 34 −>
      0.034225 : (s' = 37)
      + 0.11655 : (s' = 38)
      + 0.034225 : (s' = 39)
      + 0.11655 : (s' = 40)
      + 0.3969 : (s' = 41)
      + 0.11655 : (s' = 42)
      + 0.034225 : (s' = 43)
      + 0.11655 : (s' = 44)
      + 0.034225 : (s' = 45);
[tick] s = 35 −>
      0.034225 : (s' = 37)
      + 0.11655 : (s' = 38)
      + 0.034225 : (s' = 39)
      + 0.11655 : (s' = 40)
      + 0.3969 : (s' = 41)
      + 0.11655 : (s' = 42)
      + 0.034225 : (s' = 43)
      + 0.11655 : (s' = 44)
      + 0.034225 : (s' = 45);
[tick] s = 36 −>
      0.034225 : (s' = 37)
      + 0.11655 : (s' = 38)
      + 0.034225 : (s' = 39)
      + 0.11655 : (s' = 40)
      + 0.3969 : (s' = 41)
      + 0.11655 : (s' = 42)
      + 0.034225 : (s' = 43)
      + 0.11655 : (s' = 44)
      + 0.034225 : (s' = 45);
endmodule

// environment has 2 components. statevalue for threats and statevalue1 is for targets
formula stateValue = (s = 0 ? 0 : 0) +
                     (s = 1 ? 0.00605639 : 0) +
                     (s = 2 ? 0.00605639 : 0) +
                     (s = 3 ? 0.00605639 : 0) +
                     (s = 4 ? 0.0282836 : 0) +
                     (s = 5 ? 0.0282836 : 0) +
                     (s = 6 ? 0.0282836 : 0) +
                     (s = 7 ? 0.0778979 : 0) +
                     (s = 8 ? 0.0778979 : 0) +
                     (s = 9 ? 0.0778979 : 0) +
                     (s = 10 ? 0 : 0) +
                     (s = 11 ? 0 : 0) +
                     (s = 12 ? 0 : 0) +
                     (s = 13 ? 0 : 0) +
                     (s = 14 ? 0 : 0) +
                     (s = 15 ? 0 : 0) +
                     (s = 16 ? 0 : 0) +
                     (s = 17 ? 0 : 0) +
                     (s = 18 ? 0 : 0) +
                     (s = 19 ? 0.750751 : 0) +
                     (s = 20 ? 0.750751 : 0) +
                     (s = 21 ? 0.750751 : 0) +
                     (s = 22 ? 0.885424 : 0) +
                     (s = 23 ? 0.885424 : 0) +
                     (s = 24 ? 0.885424 : 0) +
                     (s = 25 ? 0.963485 : 0) +
                     (s = 26 ? 0.963485 : 0) +
                     (s = 27 ? 0.963485 : 0) +
                     (s = 28 ? 0.435626 : 0) +
                     (s = 29 ? 0.435626 : 0) +
                     (s = 30 ? 0.435626 : 0) +
                     (s = 31 ? 0.676196 : 0) +
                     (s = 32 ? 0.676196 : 0) +
                     (s = 33 ? 0.676196 : 0) +
                     (s = 34 ? 0.864925 : 0) +
                     (s = 35 ? 0.864925 : 0) +
                     (s = 36 ? 0.864925 : 0) +
                     (s = 37 ? 0.368403 : 0) +
                     (s = 38 ? 0.368403 : 0) +
                     (s = 39 ? 0.368403 : 0) +
                     (s = 40 ? 0.793701 : 0) +
                     (s = 41 ? 0.793701 : 0) +
                     (s = 42 ? 0.793701 : 0) +
                     (s = 43 ? 0.983048 : 0) +
                     (s = 44 ? 0.983048 : 0) +
                     (s = 45 ? 0.983048 : 0);

formula stateValue1 = (s = 0 ? 0 : 0) +
                      (s = 1 ? 0.830037 : 0) +
                      (s = 2 ? 0.904439 : 0) +
                      (s = 3 ? 0.954777 : 0) +
                      (s = 4 ? 0.830037 : 0) +
                      (s = 5 ? 0.904439 : 0) +
                      (s = 6 ? 0.954777 : 0) +
                      (s = 7 ? 0.830037 : 0) +
                      (s = 8 ? 0.904439 : 0) +
                      (s = 9 ? 0.954777 : 0) +
                      (s = 10 ? 0 : 0) +
                      (s = 11 ? 0 : 0) +
                      (s = 12 ? 0 : 0) +
                      (s = 13 ? 0 : 0) +
                      (s = 14 ? 0 : 0) +
                      (s = 15 ? 0 : 0) +
                      (s = 16 ? 0 : 0) +
                      (s = 17 ? 0 : 0) +
                      (s = 18 ? 0 : 0) +
                      (s = 19 ? 0.0156741 : 0) +
                      (s = 20 ? 0.0719057 : 0) +
                      (s = 21 ? 0.190204 : 0) +
                      (s = 22 ? 0.0156741 : 0) +
                      (s = 23 ? 0.0719057 : 0) +
                      (s = 24 ? 0.190204 : 0) +
                      (s = 25 ? 0.0156741 : 0) +
                      (s = 26 ? 0.0719057 : 0) +
                      (s = 27 ? 0.190204 : 0) +
                      (s = 28 ? 0 : 0) +
                      (s = 29 ? 0 : 0) +
                      (s = 30 ? 0 : 0) +
                      (s = 31 ? 0 : 0) +
                      (s = 32 ? 0 : 0) +
                      (s = 33 ? 0 : 0) +
                      (s = 34 ? 0 : 0) +
                      (s = 35 ? 0 : 0) +
                      (s = 36 ? 0 : 0) +
                      (s = 37 ? 0 : 0) +
                      (s = 38 ? 0 : 0) +
                      (s = 39 ? 0 : 0) +
                      (s = 40 ? 0 : 0) +
                      (s = 41 ? 0 : 0) +
                      (s = 42 ? 0 : 0) +
                      (s = 43 ? 0 : 0) +
                      (s = 44 ? 0 : 0) +
                      (s = 45 ? 0 : 0);


//***********************************************************************
// SYSTEM
//***********************************************************************

// Variable range and initialization
const a_MIN=0; const a_MAX=MAX_ALT_LEVEL; const a_INIT=init_a;
const f_MIN=0; const f_MAX=1; const f_INIT=init_f;
const c_MIN=0; const c_MAX=1; const c_INIT=init_c;

module sys
   a : [a_MIN..a_MAX] init a_INIT;
   f : [f_MIN..f_MAX] init f_INIT;
   c : [c_MIN..c_MAX] init c_INIT;

   [EcmOn_start] c=0 & ECM_ENABLED −> 1: (c'=c_EcmOn_impact);
   [EcmOff_start] c=1 & ECM_ENABLED −> 1: (c'=c_EcmOff_impact);

   [GoTight_start] f=0 −> 1: (a'=a_GoTight_impact)
         & (f'=f_GoTight_impact);
   [GoLoose_start] f=1 −> 1: (a'=a_GoLoose_impact)
         & (f'=f_GoLoose_impact);

   [IncAlt_complete] a < MAX_ALT_LEVEL & ONE_LEVEL_ENABLED −>
         1: (a'=a_IncAlt_impact)
        & (f'=f_IncAlt_impact);
   [IncAlt2_complete] a < MAX_ALT_LEVEL−1 &
         TWO_LEVEL_ENABLED −> 1: (a'=a_IncAlt2_impact);

   [DecAlt_complete] a > 0 & ONE_LEVEL_ENABLED −> 1:
         (a'=a_DecAlt_impact)
        & (f'=f_DecAlt_impact);
   [DecAlt2_complete] a > 1 & TWO_LEVEL_ENABLED −> 1:
         (a'=a_DecAlt2_impact);
endmodule
```

```
formula c_EcmOn_impact = c + (1) >= c_MIN ? ( c+(1)<=c_MAX? c+(1) : c_MAX) : c_MIN;
formula c_EcmOff_impact = c + (-1) >= c_MIN ? ( c+(-1)<=c_MAX? c+(-1) : c_MAX) : c_MIN;
formula a_GoTight_impact = a + (0) >= a_MIN ? ( a+(0)<=a_MAX? a+(0) : a_MAX) : a_MIN;
formula f_GoTight_impact = f + (1) >= f_MIN ? ( f+(1)<=f_MAX? f+(1) : f_MAX) : f_MIN;
formula a_GoLoose_impact = a + (0) >= a_MIN ? ( a+(0)<=a_MAX? a+(0) : a_MAX) : a_MIN;
formula f_GoLoose_impact = f + (-1) >= f_MIN ? ( f+(-1)<=f_MAX? f+(-1) : f_MAX) : f_MIN;
formula a_IncAlt_impact = a + (1) >= a_MIN ? ( a+(1)<=a_MAX? a+(1) : a_MAX) : a_MIN;
formula f_IncAlt_impact = f + (0) >= f_MIN ? ( f+(0)<=f_MAX? f+(0) : f_MAX) : f_MIN;
formula a_DecAlt_impact = a + (-1) >= a_MIN ? ( a+(-1)<=a_MAX? a+(-1) : a_MAX) : a_MIN;
formula f_DecAlt_impact = f + (0) >= f_MIN ? ( f+(0)<=f_MAX? f+(0) : f_MAX) : f_MIN;
formula a_IncAlt2_impact = a + (2) >= a_MIN ? ( a+(2)<=a_MAX? a+(2) : a_MAX) : a_MIN;
formula a_DecAlt2_impact = a + (-2) >= a_MIN ? ( a+(-2)<=a_MAX? a+(-2) : a_MAX) : a_MIN;

// tactic concurrency rules
formula IncAlt_used = IncAlt_state != 0;
formula DecAlt_used = DecAlt_state != 0;
formula IncAlt2_used = IncAlt2_state != 0;
formula DecAlt2_used = DecAlt2_state != 0;

formula EcmOn_compatible = !EcmOn_used;
formula EcmOff_compatible = !EcmOff_used;
formula GoTight_compatible = !GoLoose_used;
formula GoLoose_compatible = !GoTight_used;
formula IncAlt_compatible = (!DecAlt_used) & (!IncAlt2_used) & (!DecAlt2_used);
formula DecAlt_compatible = (!IncAlt_used) & (!IncAlt2_used) & (!DecAlt2_used);
formula IncAlt2_compatible = (!DecAlt_used) & (!IncAlt_used) & (!DecAlt2_used);
formula DecAlt2_compatible = (!DecAlt_used) & (!IncAlt_used) & (!IncAlt2_used);


//*********************************************************************
// TACTIC: EcmOn
//*********************************************************************

// Applicability conditions
formula EcmOn_applicable = EcmOn_compatible & c=0;

module EcmOn
    EcmOn_used : bool init false;
    EcmOn_go : bool init true;

    // Tactic applicable, start it
    [EcmOn_start] sys_go & EcmOn_go & EcmOn_applicable & ECM_ENABLED -> (EcmOn_used'=true) & (EcmOn_go'=false);

    // Tactic applicable, but do not start it
    [EcmOn_pass] sys_go & EcmOn_go & EcmOn_applicable -> (EcmOn_go'=false);

    // Pass if the tactic is not applicable
    [EcmOn_invalid] sys_go & EcmOn_go & !EcmOn_applicable -> 1 : (EcmOn_go'=false);

    [tick] !EcmOn_go -> 1: (EcmOn_go'=true) & (EcmOn_used'=false);
endmodule


//*********************************************************************
// TACTIC: EcmOff
//*********************************************************************

// Applicability conditions
formula EcmOff_applicable = EcmOff_compatible & c=1;

module EcmOff
    EcmOff_used : bool init false;
    EcmOff_go : bool init true;

    // Tactic applicable, start it
    [EcmOff_start] sys_go & EcmOff_go & EcmOff_applicable & ECM_ENABLED -> (EcmOff_used'=true) & (EcmOff_go'=false);

    // Tactic applicable, but do not start it
    [EcmOff_pass] sys_go & EcmOff_go & EcmOff_applicable -> (EcmOff_go'=false);

    // Pass if the tactic is not applicable
    [EcmOff_invalid] sys_go & EcmOff_go & !EcmOff_applicable -> 1 : (EcmOff_go'=false);

    [tick] !EcmOff_go -> 1: (EcmOff_go'=true) & (EcmOff_used'=false);
endmodule


//*********************************************************************
// TACTIC: GoTight
//*********************************************************************

// Applicability conditions
formula GoTight_applicable = GoTight_compatible & f=0;

module GoTight
    GoTight_used : bool init false;
    GoTight_go : bool init true;

    // Tactic applicable, start it
    [GoTight_start] sys_go & GoTight_go & GoTight_applicable -> (GoTight_used'=true) & (GoTight_go'=false);

    // Tactic applicable, but do not start it
    [GoTight_pass] sys_go & GoTight_go & GoTight_applicable -> (GoTight_go'=false);

    // Pass if the tactic is not applicable
    [GoTight_invalid] sys_go & GoTight_go & !GoTight_applicable -> 1 : (GoTight_go'=false);

    [tick] !GoTight_go -> 1: (GoTight_go'=true) & (GoTight_used'=false);
endmodule


//*********************************************************************
// TACTIC: GoLoose
//*********************************************************************

// Applicability conditions
formula GoLoose_applicable = GoLoose_compatible & f=1;

module GoLoose
    GoLoose_used : bool init false;
    GoLoose_go : bool init true;

    // Tactic applicable, start it
    [GoLoose_start] sys_go & GoLoose_go & GoLoose_applicable -> (GoLoose_used'=true) & (GoLoose_go'=false);

    // Tactic applicable, but do not start it
    [GoLoose_pass] sys_go & GoLoose_go & GoLoose_applicable -> (GoLoose_go'=false);

    // Pass if the tactic is not applicable
    [GoLoose_invalid] sys_go & GoLoose_go & !GoLoose_applicable -> 1 : (GoLoose_go'=false);

    [tick] !GoLoose_go -> 1: (GoLoose_go'=true) & (GoLoose_used'=false);
endmodule


//*********************************************************************
// TACTIC: IncAlt
//*********************************************************************

const int IncAlt_LATENCY_PERIODS = ceil(IncAlt_LATENCY/PERIOD);

// Applicability conditions
formula IncAlt_applicable = IncAlt_compatible & a < MAX_ALT_LEVEL;

module IncAlt
    IncAlt_state : [0..IncAlt_LATENCY_PERIODS] init ini_IncAlt_state;
    IncAlt_go : bool init true;
```

```
    // Tactic applicable, start it
    [IncAlt_start] sys_go & IncAlt_go & IncAlt_state=0 & IncAlt_applicable &
            ONE_LEVEL_ENABLED −>
            (IncAlt_state'=IncAlt_LATENCY_PERIODS) & (IncAlt_go'=false);

    // Tactic applicable, but do not start it
    [IncAlt_pass] sys_go & IncAlt_go & IncAlt_state=0 & IncAlt_applicable
            −> (IncAlt_go'=false);

    // Pass if the tactic is not applicable
    [IncAlt_invalid] sys_go & IncAlt_go & IncAlt_state=0 & !IncAlt_applicable
            −> 1 : (IncAlt_go'=false);

    // Progress of the tactic
    [IncAlt_progress] sys_go & IncAlt_go & IncAlt_state > 1 −> 1:
            (IncAlt_state'=IncAlt_state−1) & (IncAlt_go'=false);

    // Completion of the tactic
    [IncAlt_complete] sys_go & IncAlt_go & IncAlt_state=1 −> 1:
            (IncAlt_state'=0) & (IncAlt_go'=true);

    [tick] !IncAlt_go −> 1: (IncAlt_go'=true);
endmodule


//************************************************************************
// TACTIC: DecAlt
//************************************************************************

const int DecAlt_LATENCY_PERIODS = ceil(DecAlt_LATENCY/PERIOD);

// Applicability conditions
formula DecAlt_applicable = DecAlt_compatible & a > 0;

module DecAlt
    DecAlt_state : [0..DecAlt_LATENCY_PERIODS] init ini_DecAlt_state;
    DecAlt_go : bool init true;

    // Tactic applicable, start it
    [DecAlt_start] sys_go & DecAlt_go & DecAlt_state=0 &
            DecAlt_applicable & ONE_LEVEL_ENABLED −>
            (DecAlt_state'=DecAlt_LATENCY_PERIODS) &
            (DecAlt_go'=false);

    // Tactic applicable, but do not start it
    [DecAlt_pass] sys_go & DecAlt_go & DecAlt_state=0 &
            DecAlt_applicable −> (DecAlt_go'=false);

    // Pass if the tactic is not applicable
    [DecAlt_invalid] sys_go & DecAlt_go & DecAlt_state=0 &
            !DecAlt_applicable −> 1 : (DecAlt_go'=false);

    // Progress of the tactic
    [DecAlt_progress] sys_go & DecAlt_go & DecAlt_state > 1 −> 1:
            (DecAlt_state'=DecAlt_state−1) & (DecAlt_go'=false);

    // Completion of the tactic
    [DecAlt_complete] sys_go & DecAlt_go & DecAlt_state=1 −> 1:
            (DecAlt_state'=0) & (DecAlt_go'=true);

    [tick] !DecAlt_go −> 1: (DecAlt_go'=true);
endmodule

//************************************************************************
// TACTIC: IncAlt2
//************************************************************************

// Applicability conditions
formula IncAlt2_applicable = IncAlt2_compatible & a <
        MAX_ALT_LEVEL−1;

module IncAlt2
    IncAlt2_state : [0..IncAlt_LATENCY_PERIODS] init ini_IncAlt2_state;
    IncAlt2_go : bool init true;

    // Tactic applicable, start it
    [IncAlt2_start] sys_go & IncAlt2_go & IncAlt2_state=0 &
            IncAlt2_applicable & TWO_LEVEL_ENABLED −>
            (IncAlt2_state'=IncAlt_LATENCY_PERIODS) &
            (IncAlt2_go'=false);

    // Tactic applicable, but do not start it
    [IncAlt2_pass] sys_go & IncAlt2_go & IncAlt2_state=0 &
            IncAlt2_applicable −> (IncAlt2_go'=false);

    // Pass if the tactic is not applicable
    [IncAlt2_invalid] sys_go & IncAlt2_go & IncAlt2_state=0 &
            !IncAlt2_applicable −> 1 : (IncAlt2_go'=false);

    // Progress of the tactic
    [IncAlt2_progress] sys_go & IncAlt2_go & IncAlt2_state > 1 −> 1:
            (IncAlt2_state'=IncAlt2_state−1) & (IncAlt2_go'=false);

    // Completion of the tactic
    [IncAlt2_complete] sys_go & IncAlt2_go & IncAlt2_state=1 −> 1:
            (IncAlt2_state'=0) & (IncAlt2_go'=true);

    [tick] !IncAlt2_go −> 1: (IncAlt2_go'=true);
endmodule


//************************************************************************
// TACTIC: DecAlt2
//************************************************************************

const int DecAlt2_LATENCY_PERIODS = ceil(DecAlt_LATENCY/PERIOD);

// Applicability conditions
formula DecAlt2_applicable = DecAlt2_compatible & a > 1;

module DecAlt2
    DecAlt2_state : [0..DecAlt2_LATENCY_PERIODS] init
            ini_DecAlt2_state;
    DecAlt2_go : bool init true;

    // Tactic applicable, start it
    [DecAlt2_start] sys_go & DecAlt2_go & DecAlt2_state=0 &
            DecAlt2_applicable & TWO_LEVEL_ENABLED −>
            (DecAlt2_state'=DecAlt2_LATENCY_PERIODS) &
            (DecAlt2_go'=false);

    // Tactic applicable, but do not start it
    [DecAlt2_pass] sys_go & DecAlt2_go & DecAlt2_state=0 &
            DecAlt2_applicable −> (DecAlt2_go'=false);

    // Pass if the tactic is not applicable
    [DecAlt2_invalid] sys_go & DecAlt2_go & DecAlt2_state=0 &
            !DecAlt2_applicable −> 1 : (DecAlt2_go'=false);

    // Progress of the tactic
    [DecAlt2_progress] sys_go & DecAlt2_go & DecAlt2_state > 1 −> 1:
            (DecAlt2_state'=DecAlt2_state−1) & (DecAlt2_go'=false);

    // Completion of the tactic
    [DecAlt2_complete] sys_go & DecAlt2_go & DecAlt2_state=1 −> 1:
            (DecAlt2_state'=0) & (DecAlt2_go'=true);

    [tick] !DecAlt2_go −> 1: (DecAlt2_go'=true);
endmodule


//******************************
// Utility Function
//******************************
const int LOOSE = 0;
const int TIGHT = 1;
const int EMC_ON = 1;

formula probOfThreat = stateValue;

formula probabilityOfDestruction = probOfThreat
    * ((f = LOOSE) ? 1.0 : (1.0 / destructionFormationFactor))
    * ((c = EMC_ON) ? ecm_threat_prob : 1.0)
        * max(0.0, threatRange − (a + 1)) / threatRange; // +1 because
            level 0 is one level above ground

module constraint // in this case the constraint is surviving
    satisfied: bool init true;
    [tick2] satisfied −> (1.0 − probabilityOfDestruction): (satisfied'=true)
        + probabilityOfDestruction: (satisfied'=false);
    [tick2] !satisfied −> true;
endmodule


formula probOfTarget= stateValue1;
```

```
formula probOfDetection = probOfTarget
    * ((f = LOOSE) ? 1.0 : (1.0 / detectionFormationFactor))
    * ((c = EMC_ON) ? ecm_target_prob : 1.0)
    * max(0.0, sensorRange − (a + 1)) / sensorRange; // +1 because level 0 is one level above ground

module sensor
    targetDetected: bool init false;
    [tick2] true −> probOfDetection: (targetDetected'=true) + (1.0 − probOfDetection): (targetDetected'=false);
endmodule

rewards "util"
    [tack] (time < HORIZON) & satisfied & targetDetected : 1;
    [tack] (time = HORIZON) & satisfied : (targetDetected ? 1 : 0) + survival_reward;

    // give slight preference to not adapting
    [tick] time = 0 & IncAlt_state=ini_IncAlt_state & DecAlt_state=ini_DecAlt_state & a=init_a & f=init_f : 0.000000001;
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 ρ_{mdps} and ρ_{mdpl}.

```
mdp
const double PERIOD = 60;
const int HORIZON = 5; // Planning horizon for deliberative planning
const double IncAlt_LATENCY = 60;
const double DecAlt_LATENCY = 60;
const int MAX_ALT_LEVEL = 3;
const double destructionFormationFactor = 1.5;
const double threatRange = 3;
const double detectionFormationFactor = 1.2;
const double sensorRange = 4;
const init_a = 0;
const init_c = 0;
const init_f = 0;
const bool ECM_ENABLED = true;
const bool ONE_LEVEL_ENABLED = true; // Unlike reactive planning, one level increase/decrease altitude enabled
const bool TWO_LEVEL_ENABLED = true; // Two level increase/decrease altitude also enabled
const int ini_IncAlt_state = 0;
const int ini_DecAlt_state = 0;
const int ini_IncAlt2_state = 0;
const int ini_DecAlt2_state = 0;
const double ecm_threat_prob = 0.15;
const double ecm_target_prob = 0.3;
const double survival_reward = 1;


//***********************************************************************
// CLOCK
//***********************************************************************
const int TO_TICK = 0;
const int TO_TICK2 = 1; // intermediate tick for constraint satisf. update
const int TO_TACK = 2;

label "final" = time = HORIZON & clockstep=TO_TICK;
formula sys_go = clockstep=TO_TICK;

module clk
    time : [0..HORIZON] init 0;
    clockstep : [0..2] init TO_TICK;

    [tick] clockstep=TO_TICK & time < HORIZON −> 1: (time'=time+1) & (clockstep'=TO_TICK2);
    [tick2] clockstep=TO_TICK2 −> 1 : (clockstep'=TO_TACK);
    [tack] clockstep=TO_TACK −> 1: (clockstep'=TO_TICK);
endmodule

module env
s : [0..45] init 0;
[tick] s = 0 −>
    0.034225 : (s' = 1)
    + 0.11655 : (s' = 2)
    + 0.034225 : (s' = 3)
    + 0.11655 : (s' = 4)
    + 0.3969 : (s' = 5)
    + 0.11655 : (s' = 6)
    + 0.034225 : (s' = 7)
    + 0.11655 : (s' = 8)
    + 0.034225 : (s' = 9);
[tick] s = 1 −>
    0.034225 : (s' = 10)
    + 0.11655 : (s' = 11)
    + 0.034225 : (s' = 12)
    + 0.11655 : (s' = 13)
    + 0.3969 : (s' = 14)
    + 0.11655 : (s' = 15)
    + 0.034225 : (s' = 16)
    + 0.11655 : (s' = 17)
```

```
        + 0.034225 : (s’ = 18);
    [tick] s = 2 −>
        0.034225 : (s’ = 10)
        + 0.11655 : (s’ = 11)
        + 0.034225 : (s’ = 12)
        + 0.11655 : (s’ = 13)
        + 0.3969 : (s’ = 14)
        + 0.11655 : (s’ = 15)
        + 0.034225 : (s’ = 16)
        + 0.11655 : (s’ = 17)
        + 0.034225 : (s’ = 18);
    [tick] s = 3 −>
        0.034225 : (s’ = 10)
        + 0.11655 : (s’ = 11)
        + 0.034225 : (s’ = 12)
        + 0.11655 : (s’ = 13)
        + 0.3969 : (s’ = 14)
        + 0.11655 : (s’ = 15)
        + 0.034225 : (s’ = 16)
        + 0.11655 : (s’ = 17)
        + 0.034225 : (s’ = 18);
    [tick] s = 4 −>
        0.034225 : (s’ = 10)
        + 0.11655 : (s’ = 11)
        + 0.034225 : (s’ = 12)
        + 0.11655 : (s’ = 13)
        + 0.3969 : (s’ = 14)
        + 0.11655 : (s’ = 15)
        + 0.034225 : (s’ = 16)
        + 0.11655 : (s’ = 17)
        + 0.034225 : (s’ = 18);
    [tick] s = 5 −>
        0.034225 : (s’ = 10)
        + 0.11655 : (s’ = 11)
        + 0.034225 : (s’ = 12)
        + 0.11655 : (s’ = 13)
        + 0.3969 : (s’ = 14)
        + 0.11655 : (s’ = 15)
        + 0.034225 : (s’ = 16)
        + 0.11655 : (s’ = 17)
        + 0.034225 : (s’ = 18);
    [tick] s = 6 −>
        0.034225 : (s’ = 10)
        + 0.11655 : (s’ = 11)
        + 0.034225 : (s’ = 12)
        + 0.11655 : (s’ = 13)
        + 0.3969 : (s’ = 14)
        + 0.11655 : (s’ = 15)
        + 0.034225 : (s’ = 16)
        + 0.11655 : (s’ = 17)
        + 0.034225 : (s’ = 18);
    [tick] s = 7 −>
        0.034225 : (s’ = 10)
        + 0.11655 : (s’ = 11)
        + 0.034225 : (s’ = 12)
        + 0.11655 : (s’ = 13)
        + 0.3969 : (s’ = 14)
        + 0.11655 : (s’ = 15)
        + 0.034225 : (s’ = 16)
        + 0.11655 : (s’ = 17)
        + 0.034225 : (s’ = 18);
    [tick] s = 8 −>
        0.034225 : (s’ = 10)
        + 0.11655 : (s’ = 11)
        + 0.034225 : (s’ = 12)
        + 0.11655 : (s’ = 13)
        + 0.3969 : (s’ = 14)
        + 0.11655 : (s’ = 15)
        + 0.034225 : (s’ = 16)
        + 0.11655 : (s’ = 17)
        + 0.034225 : (s’ = 18);
    [tick] s = 9 −>
        0.034225 : (s’ = 10)
        + 0.11655 : (s’ = 11)
        + 0.034225 : (s’ = 12)
        + 0.11655 : (s’ = 13)
        + 0.3969 : (s’ = 14)
        + 0.11655 : (s’ = 15)
        + 0.034225 : (s’ = 16)
        + 0.11655 : (s’ = 17)
        + 0.034225 : (s’ = 18);
    [tick] s = 10 −>
        0.034225 : (s’ = 19)
        + 0.11655 : (s’ = 20)
        + 0.034225 : (s’ = 21)
        + 0.11655 : (s’ = 22)
        + 0.3969 : (s’ = 23)
        + 0.11655 : (s’ = 24)
        + 0.034225 : (s’ = 25)
        + 0.11655 : (s’ = 26)
        + 0.034225 : (s’ = 27);
    [tick] s = 11 −>
        0.034225 : (s’ = 19)
        + 0.11655 : (s’ = 20)
        + 0.034225 : (s’ = 21)
        + 0.11655 : (s’ = 22)
        + 0.3969 : (s’ = 23)
        + 0.11655 : (s’ = 24)
        + 0.034225 : (s’ = 25)
        + 0.11655 : (s’ = 26)
        + 0.034225 : (s’ = 27);
    [tick] s = 12 −>
        0.034225 : (s’ = 19)
        + 0.11655 : (s’ = 20)
        + 0.034225 : (s’ = 21)
        + 0.11655 : (s’ = 22)
        + 0.3969 : (s’ = 23)
        + 0.11655 : (s’ = 24)
        + 0.034225 : (s’ = 25)
        + 0.11655 : (s’ = 26)
        + 0.034225 : (s’ = 27);
    [tick] s = 13 −>
        0.034225 : (s’ = 19)
        + 0.11655 : (s’ = 20)
        + 0.034225 : (s’ = 21)
        + 0.11655 : (s’ = 22)
        + 0.3969 : (s’ = 23)
        + 0.11655 : (s’ = 24)
        + 0.034225 : (s’ = 25)
        + 0.11655 : (s’ = 26)
        + 0.034225 : (s’ = 27);
    [tick] s = 14 −>
        0.034225 : (s’ = 19)
        + 0.11655 : (s’ = 20)
        + 0.034225 : (s’ = 21)
        + 0.11655 : (s’ = 22)
        + 0.3969 : (s’ = 23)
        + 0.11655 : (s’ = 24)
        + 0.034225 : (s’ = 25)
        + 0.11655 : (s’ = 26)
        + 0.034225 : (s’ = 27);
    [tick] s = 15 −>
        0.034225 : (s’ = 19)
        + 0.11655 : (s’ = 20)
        + 0.034225 : (s’ = 21)
        + 0.11655 : (s’ = 22)
        + 0.3969 : (s’ = 23)
        + 0.11655 : (s’ = 24)
        + 0.034225 : (s’ = 25)
        + 0.11655 : (s’ = 26)
        + 0.034225 : (s’ = 27);
    [tick] s = 16 −>
        0.034225 : (s’ = 19)
        + 0.11655 : (s’ = 20)
        + 0.034225 : (s’ = 21)
        + 0.11655 : (s’ = 22)
        + 0.3969 : (s’ = 23)
        + 0.11655 : (s’ = 24)
        + 0.034225 : (s’ = 25)
        + 0.11655 : (s’ = 26)
        + 0.034225 : (s’ = 27);
    [tick] s = 17 −>
        0.034225 : (s’ = 19)
        + 0.11655 : (s’ = 20)
        + 0.034225 : (s’ = 21)
        + 0.11655 : (s’ = 22)
        + 0.3969 : (s’ = 23)
        + 0.11655 : (s’ = 24)
        + 0.034225 : (s’ = 25)
        + 0.11655 : (s’ = 26)
        + 0.034225 : (s’ = 27);
    [tick] s = 18 −>
        0.034225 : (s’ = 19)
        + 0.11655 : (s’ = 20)
        + 0.034225 : (s’ = 21)
        + 0.11655 : (s’ = 22)
```

```
        + 0.3969 : (s' = 23)
        + 0.11655 : (s' = 24)
        + 0.034225 : (s' = 25)
        + 0.11655 : (s' = 26)
        + 0.034225 : (s' = 27);
    [tick] s = 19 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 20 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 21 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 22 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 23 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 24 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 25 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 26 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 27 ->
        0.034225 : (s' = 28)
        + 0.11655 : (s' = 29)
        + 0.034225 : (s' = 30)
        + 0.11655 : (s' = 31)
        + 0.3969 : (s' = 32)
        + 0.11655 : (s' = 33)
        + 0.034225 : (s' = 34)
        + 0.11655 : (s' = 35)
        + 0.034225 : (s' = 36);
    [tick] s = 28 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 29 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 30 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 31 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 32 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 33 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 34 ->
        0.034225 : (s' = 37)
        + 0.11655 : (s' = 38)
        + 0.034225 : (s' = 39)
        + 0.11655 : (s' = 40)
        + 0.3969 : (s' = 41)
        + 0.11655 : (s' = 42)
        + 0.034225 : (s' = 43)
        + 0.11655 : (s' = 44)
        + 0.034225 : (s' = 45);
    [tick] s = 35 ->
```

```
    0.034225 : (s' = 37)
    + 0.11655 : (s' = 38)
    + 0.034225 : (s' = 39)
    + 0.11655 : (s' = 40)
    + 0.3969 : (s' = 41)
    + 0.11655 : (s' = 42)
    + 0.034225 : (s' = 43)
    + 0.11655 : (s' = 44)
    + 0.034225 : (s' = 45);
[tick] s = 36 −>
    0.034225 : (s' = 37)
    + 0.11655 : (s' = 38)
    + 0.034225 : (s' = 39)
    + 0.11655 : (s' = 40)
    + 0.3969 : (s' = 41)
    + 0.11655 : (s' = 42)
    + 0.034225 : (s' = 43)
    + 0.11655 : (s' = 44)
    + 0.034225 : (s' = 45);
endmodule

// environment has 2 components. statevalue for threats and statevalue1 is
      for targets
formula stateValue = (s = 0 ? 0 : 0) +
                     (s = 1 ? 0.00605639 : 0) +
                     (s = 2 ? 0.00605639 : 0) +
                     (s = 3 ? 0.00605639 : 0) +
                     (s = 4 ? 0.0282836 : 0) +
                     (s = 5 ? 0.0282836 : 0) +
                     (s = 6 ? 0.0282836 : 0) +
                     (s = 7 ? 0.0778979 : 0) +
                     (s = 8 ? 0.0778979 : 0) +
                     (s = 9 ? 0.0778979 : 0) +
                     (s = 10 ? 0 : 0) +
                     (s = 11 ? 0 : 0) +
                     (s = 12 ? 0 : 0) +
                     (s = 13 ? 0 : 0) +
                     (s = 14 ? 0 : 0) +
                     (s = 15 ? 0 : 0) +
                     (s = 16 ? 0 : 0) +
                     (s = 17 ? 0 : 0) +
                     (s = 18 ? 0 : 0) +
                     (s = 19 ? 0.750751 : 0) +
                     (s = 20 ? 0.750751 : 0) +
                     (s = 21 ? 0.750751 : 0) +
                     (s = 22 ? 0.885424 : 0) +
                     (s = 23 ? 0.885424 : 0) +
                     (s = 24 ? 0.885424 : 0) +
                     (s = 25 ? 0.963485 : 0) +
                     (s = 26 ? 0.963485 : 0) +
                     (s = 27 ? 0.963485 : 0) +
                     (s = 28 ? 0.435626 : 0) +
                     (s = 29 ? 0.435626 : 0) +
                     (s = 30 ? 0.435626 : 0) +
                     (s = 31 ? 0.676196 : 0) +
                     (s = 32 ? 0.676196 : 0) +
                     (s = 33 ? 0.676196 : 0) +
                     (s = 34 ? 0.864925 : 0) +
                     (s = 35 ? 0.864925 : 0) +
                     (s = 36 ? 0.864925 : 0) +
                     (s = 37 ? 0.368403 : 0) +
                     (s = 38 ? 0.368403 : 0) +
                     (s = 39 ? 0.368403 : 0) +
                     (s = 40 ? 0.793701 : 0) +
                     (s = 41 ? 0.793701 : 0) +
                     (s = 42 ? 0.793701 : 0) +
                     (s = 43 ? 0.983048 : 0) +
                     (s = 44 ? 0.983048 : 0) +
                     (s = 45 ? 0.983048 : 0);

formula stateValue1 = (s = 0 ? 0 : 0) +
                     (s = 1 ? 0.830037 : 0) +
                     (s = 2 ? 0.904439 : 0) +
                     (s = 3 ? 0.954777 : 0) +
                     (s = 4 ? 0.830037 : 0) +
                     (s = 5 ? 0.904439 : 0) +
                     (s = 6 ? 0.954777 : 0) +
                     (s = 7 ? 0.830037 : 0) +
                     (s = 8 ? 0.904439 : 0) +
                     (s = 9 ? 0.954777 : 0) +
                     (s = 10 ? 0 : 0) +
                     (s = 11 ? 0 : 0) +
                     (s = 12 ? 0 : 0) +
                     (s = 13 ? 0 : 0) +
                     (s = 14 ? 0 : 0) +
                     (s = 15 ? 0 : 0) +
                     (s = 16 ? 0 : 0) +
                     (s = 17 ? 0 : 0) +
                     (s = 18 ? 0 : 0) +
                     (s = 19 ? 0.0156741 : 0) +
                     (s = 20 ? 0.0719057 : 0) +
                     (s = 21 ? 0.190204 : 0) +
                     (s = 22 ? 0.0156741 : 0) +
                     (s = 23 ? 0.0719057 : 0) +
                     (s = 24 ? 0.190204 : 0) +
                     (s = 25 ? 0.0156741 : 0) +
                     (s = 26 ? 0.0719057 : 0) +
                     (s = 27 ? 0.190204 : 0) +
                     (s = 28 ? 0 : 0) +
                     (s = 29 ? 0 : 0) +
                     (s = 30 ? 0 : 0) +
                     (s = 31 ? 0 : 0) +
                     (s = 32 ? 0 : 0) +
                     (s = 33 ? 0 : 0) +
                     (s = 34 ? 0 : 0) +
                     (s = 35 ? 0 : 0) +
                     (s = 36 ? 0 : 0) +
                     (s = 37 ? 0 : 0) +
                     (s = 38 ? 0 : 0) +
                     (s = 39 ? 0 : 0) +
                     (s = 40 ? 0 : 0) +
                     (s = 41 ? 0 : 0) +
                     (s = 42 ? 0 : 0) +
                     (s = 43 ? 0 : 0) +
                     (s = 44 ? 0 : 0) +
                     (s = 45 ? 0 : 0);
// #ENV ENDS


//***********************************************************************
// SYSTEM
//***********************************************************************

// Variable range and initialization
const a_MIN=0; const a_MAX=MAX_ALT_LEVEL; const a_INIT=init_a;
const f_MIN=0; const f_MAX=1; const f_INIT=init_f;
const c_MIN=0; const c_MAX=1; const c_INIT=init_c;

module sys
    a : [a_MIN..a_MAX] init a_INIT;
    f : [f_MIN..f_MAX] init f_INIT;
    c : [c_MIN..c_MAX] init c_INIT;

    [EcmOn_start] c=0 & ECM_ENABLED −> 1: (c'=c_EcmOn_impact);
    [EcmOff_start] c=1 & ECM_ENABLED −> 1: (c'=c_EcmOff_impact);

    [GoTight_start] f=0 −> 1: (a'=a_GoTight_impact)
         & (f'=f_GoTight_impact);
    [GoLoose_start] f=1 −> 1: (a'=a_GoLoose_impact)
         & (f'=f_GoLoose_impact);

    [IncAlt_complete] a < MAX_ALT_LEVEL & ONE_LEVEL_ENABLED −>
           1: (a'=a_IncAlt_impact)
        & (f'=f_IncAlt_impact);
    [IncAlt2_complete] a < MAX_ALT_LEVEL−1 &
           TWO_LEVEL_ENABLED −> 1: (a'=a_IncAlt2_impact);

    [DecAlt_complete] a > 0 & ONE_LEVEL_ENABLED −> 1:
           (a'=a_DecAlt_impact)
         & (f'=f_DecAlt_impact);
    [DecAlt2_complete] a > 1 & TWO_LEVEL_ENABLED −> 1:
           (a'=a_DecAlt2_impact);
endmodule


formula c_EcmOn_impact = c + (1) >= c_MIN ? ( c+(1)<=c_MAX? c+(1) :
      c_MAX) : c_MIN;
formula c_EcmOff_impact = c + (−1) >= c_MIN ? ( c+(−1)<=c_MAX?
      c+(−1) : c_MAX) : c_MIN;
formula a_GoTight_impact = a + (0) >= a_MIN ? ( a+(0)<=a_MAX? a+(0) :
      a_MAX) : a_MIN;
formula f_GoTight_impact = f + (1) >= f_MIN ? ( f+(1)<=f_MAX? f+(1) :
      f_MAX) : f_MIN;
formula a_GoLoose_impact = a + (0) >= a_MIN ? ( a+(0)<=a_MAX? a+(0) :
      a_MAX) : a_MIN;
formula f_GoLoose_impact = f + (−1) >= f_MIN ? ( f+(−1)<=f_MAX? f+(−1)
```

```
        : f_MAX) : f_MIN;
formula a_IncAlt_impact = a + (1) >= a_MIN ? ( a+(1)<=a_MAX? a+(1) :
        a_MAX) : a_MIN;
formula f_IncAlt_impact = f + (0) >= f_MIN ? ( f+(0)<=f_MAX? f+(0) :
        f_MAX) : f_MIN;
formula a_DecAlt_impact = a + (-1) >= a_MIN ? ( a+(-1)<=a_MAX?
        a+(-1) : a_MAX) : a_MIN;
formula f_DecAlt_impact = f + (0) >= f_MIN ? ( f+(0)<=f_MAX? f+(0) :
        f_MAX) : f_MIN;
formula a_IncAlt2_impact = a + (2) >= a_MIN ? ( a+(2)<=a_MAX? a+(2) :
        a_MAX) : a_MIN;
formula a_DecAlt2_impact = a + (-2) >= a_MIN ? ( a+(-2)<=a_MAX?
        a+(-2) : a_MAX) : a_MIN;

// tactic concurrency rules
formula IncAlt_used = IncAlt_state != 0;
formula DecAlt_used = DecAlt_state != 0;
formula IncAlt2_used = IncAlt2_state != 0;
formula DecAlt2_used = DecAlt2_state != 0;

formula EcmOn_compatible = !EcmOn_used;
formula EcmOff_compatible = !EcmOff_used;
formula GoTight_compatible = !GoLoose_used;
formula GoLoose_compatible = !GoTight_used;
formula IncAlt_compatible = (!DecAlt_used) & (!IncAlt2_used) &
        (!DecAlt2_used);
formula DecAlt_compatible = (!IncAlt_used) & (!IncAlt2_used) &
        (!DecAlt2_used);
formula IncAlt2_compatible = (!DecAlt_used) & (!IncAlt_used) &
        (!DecAlt2_used);
formula DecAlt2_compatible = (!DecAlt_used) & (!IncAlt_used) &
        (!IncAlt2_used);


//*********************************************************************
// TACTIC: EcmOn
//*********************************************************************

// Applicability conditions
formula EcmOn_applicable = EcmOn_compatible & c=0;

module EcmOn
    EcmOn_used : bool init false;
    EcmOn_go : bool init true;

    // Tactic applicable, start it
    [EcmOn_start] sys_go & EcmOn_go & EcmOn_applicable &
            ECM_ENABLED -> (EcmOn_used'=true) & (EcmOn_go'=false);

    // Tactic applicable, but do not start it
    [EcmOn_pass] sys_go & EcmOn_go & EcmOn_applicable ->
            (EcmOn_go'=false);

    // Pass if the tactic is not applicable
    [EcmOn_invalid] sys_go & EcmOn_go & !EcmOn_applicable -> 1 :
            (EcmOn_go'=false);

    [tick] !EcmOn_go -> 1: (EcmOn_go'=true) & (EcmOn_used'=false);
endmodule


//*********************************************************************
// TACTIC: EcmOff
//*********************************************************************

// Applicability conditions
formula EcmOff_applicable = EcmOff_compatible & c=1;

module EcmOff
    EcmOff_used : bool init false;
    EcmOff_go : bool init true;

    // Tactic applicable, start it
    [EcmOff_start] sys_go & EcmOff_go & EcmOff_applicable &
            ECM_ENABLED -> (EcmOff_used'=true) & (EcmOff_go'=false);

    // Tactic applicable, but do not start it
    [EcmOff_pass] sys_go & EcmOff_go & EcmOff_applicable ->
            (EcmOff_go'=false);

    // Pass if the tactic is not applicable
    [EcmOff_invalid] sys_go & EcmOff_go & !EcmOff_applicable -> 1 :
            (EcmOff_go'=false);

    [tick] !EcmOff_go -> 1: (EcmOff_go'=true) & (EcmOff_used'=false);
endmodule


//*********************************************************************
// TACTIC: GoTight
//*********************************************************************

// Applicability conditions
formula GoTight_applicable = GoTight_compatible & f=0;

module GoTight
    GoTight_used : bool init false;
    GoTight_go : bool init true;

    // Tactic applicable, start it
    [GoTight_start] sys_go & GoTight_go & GoTight_applicable ->
            (GoTight_used'=true) & (GoTight_go'=false);

    // Tactic applicable, but do not start it
    [GoTight_pass] sys_go & GoTight_go & GoTight_applicable ->
            (GoTight_go'=false);

    // Pass if the tactic is not applicable
    [GoTight_invalid] sys_go & GoTight_go & !GoTight_applicable -> 1 :
            (GoTight_go'=false);

    [tick] !GoTight_go -> 1: (GoTight_go'=true) & (GoTight_used'=false);
endmodule


//*********************************************************************
// TACTIC: GoLoose
//*********************************************************************

// Applicability conditions
formula GoLoose_applicable = GoLoose_compatible & f=1;

module GoLoose
    GoLoose_used : bool init false;
    GoLoose_go : bool init true;

    // Tactic applicable, start it
    [GoLoose_start] sys_go & GoLoose_go & GoLoose_applicable ->
            (GoLoose_used'=true) & (GoLoose_go'=false);

    // Tactic applicable, but do not start it
    [GoLoose_pass] sys_go & GoLoose_go & GoLoose_applicable ->
            (GoLoose_go'=false);

    // Pass if the tactic is not applicable
    [GoLoose_invalid] sys_go & GoLoose_go & !GoLoose_applicable -> 1 :
            (GoLoose_go'=false);

    [tick] !GoLoose_go -> 1: (GoLoose_go'=true) &
            (GoLoose_used'=false);
endmodule


//*********************************************************************
// TACTIC: IncAlt
//*********************************************************************

const int IncAlt_LATENCY_PERIODS = ceil(IncAlt_LATENCY/PERIOD);

// Applicability conditions
formula IncAlt_applicable = IncAlt_compatible & a < MAX_ALT_LEVEL;

module IncAlt
    IncAlt_state : [0..IncAlt_LATENCY_PERIODS] init ini_IncAlt_state;
    IncAlt_go : bool init true;

    // Tactic applicable, start it
    [IncAlt_start] sys_go & IncAlt_go & IncAlt_state=0 & IncAlt_applicable &
            ONE_LEVEL_ENABLED ->
            (IncAlt_state'=IncAlt_LATENCY_PERIODS) & (IncAlt_go'=false);

    // Tactic applicable, but do not start it
    [IncAlt_pass] sys_go & IncAlt_go & IncAlt_state=0 & IncAlt_applicable
            -> (IncAlt_go'=false);

    // Pass if the tactic is not applicable
```

```
    [IncAlt_invalid] sys_go & IncAlt_go & IncAlt_state=0 & !IncAlt_applicable
        -> 1 : (IncAlt_go'=false);

    // Progress of the tactic
    [IncAlt_progress] sys_go & IncAlt_go & IncAlt_state > 1 -> 1:
        (IncAlt_state'=IncAlt_state-1) & (IncAlt_go'=false);

    // Completion of the tactic
    [IncAlt_complete] sys_go & IncAlt_go & IncAlt_state=1 -> 1:
        (IncAlt_state'=0) & (IncAlt_go'=true);

    [tick] !IncAlt_go -> 1: (IncAlt_go'=true);
endmodule


//*********************************************************************
// TACTIC: DecAlt
//*********************************************************************

const int DecAlt_LATENCY_PERIODS = ceil(DecAlt_LATENCY/PERIOD);

// Applicability conditions
formula DecAlt_applicable = DecAlt_compatible & a > 0;

module DecAlt
    DecAlt_state : [0..DecAlt_LATENCY_PERIODS] init ini_DecAlt_state;
    DecAlt_go : bool init true;

    // Tactic applicable, start it
    [DecAlt_start] sys_go & DecAlt_go & DecAlt_state=0 &
        DecAlt_applicable & ONE_LEVEL_ENABLED ->
        (DecAlt_state'=DecAlt_LATENCY_PERIODS) &
        (DecAlt_go'=false);

    // Tactic applicable, but do not start it
    [DecAlt_pass] sys_go & DecAlt_go & DecAlt_state=0 &
        DecAlt_applicable -> (DecAlt_go'=false);

    // Pass if the tactic is not applicable
    [DecAlt_invalid] sys_go & DecAlt_go & DecAlt_state=0 &
        !DecAlt_applicable -> 1 : (DecAlt_go'=false);

    // Progress of the tactic
    [DecAlt_progress] sys_go & DecAlt_go & DecAlt_state > 1 -> 1:
        (DecAlt_state'=DecAlt_state-1) & (DecAlt_go'=false);

    // Completion of the tactic
    [DecAlt_complete] sys_go & DecAlt_go & DecAlt_state=1 -> 1:
        (DecAlt_state'=0) & (DecAlt_go'=true);

    [tick] !DecAlt_go -> 1: (DecAlt_go'=true);
endmodule

//*********************************************************************
// TACTIC: IncAlt2
//*********************************************************************

// Applicability conditions
formula IncAlt2_applicable = IncAlt2_compatible & a <
    MAX_ALT_LEVEL-1;

module IncAlt2
    IncAlt2_state : [0..IncAlt_LATENCY_PERIODS] init ini_IncAlt2_state;
    IncAlt2_go : bool init true;

    // Tactic applicable, start it
    [IncAlt2_start] sys_go & IncAlt2_go & IncAlt2_state=0 &
        IncAlt2_applicable & TWO_LEVEL_ENABLED ->
        (IncAlt2_state'=IncAlt_LATENCY_PERIODS) &
        (IncAlt2_go'=false);

    // Tactic applicable, but do not start it
    [IncAlt2_pass] sys_go & IncAlt2_go & IncAlt2_state=0 &
        IncAlt2_applicable -> (IncAlt2_go'=false);

    // Pass if the tactic is not applicable
    [IncAlt2_invalid] sys_go & IncAlt2_go & IncAlt2_state=0 &
        !IncAlt2_applicable -> 1 : (IncAlt2_go'=false);

    // Progress of the tactic
    [IncAlt2_progress] sys_go & IncAlt2_go & IncAlt2_state > 1 -> 1:
        (IncAlt2_state'=IncAlt2_state-1) & (IncAlt2_go'=false);

    // Completion of the tactic
    [IncAlt2_complete] sys_go & IncAlt2_go & IncAlt2_state=1 -> 1:
        (IncAlt2_state'=0) & (IncAlt2_go'=true);

    [tick] !IncAlt2_go -> 1: (IncAlt2_go'=true);
endmodule


//*********************************************************************
// TACTIC: DecAlt2
//*********************************************************************

const int DecAlt2_LATENCY_PERIODS = ceil(DecAlt_LATENCY/PERIOD);

// Applicability conditions
formula DecAlt2_applicable = DecAlt2_compatible & a > 1;

module DecAlt2
    DecAlt2_state : [0..DecAlt2_LATENCY_PERIODS] init
        ini_DecAlt2_state;
    DecAlt2_go : bool init true;

    // Tactic applicable, start it
    [DecAlt2_start] sys_go & DecAlt2_go & DecAlt2_state=0 &
        DecAlt2_applicable & TWO_LEVEL_ENABLED ->
        (DecAlt2_state'=DecAlt2_LATENCY_PERIODS) &
        (DecAlt2_go'=false);

    // Tactic applicable, but do not start it
    [DecAlt2_pass] sys_go & DecAlt2_go & DecAlt2_state=0 &
        DecAlt2_applicable -> (DecAlt2_go'=false);

    // Pass if the tactic is not applicable
    [DecAlt2_invalid] sys_go & DecAlt2_go & DecAlt2_state=0 &
        !DecAlt2_applicable -> 1 : (DecAlt2_go'=false);

    // Progress of the tactic
    [DecAlt2_progress] sys_go & DecAlt2_go & DecAlt2_state > 1 -> 1:
        (DecAlt2_state'=DecAlt2_state-1) & (DecAlt2_go'=false);

    // Completion of the tactic
    [DecAlt2_complete] sys_go & DecAlt2_go & DecAlt2_state=1 -> 1:
        (DecAlt2_state'=0) & (DecAlt2_go'=true);

    [tick] !DecAlt2_go -> 1: (DecAlt2_go'=true);
endmodule


//*****************************
// Utility Function
//*****************************
const int LOOSE = 0;
const int TIGHT = 1;
const int EMC_ON = 1;

formula probOfThreat = stateValue;

formula probabilityOfDestruction = probOfThreat
    * ((f = LOOSE) ? 1.0 : (1.0 / destructionFormationFactor))
    * ((c = EMC_ON) ? ecm_threat_prob : 1.0)
        * max(0.0, threatRange - (a + 1)) / threatRange; // +1 because
            level 0 is one level above ground

module constraint // in this case the constraint is surviving
    satisfied: bool init true;
    [tick2] satisfied -> (1.0 - probabilityOfDestruction): (satisfied'=true)
        + probabilityOfDestruction: (satisfied'=false);
    [tick2] !satisfied -> true;
endmodule


formula probOfTarget= stateValue1;

formula probOfDetection = probOfTarget
    * ((f = LOOSE) ? 1.0 : (1.0 / detectionFormationFactor))
    * ((c = EMC_ON) ? ecm_target_prob : 1.0)
    * max(0.0, sensorRange - (a + 1)) / sensorRange; // +1 because level 0
        is one level above ground

module sensor
    targetDetected: bool init false;
    [tick2] true -> probOfDetection: (targetDetected'=true) + (1.0 -
        probOfDetection): (targetDetected'=false);
```

```
endmodule

rewards "util"
    //[tack] satisfied & targetDetected : 1;

    [tack] (time < HORIZON) & satisfied & targetDetected : 1;
    [tack] (time = HORIZON) & satisfied : (targetDetected ? 1 : 0) +
            survival_reward;

    // give slight preference to not adapting
    [tick] time = 0 & IncAlt_state=ini_IncAlt_state &
            DecAlt_state=ini_DecAlt_state & a=init_a & f=init_f :
            0.000000001;
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.

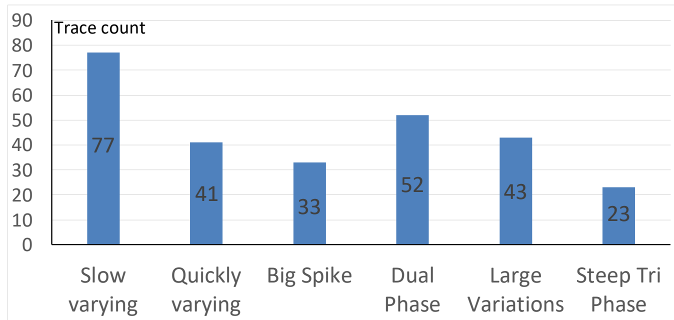
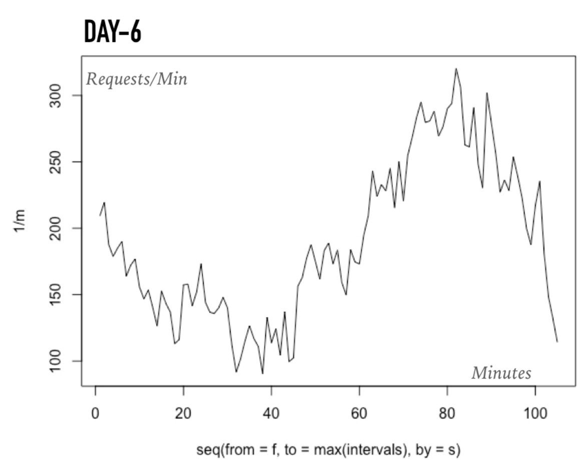


Figure 2. Patterns observed in traces. A single trace can have multiple patterns.

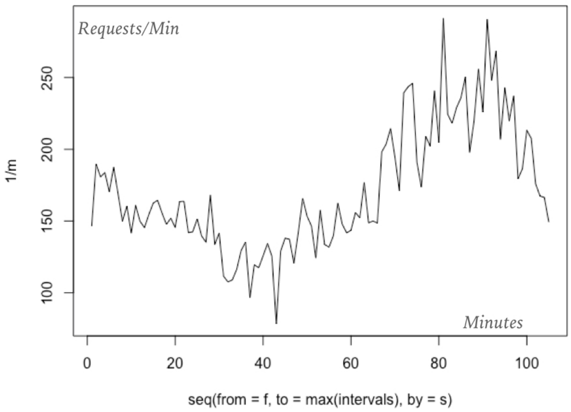
Total traces were 92 but 5 were incomplete/corrupted. Therefore, we used 87 traces for validation. Each trace is scaled such that

- 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.

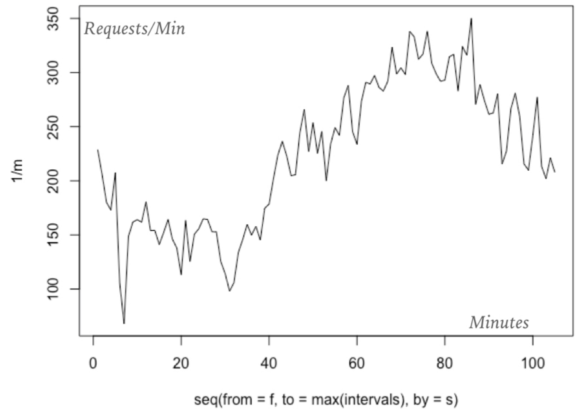
Below are the visualizations of load patterns for each day.



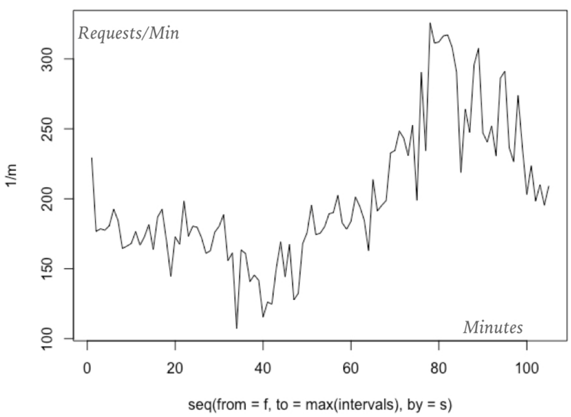
DAY-7



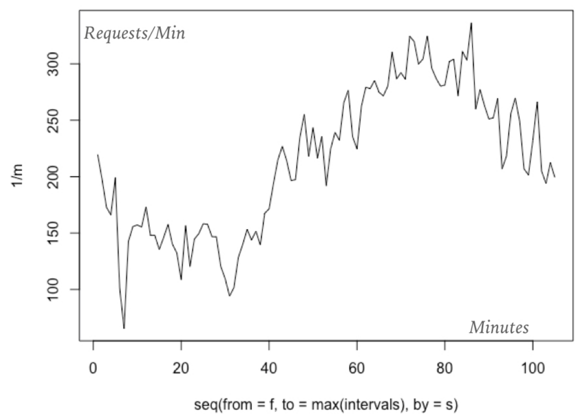
DAY-10



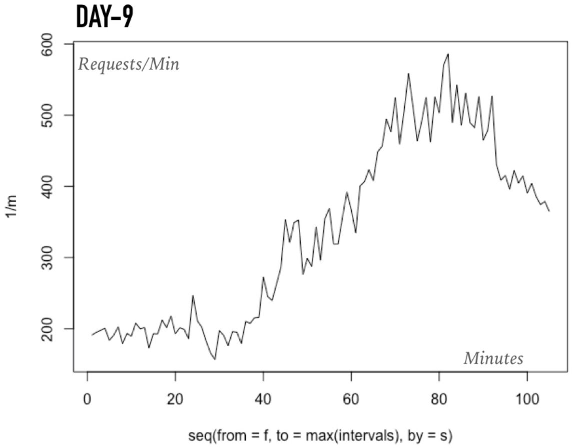
DAY-8



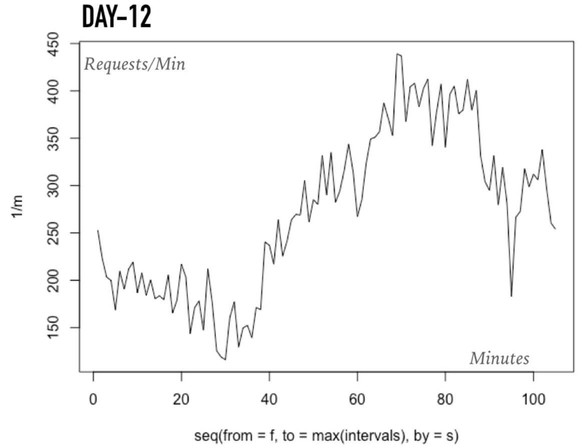
DAY-11



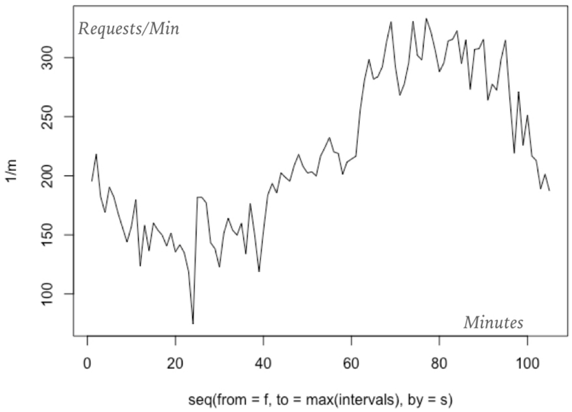
DAY-9



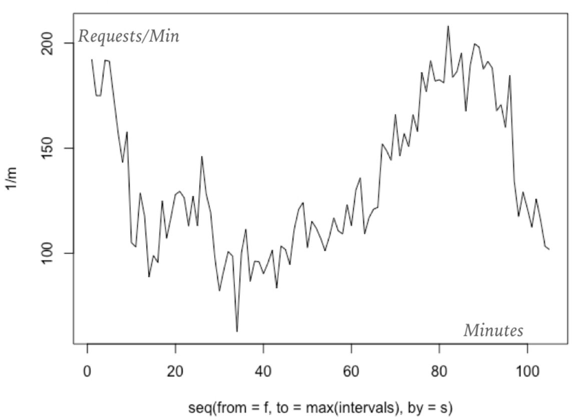
DAY-12



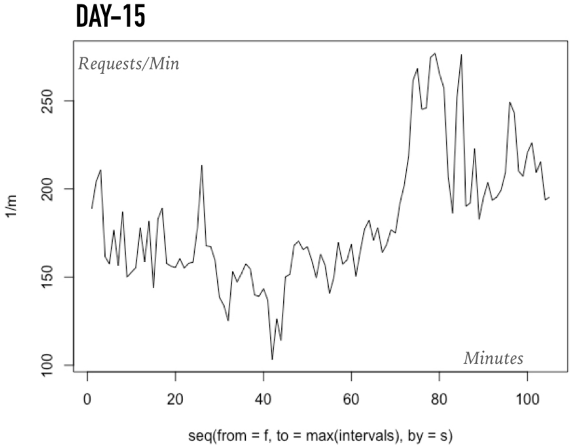
DAY-13



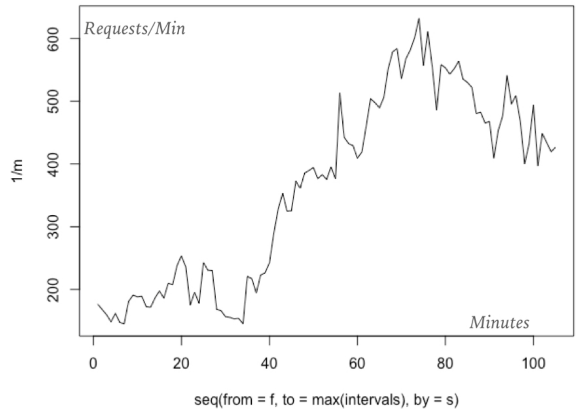
DAY-14



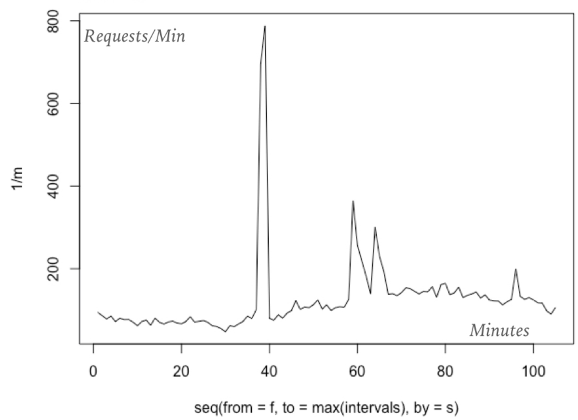
DAY-15



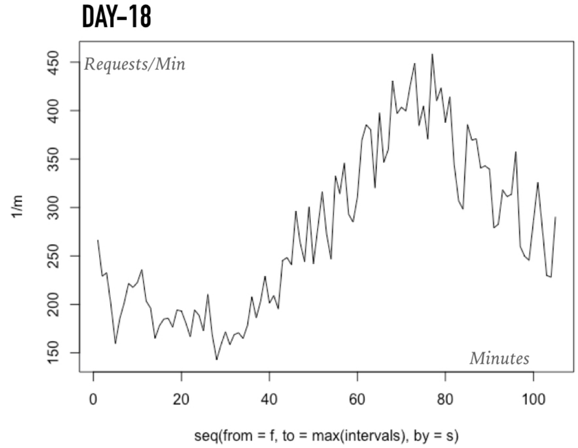
DAY-16



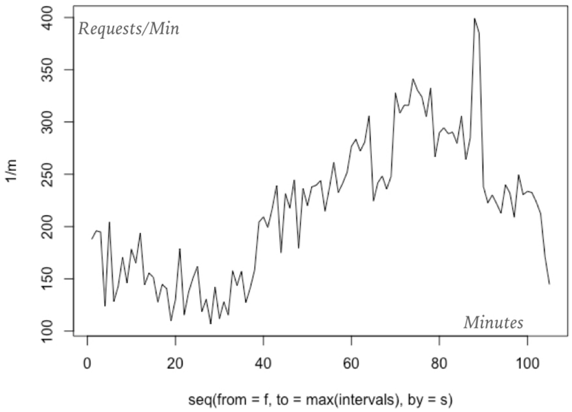
DAY-17



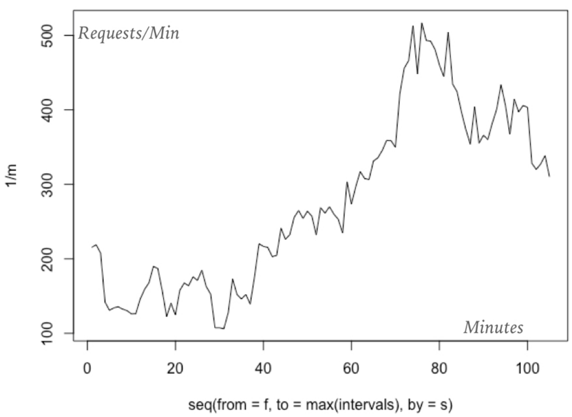
DAY-18



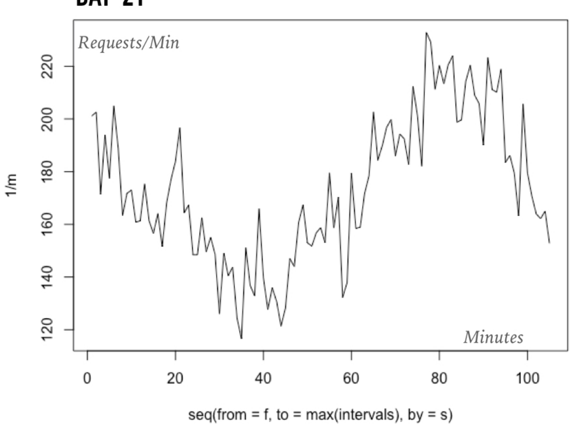
DAY-19



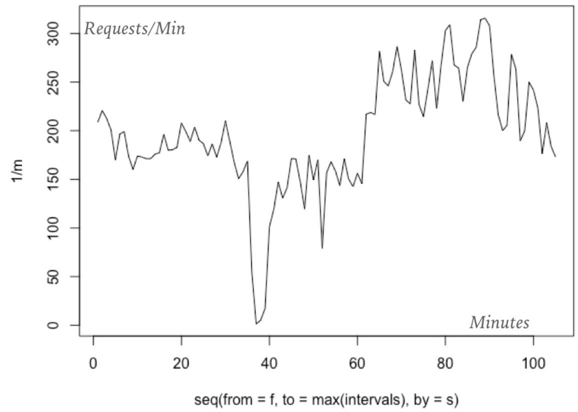
DAY-20



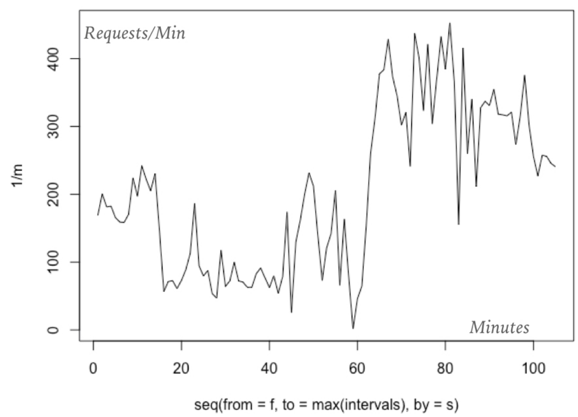
DAY-21



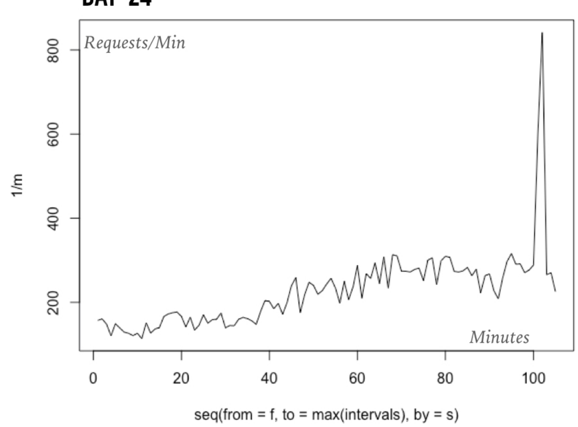
DAY-22



DAY-23

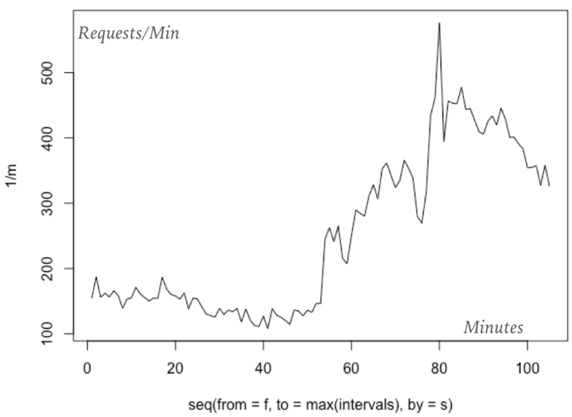


DAY-24

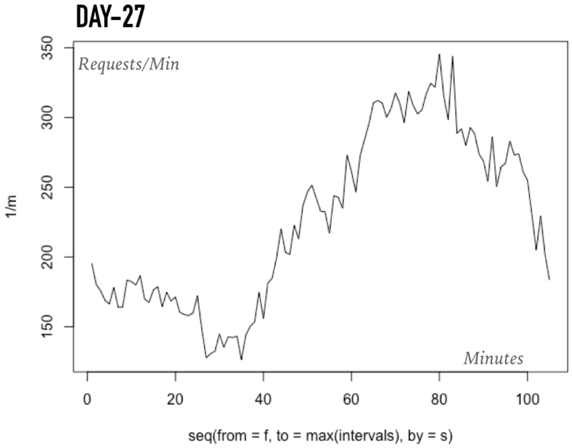


DAY-25

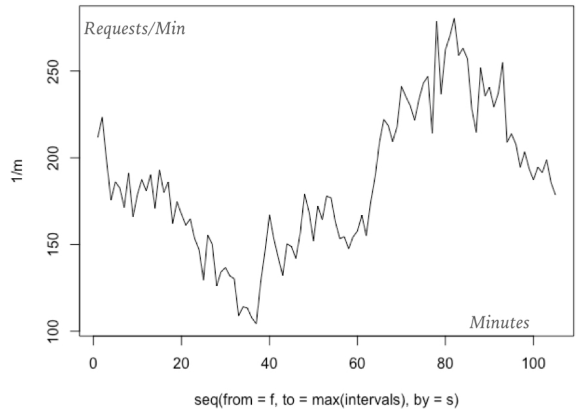
DAY-26



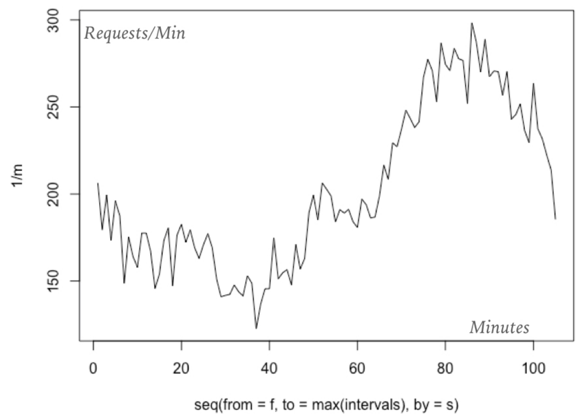
DAY-27



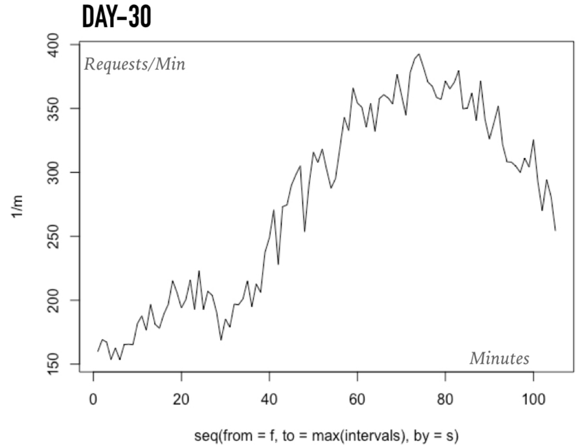
DAY-28



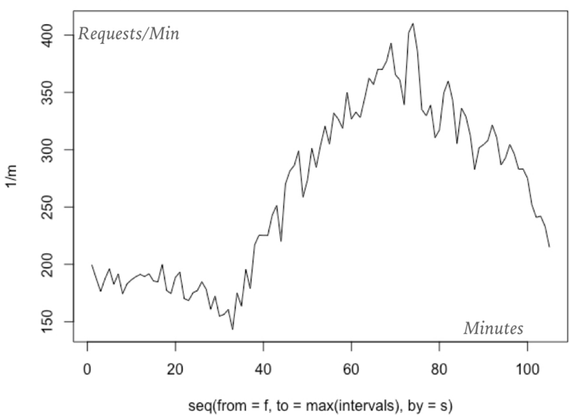
DAY-29



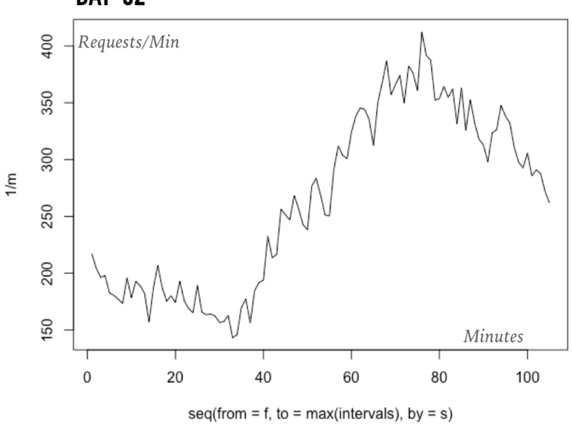
DAY-30



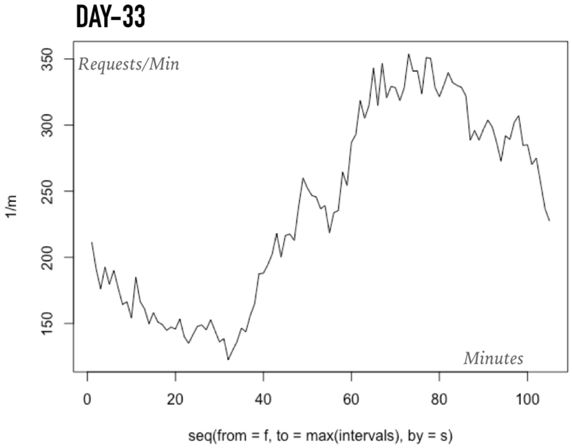
DAY-31



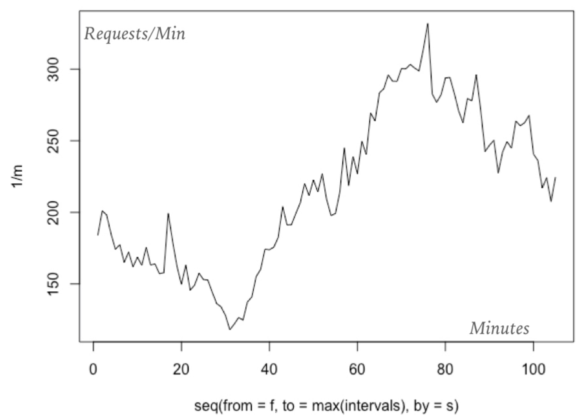
DAY-32



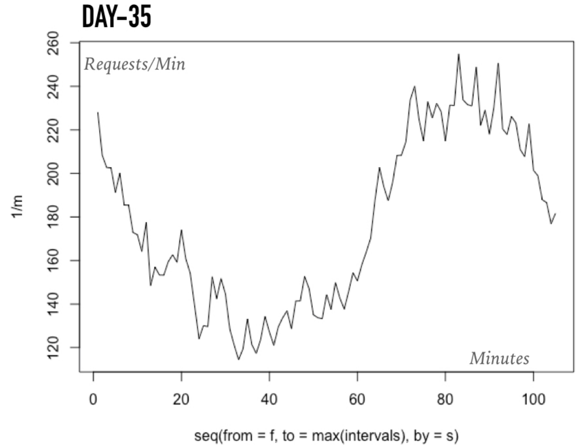
DAY-33



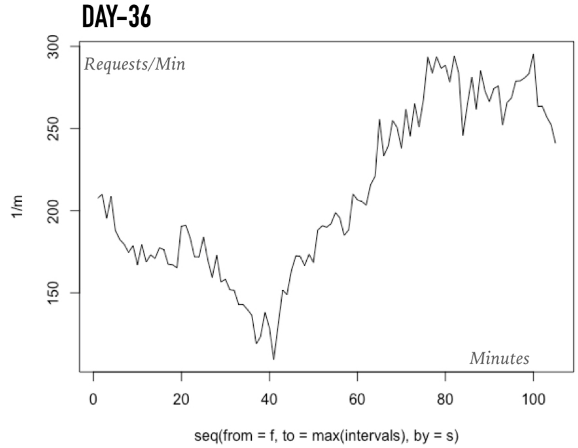
DAY-34



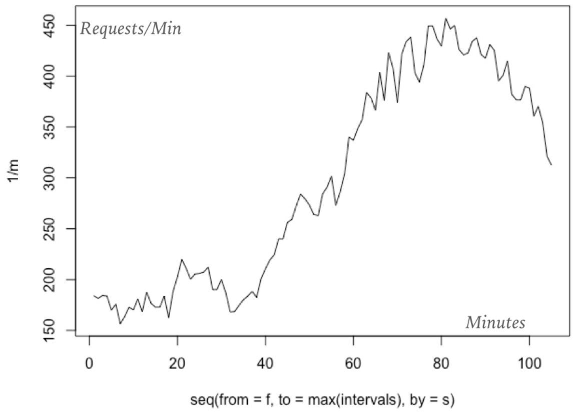
DAY-35



DAY-36

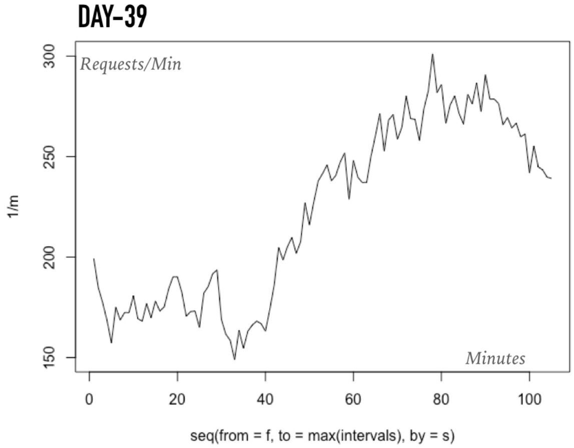


DAY-37

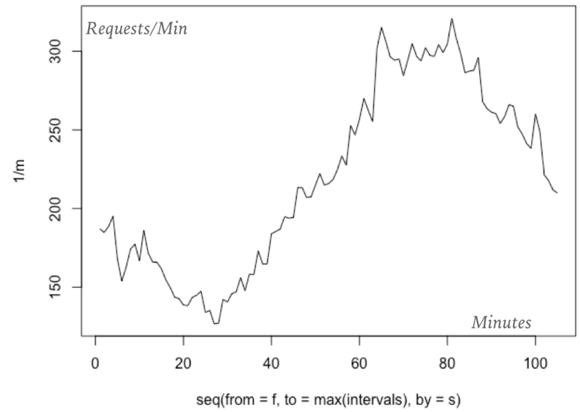


DAY-38

DAY-39

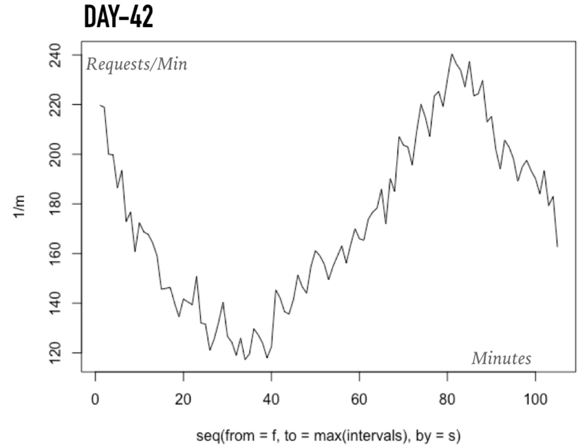


DAY-40



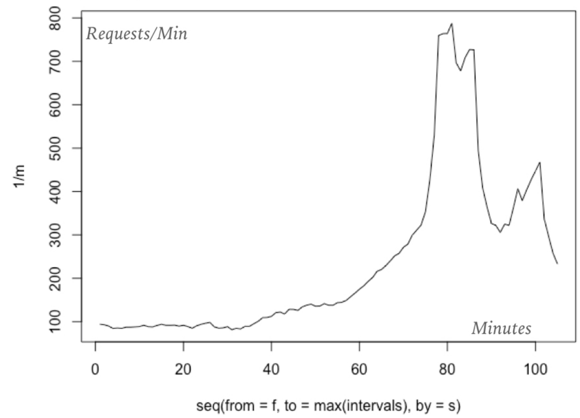
DAY-41

DAY-42

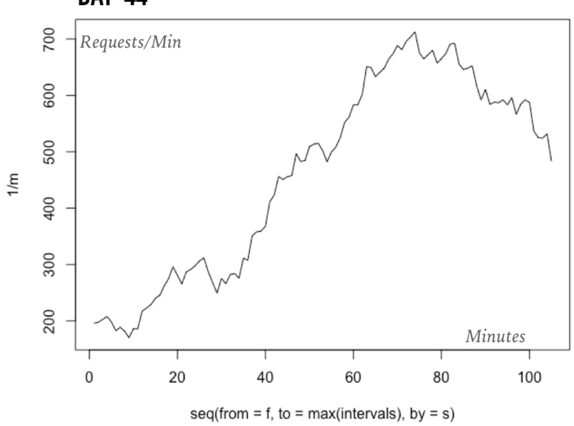


DAY-43

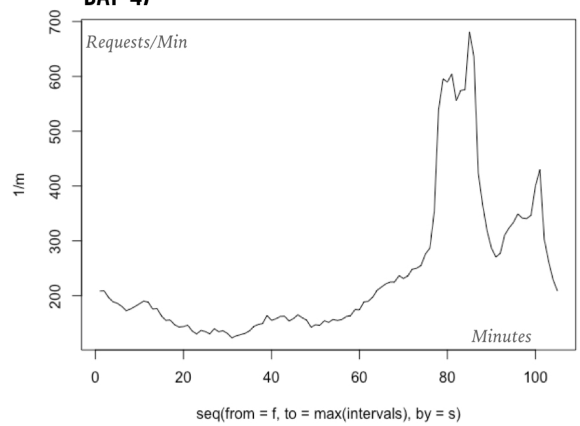
DAY-46



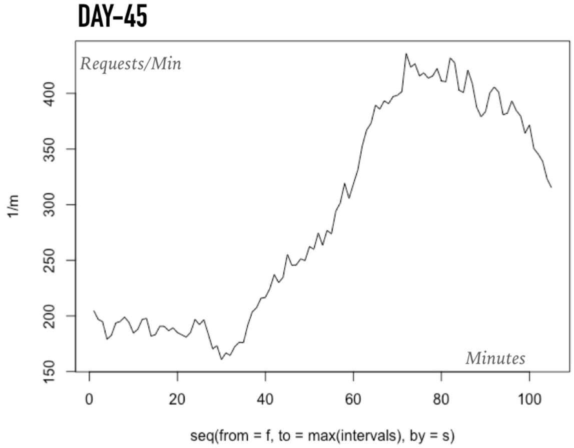
DAY-44



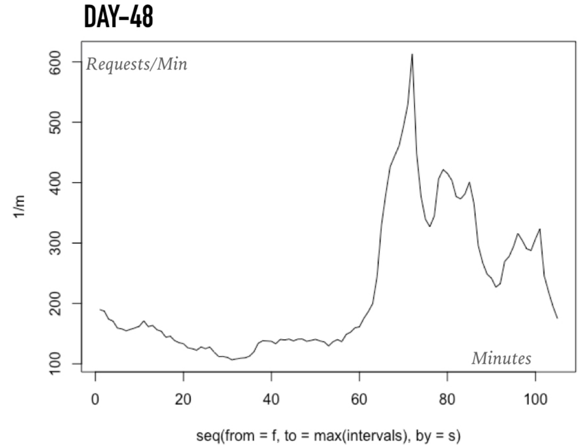
DAY-47



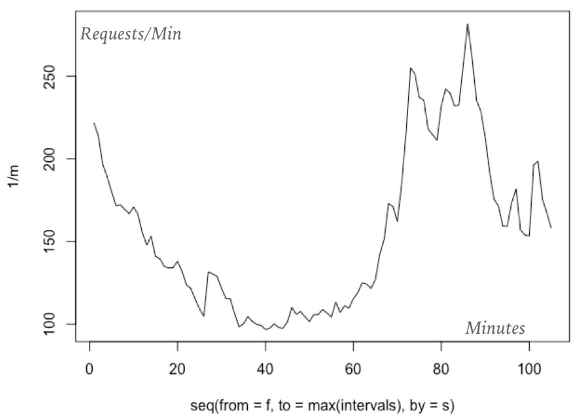
DAY-45



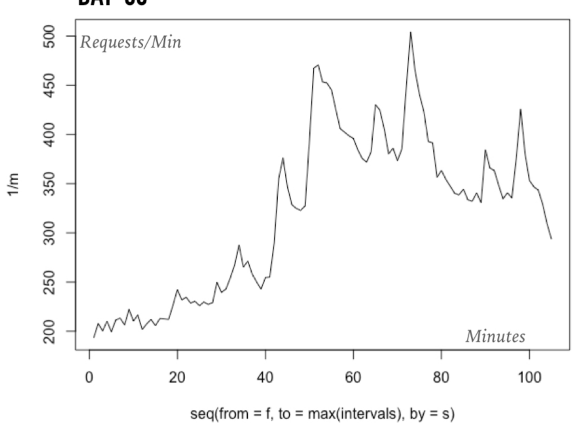
DAY-48



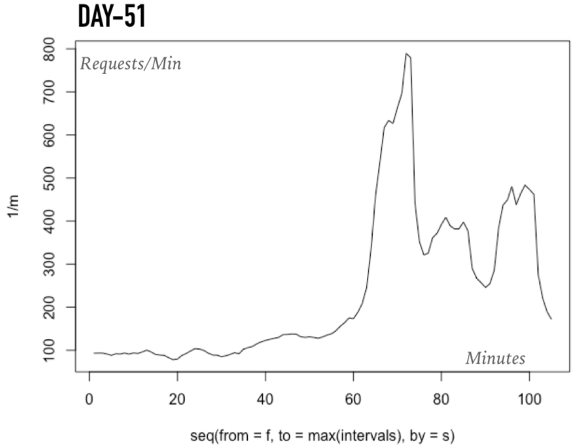
DAY-49



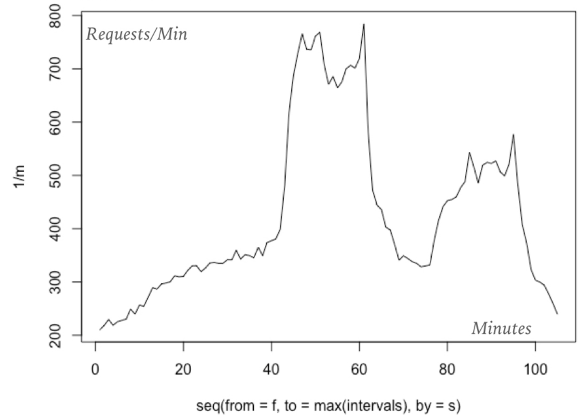
DAY-50



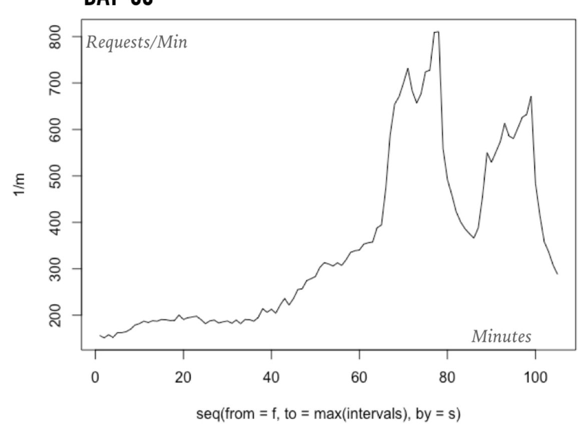
DAY-51



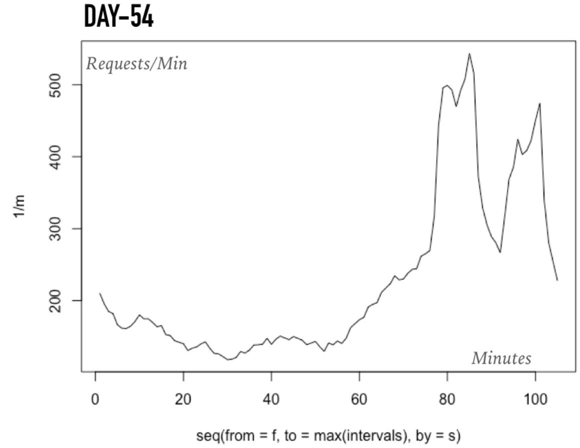
DAY-52



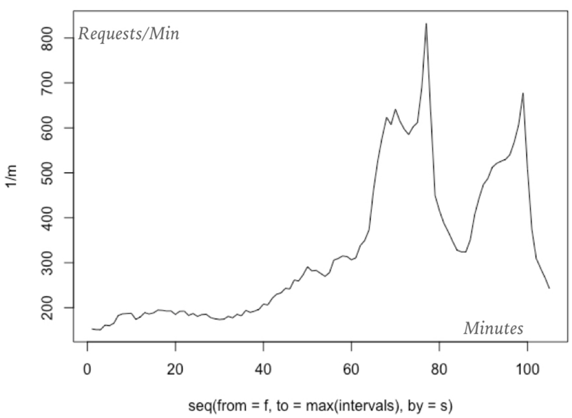
DAY-53



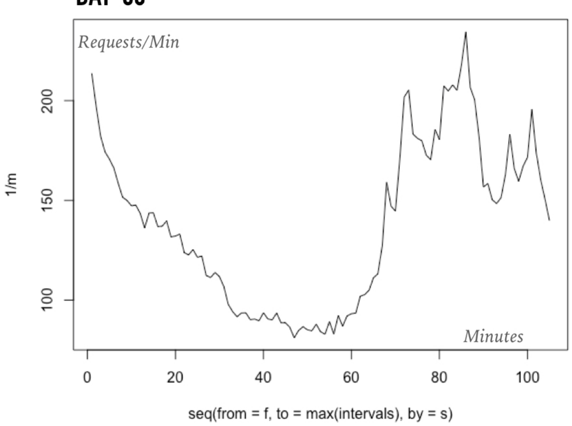
DAY-54



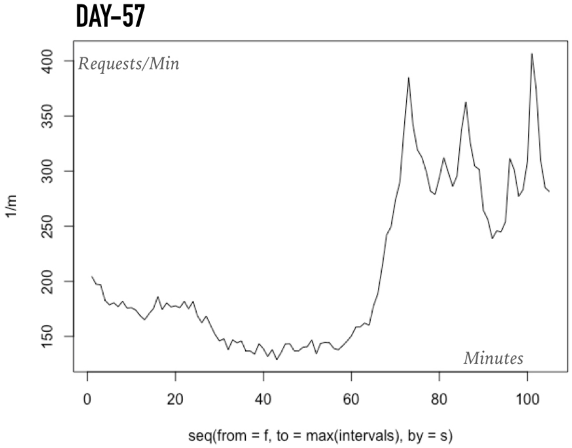
DAY-55



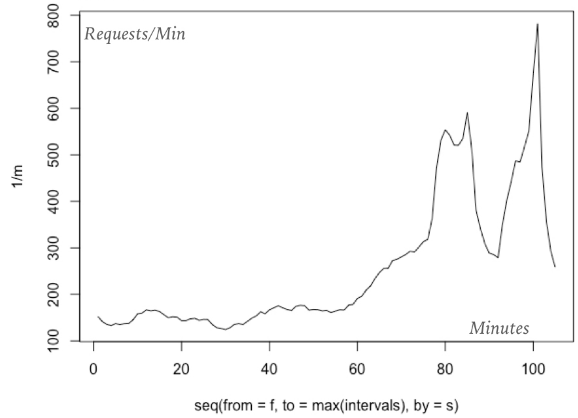
DAY-56



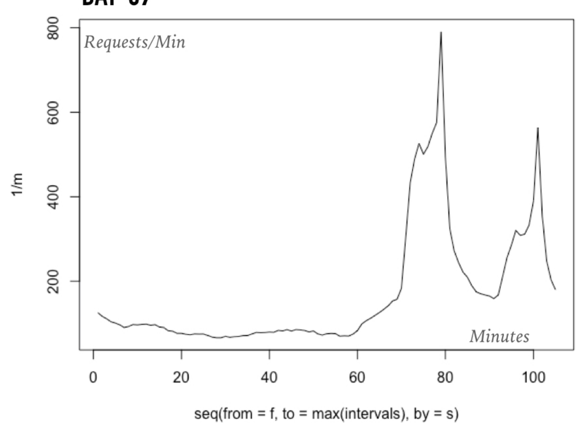
DAY-57



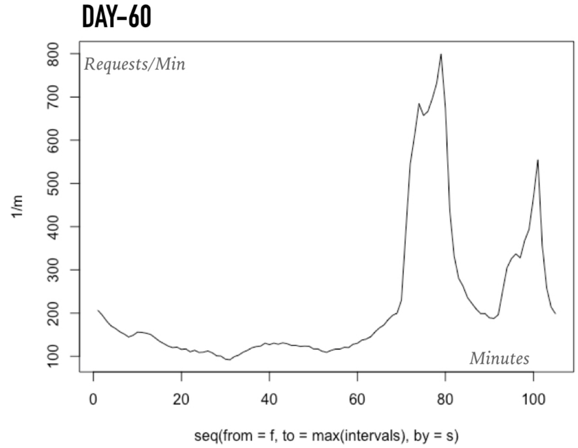
DAY-58



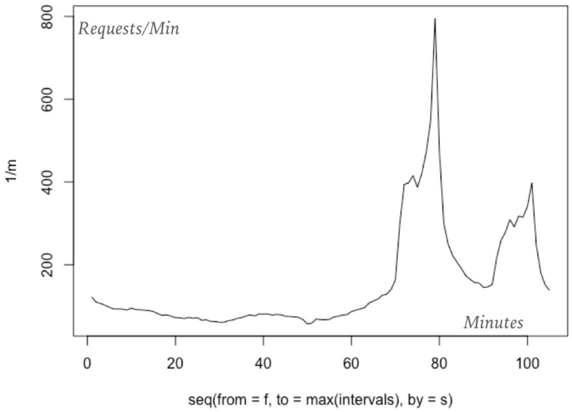
DAY-59



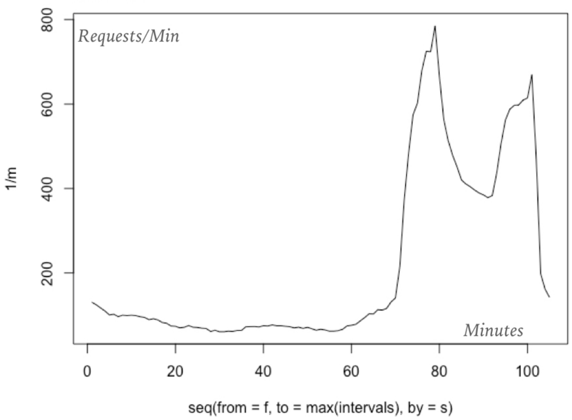
DAY-60



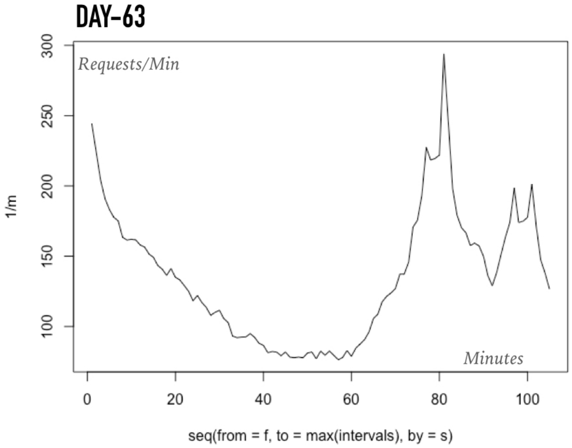
DAY-61



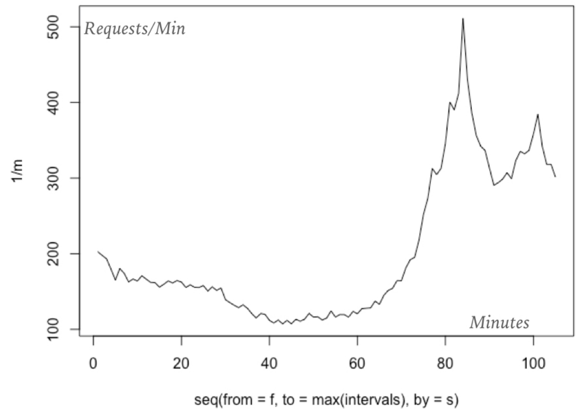
DAY-62



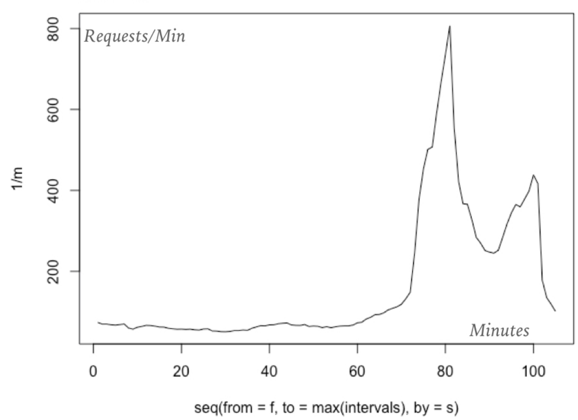
DAY-63



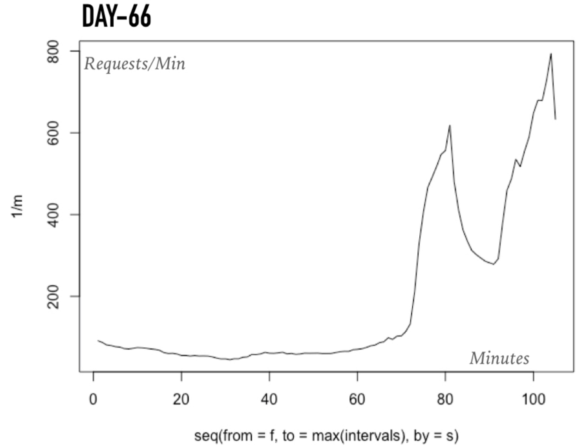
DAY-64



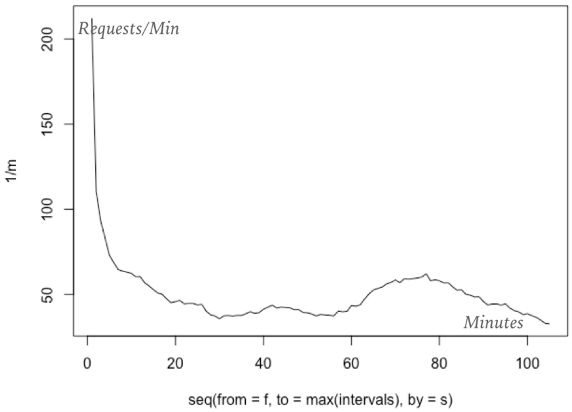
DAY-65



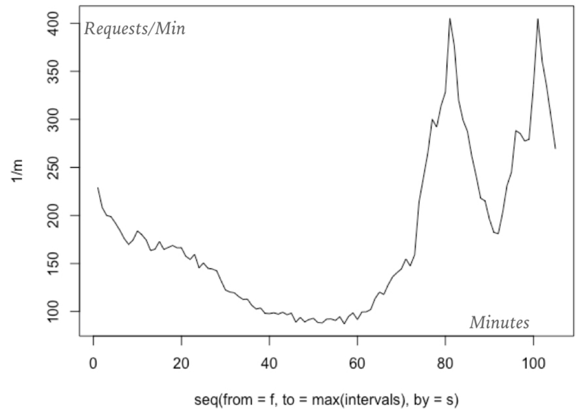
DAY-66



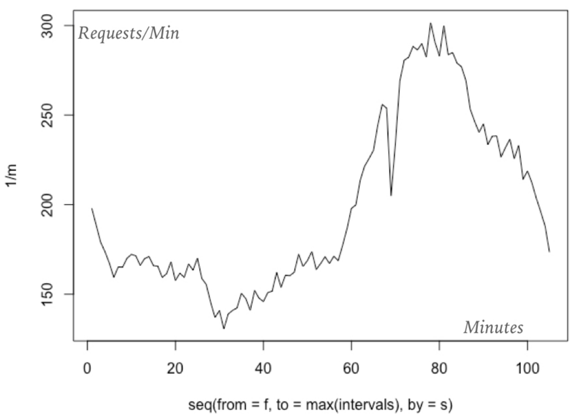
DAY-67



DAY-70

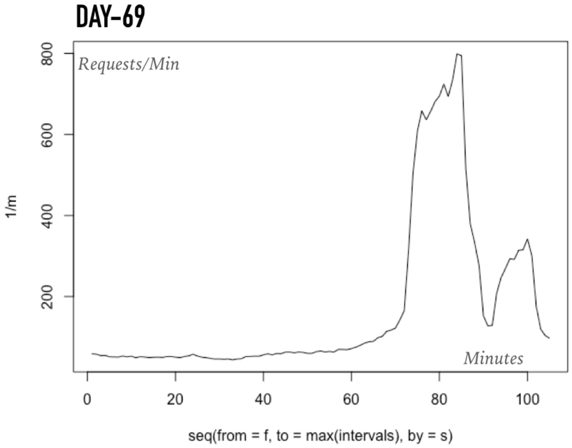


DAY-68

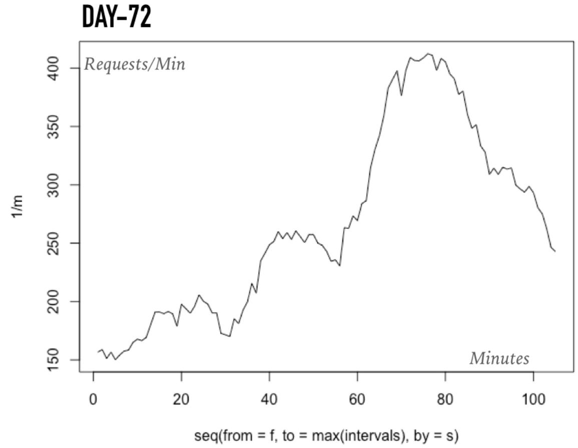


DAY-71

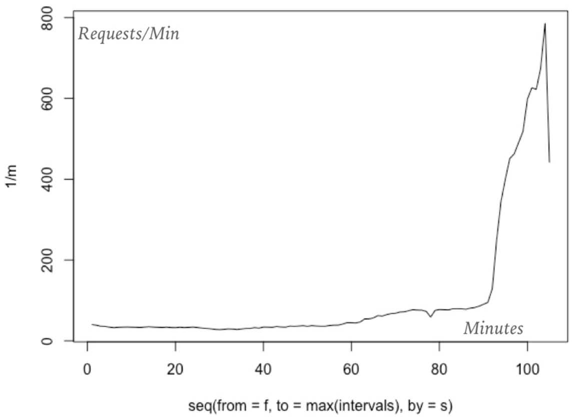
DAY-69



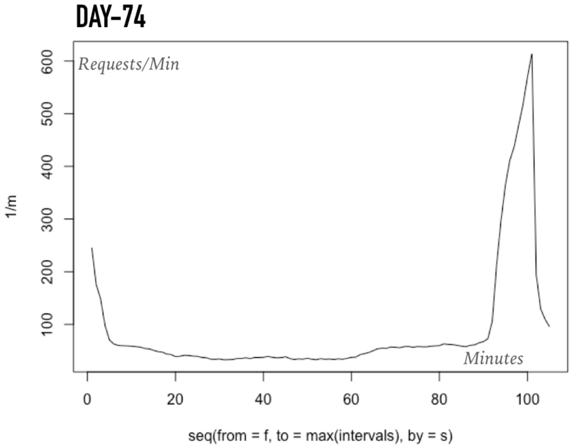
DAY-72



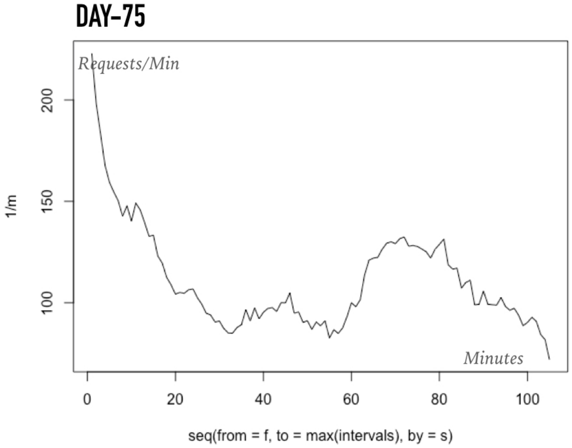
DAY-73



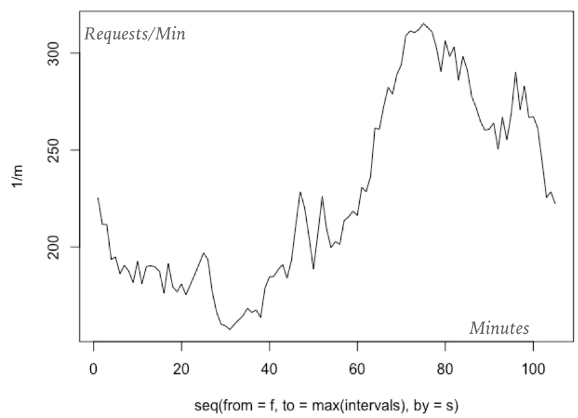
DAY-74



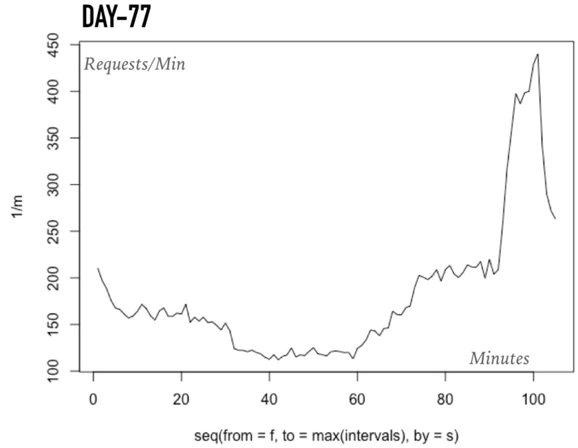
DAY-75



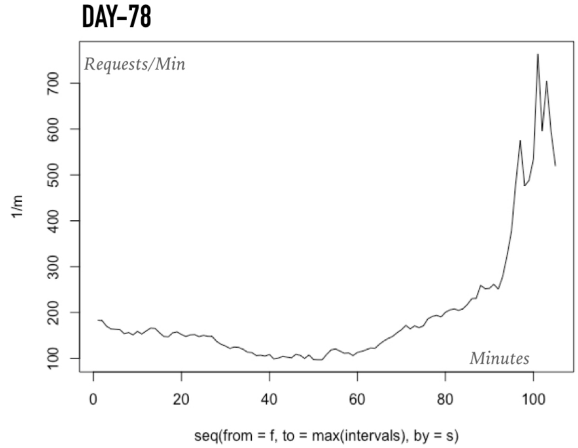
DAY-76



DAY-77

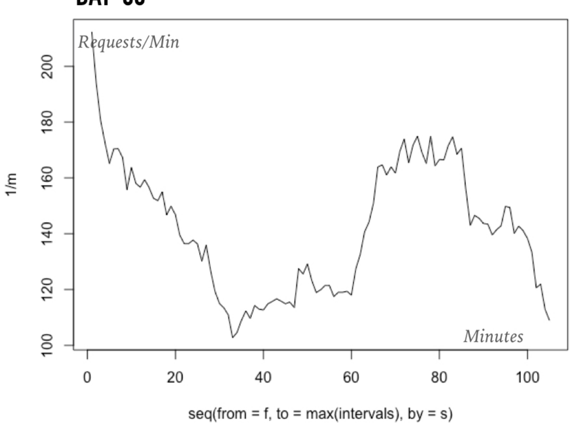


DAY-78

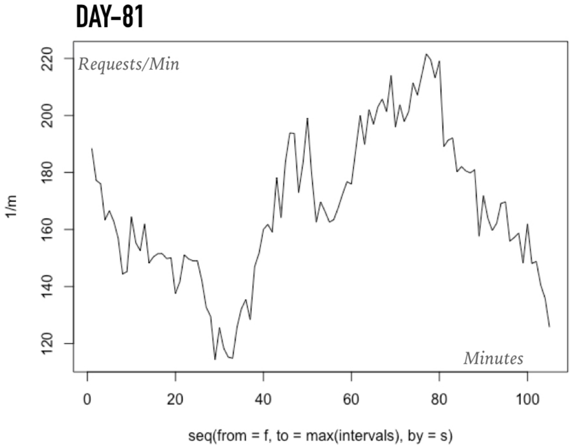


DAY-79

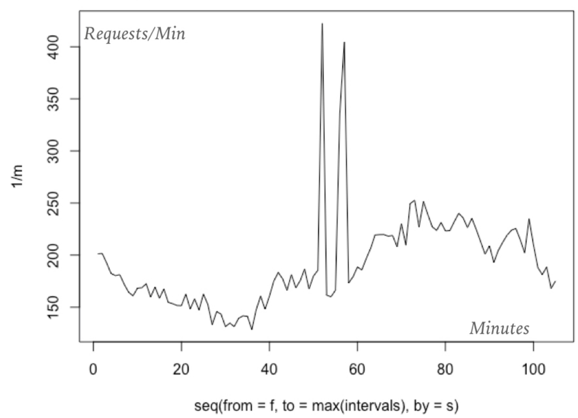
DAY-80



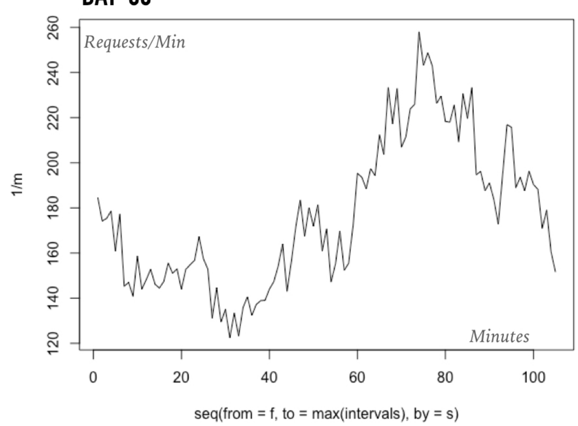
DAY-81



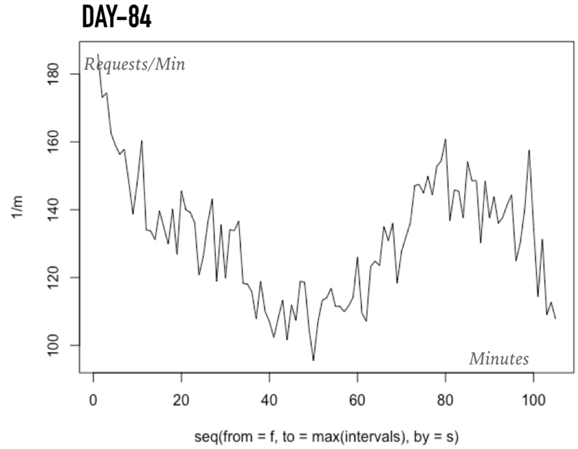
DAY-82



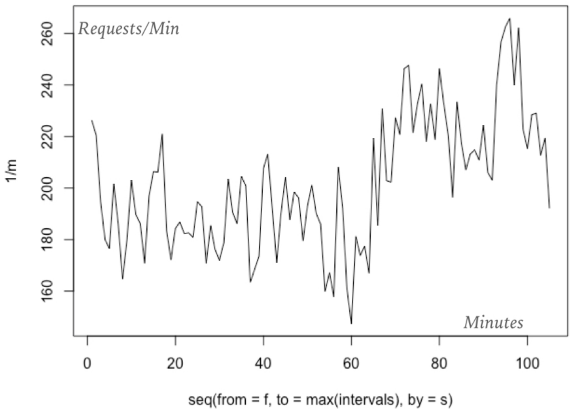
DAY-83



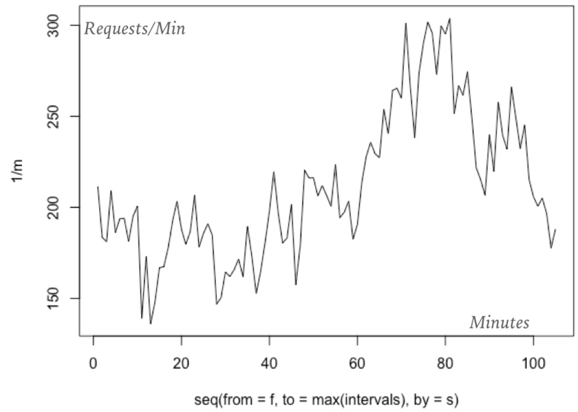
DAY-84



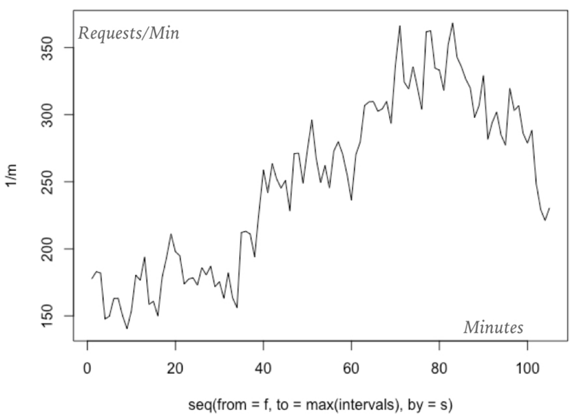
DAY-85



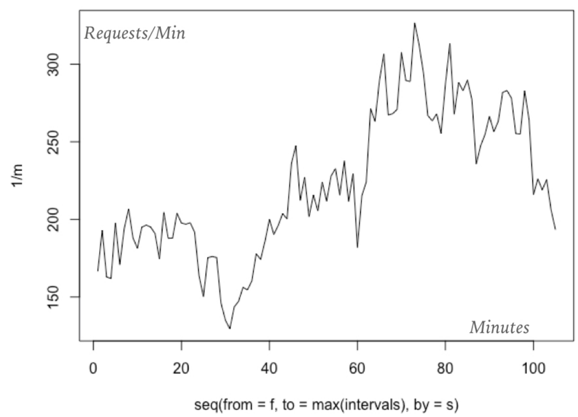
DAY-88



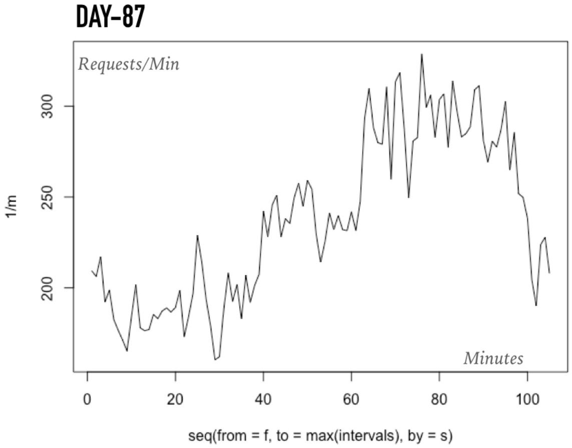
DAY-86



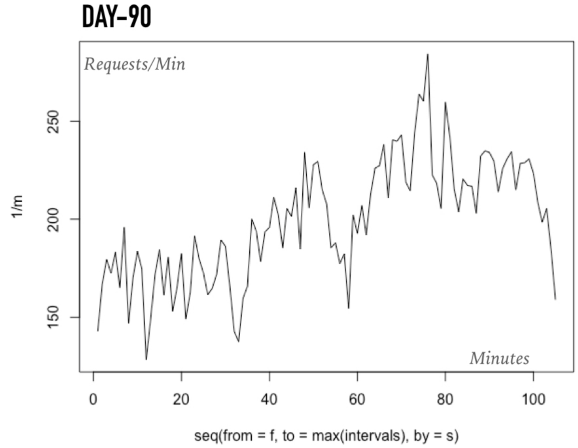
DAY-89

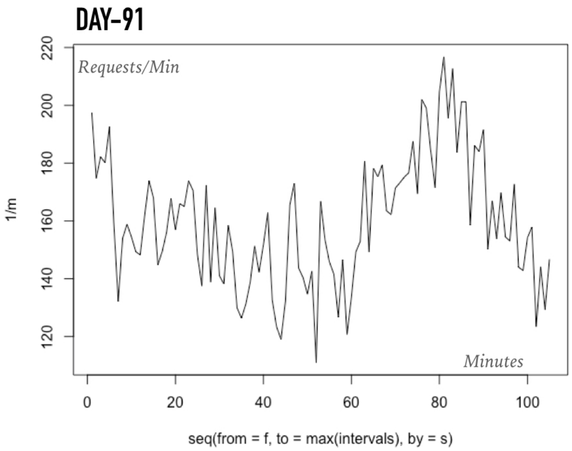


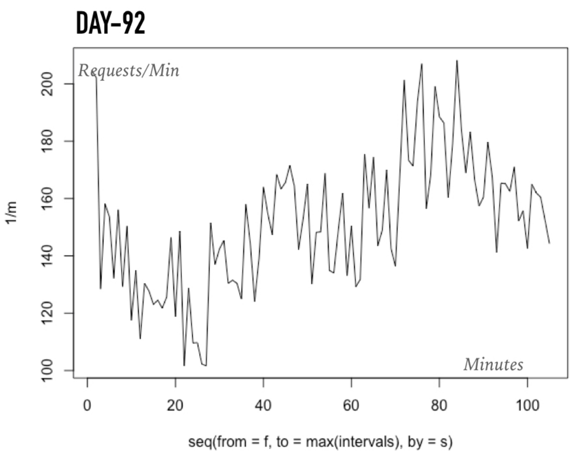
DAY-87



DAY-90







REFERENCES

[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