Probabilistic Model Checking of Peterson's Two-Process Protocol

A Matched-Horizon Waiting Comparison with Dekker

CS490 Capstone Research Notes

16 October 2026

Abstract

We model Peterson's two-process mutual-exclusion algorithm in PRISM as a Markov decision process (MDP). Two request profiles are studied: process 0 and process 1 request with probabilities 0.5/0.5 or 0.7/0.3, conditional on being scheduled outside the critical section. The scheduler is nondeterministic, not a 50/50 or 70/30 CPU lottery. We compare Peterson with the previously studied cyclic Dekker model under matched initial conditions, request profiles, horizons, and a harmonised request-to-entry reward. At $K = 20$ global transitions, maximum expected cumulative request-wait rewards are 25.221 versus 25.682 for 0.5/0.5, and 25.291 versus 25.835 for 0.7/0.3 (Dekker versus Peterson). These are abstract worst-scheduler model-step quantities, not seconds or mean delay per completed request.

1 Peterson's model

Peterson's original two-process protocol uses a flag to announce interest and a turn variable to resolve conflict [1]. The PRISM model splits P0's protocol into four local stages; P1 is symmetric:

```
[] s0=0 -> request0 : (want0'=true) & (s0'=1)
         + (1-request0) : (s0'=0);
[] s0=1 -> (turn'=1) & (s0'=2);
[] s0=2 & (!want1 | turn=0) -> (s0'=3);
[] s0=3 -> (want0'=false) & (s0'=0);
```

Table 1: Peterson's local states.

State	Meaning
$s_i = 0$	Outside; when scheduled, request or remain outside.
$s_i = 1$	Flag raised; turn assignment is pending.
$s_i = 2$	Turn assigned; test entry condition, or remain blocked.
$s_i = 3$	Inside the critical section; next action exits and clears its flag.

The model is cyclic: a process may request repeatedly. Both modules interleave asynchronously, and the next enabled module is chosen nondeterministically in the MDP [2]. Busy-wait reads without a state change are abstracted away. Both request profiles start with `turn=0`.

2 A common definition of waiting

The earlier Dekker report's `rest` reward counted only the period after a process withdrew its flag. Peterson never withdraws its flag while waiting, so we do not compare that reward directly with

Peterson's `blocked_wait`. Instead, both models now count a process from the state *after it raises its request flag until it enters the critical section.* One unit is charged for every global transition whose source state is pending:

$$r_D(s) = \mathbf{1}[s_0 = 1] + \mathbf{1}[s_1 = 1], \tag{1}$$

$$r_P(s) = \mathbf{1}[s_0 \in \{1, 2\}] + \mathbf{1}[s_1 \in \{1, 2\}], \tag{2}$$

$$W_A(K) = \max_{\sigma} \mathbb{E}_{\sigma} \left[\sum_{t=0}^{K-1} r_A(S_t) \right], \quad A \in \{D, P\}. \tag{3}$$

For Dekker, state 1 includes active trying and withdrawn yielding. For Peterson, states 1 and 2 cover flag-setting through entry testing. In both models the PRISM property is

```
R{"total_request_wait"}max=? [ C<=K ]
```

where `C<=K` accumulates rewards over K discrete global transitions [4]. Two pending processes can contribute two units during one global step. $W_A(K)$ is *not* divided by the number of entries, so it is not average wait per request.

3 Properties and the questions they ask

K means global transitions, not independent program runs. The `max` and `min` suffixes optimise over scheduler choices [3]. The same `comparison_properties.pctl` file is valid with both algorithms. Peterson additionally has `Pmax=? [ F<=K "p0_blocked" ]`: has P0 *ever* reached a guard-blocked state by K?

Table 2: Core checks. Both P0 and P1 variants are included in the property file.

Property	Question
`Pmax=? [ F "both_inside" ]`	Can any scheduler cause simultaneous critical-section entry at any reachable time?
`Pmax=? [ F<=K "p0_inside" ]`	Highest probability that P0 has entered at least once by K?
`Pmin=? [ F<=K "p0_inside" ]`	Entry probability guaranteed against the least favourable scheduler by K?
`R{"p0_request_wait"}max=? [ C<=K ]`	Maximum expected cumulative pending-request steps for P0?
`R{"total_request_wait"}max=? [ C<=K ]`	Maximum expected sum of both waiting rewards under one scheduler?
`R{"total_request_wait"}min=? [ C<=K ]`	Minimum expected sum under a favourable scheduling objective?

This is an event probability, not a duration. The P0 and P1 individual reward maxima must *not* be added to infer the total maximum: each is optimised with its own scheduler.

4 Results from PRISM model checking

The 31 points per curve were exported from PRISM 4.10.1's explicit engine after finite-horizon model checking, not path sampling. The numbers are in `comparison_graphs.dat`; `pgfplots` renders the graphs directly in this project. Every point begins at the same initial state and separately optimises over schedulers; a curve is not one run's trace.

Table 3: Maximum expected cumulative request-wait steps $W_A(K)$.

	0.5/0.5 requests			0.7/0.3 requests		
K	Dekker	Peterson	$P - D$	Dekker	Peterson	$P - D$
10	10.885	11.697	0.813	11.397	12.106	0.709
20	25.221	25.682	0.462	25.291	25.835	0.543
30	39.486	39.679	0.194	39.481	39.540	0.059

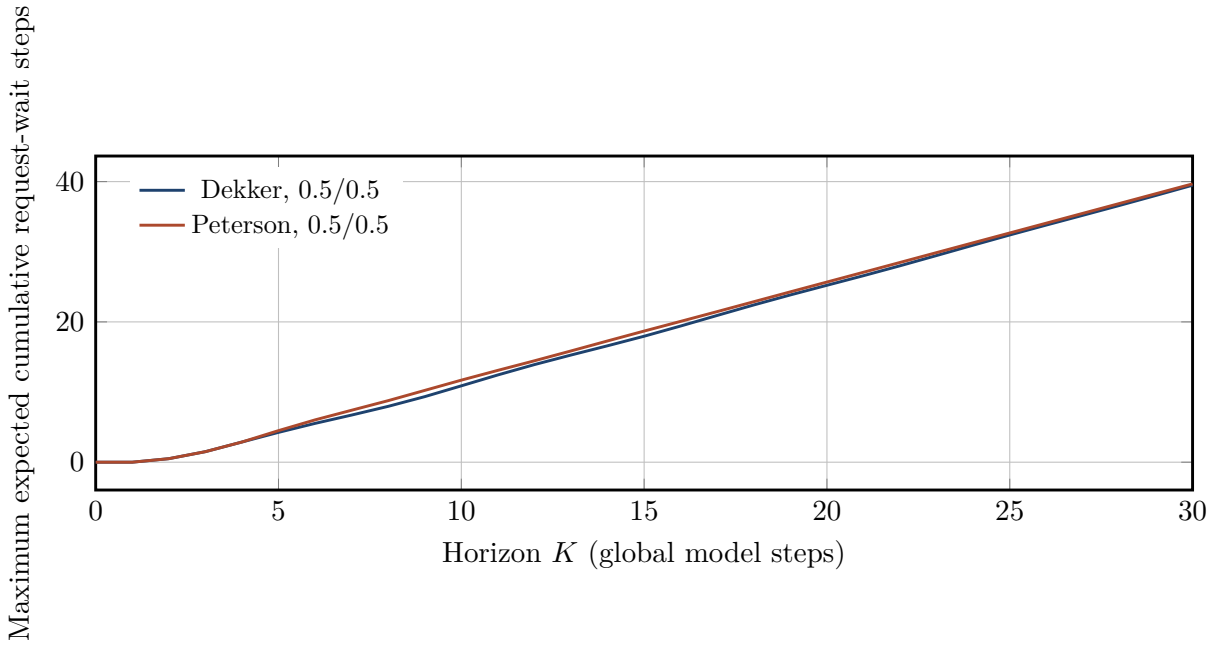


Figure 1: Matched equal-request comparison. At $K = 20$, Peterson's model has 25.682 expected cumulative request-wait steps versus Dekker's 25.221.

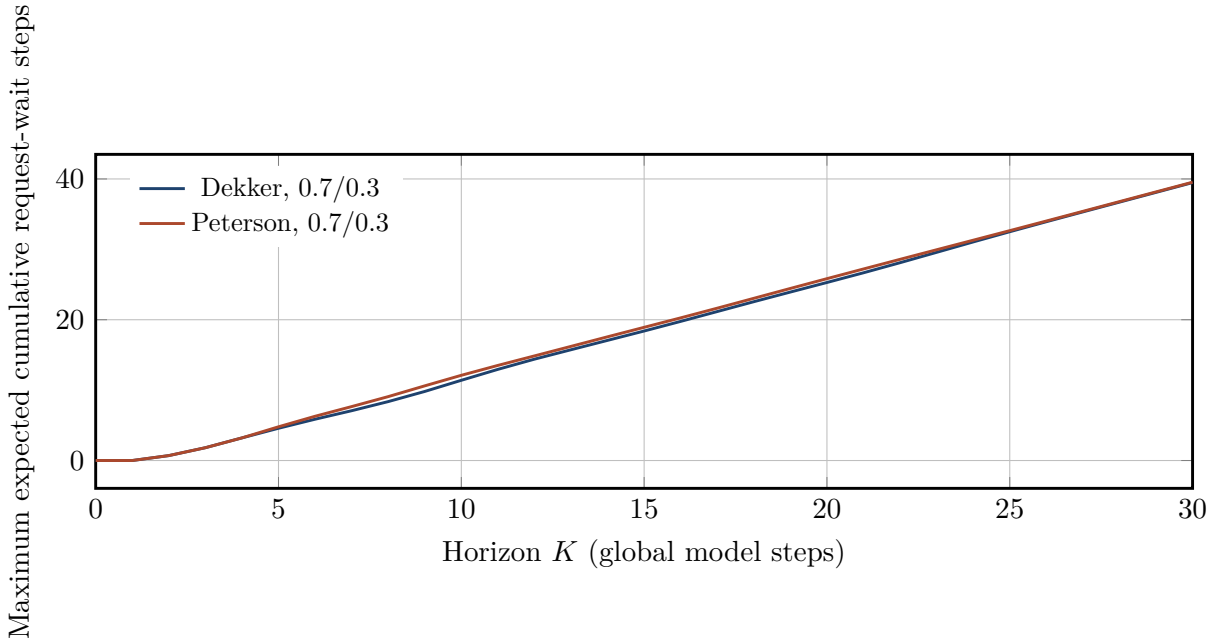


Figure 2: Matched unequal-request comparison. At $K = 20$, the values are 25.291 and 25.835, respectively.

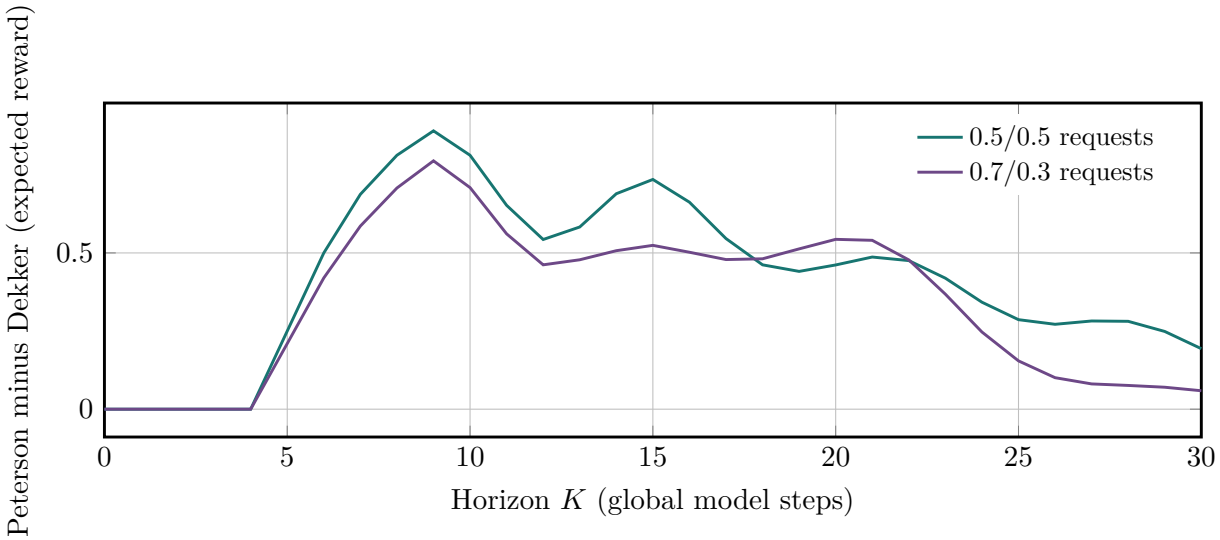


Figure 3: Differences are small but nonzero. Positive values mean Peterson has the higher worst-scheduler cumulative request-wait reward in these abstract models.

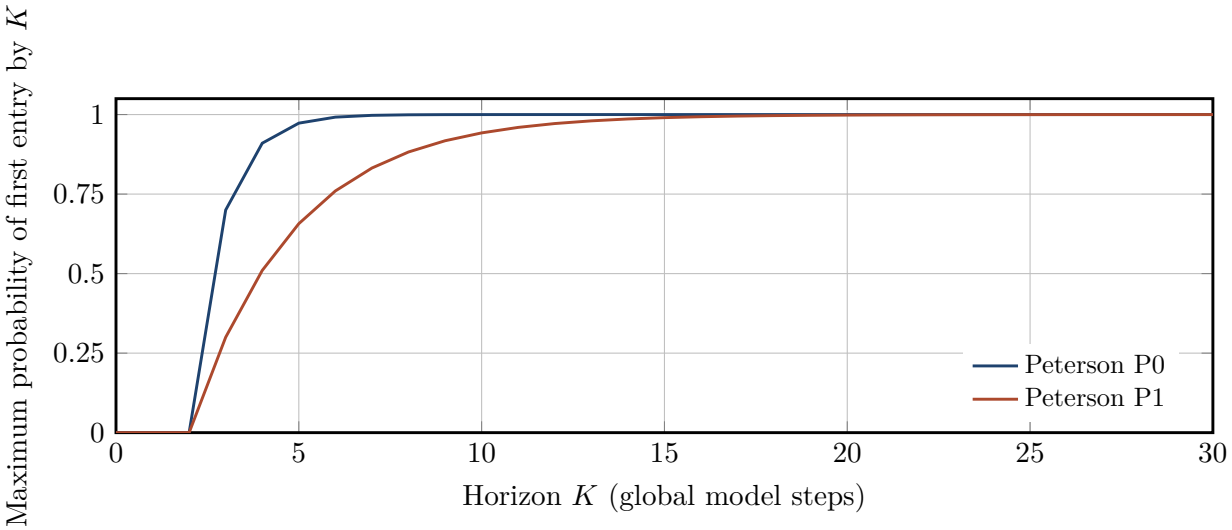


Figure 4: Peterson under 0.7/0.3 requests. At $K = 20$, best-scheduler first-entry probability is approximately 1 for P0 and 0.998372 for P1. These are not fairness guarantees.

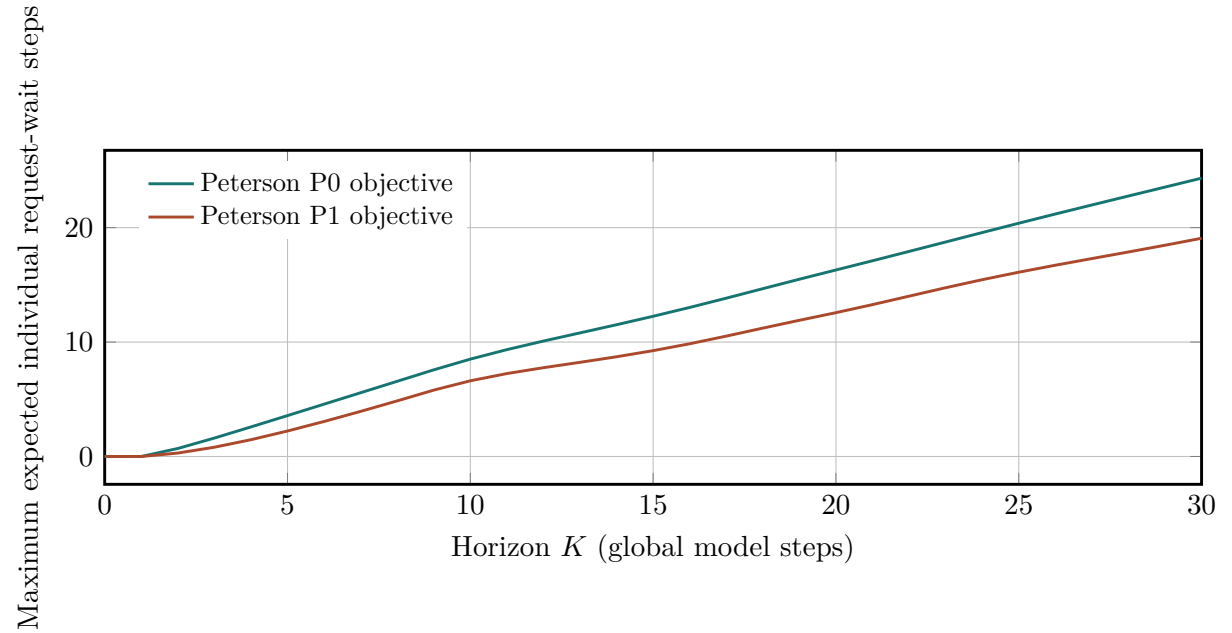


Figure 5: Individual Peterson reward maxima under 0.7/0.3 requests. Each line is separately scheduler-optimised; their sum is not Fig. 2's Peterson total.

5 Conclusions and limitations

All four algorithm/profile combinations return $P_{\max}(F\,\texttt{both_inside}) = 0$ over the full reachable finite model. At $K = 0{:}30$, $P_{\min}(F^{\leq K}\,\texttt{inside}) = 0$ for either process: an unfair scheduler need not run a selected process. This is a scheduler/liveness limitation, not a safety failure.

For the defined model-step reward, Dekker and Peterson are *not equal* at the same K once their paths diverge. Peterson's maximum is slightly higher at $K = 10, 20, 30$ in both matched request profiles. This does *not* prove that Peterson is slower on hardware. Global PRISM steps are abstract actions of unequal real cost: Peterson separates flag and turn writes; Dekker has a different entry/exit decomposition. Busy-wait reads, cache effects, memory ordering, and wall-clock timing are omitted. Also, each $R_{\max}$ uses its own worst-case scheduler, not a common realistic policy.

The earlier Dekker report's 0.7/0.3 `total_rest` result at $K = 20$ was 4.770, whereas the new `total_request_wait` result is 25.291. The new metric includes *all* pending-request states, not only flag-withdrawn yielding. To estimate an ordinary mean waiting time per completed request, a future study should fix the same explicit scheduling policy for both protocols and count completions; a maximum cumulative reward does not supply that average.

Reproducibility

The project contains `comparison_graphs.dat`, three model files under `models/`, and two property files under `properties/`. The Peterson files fix their request probabilities. The Dekker comparison model leaves `request0` and `request1` undefined; each paired run assigns the matching profile. PRISM 4.10.1's explicit engine was used for $K = 0{:}30$. For example, reproduce the unequal-profile total curves with:

```
prism models/dekker_comparison_param.pm properties/comparison_properties.pctl -prop 8 -const request0
    =0.7,request1=0.3,K=0:30 -exportresults stdout -explicit
prism models/peterson_unequal_07_03.pm properties/comparison_properties.pctl -prop 8 -const K=0:30 -
    exportresults stdout -explicit
```

Property 8 is the jointly optimised total reward. PRISM experiments can export every horizon's result [5].

References

[1] G. L. Peterson, *Myths About the Mutual Exclusion Problem*, Information Processing Letters, 12(3), 115–116, 1981. https://zoo.cs.yale.edu/classes/cs323/doc/Peterson.pdf

[2] PRISM Manual, Parallel Composition. https://www.prismmodelchecker.org/manual/ThePRISMLanguage/ParallelComposition

[3] PRISM Manual, The P Operator. https://www.prismmodelchecker.org/manual/PropertySpecification/ThePOperator

[4] PRISM Manual, Reward-based Properties. https://www.prismmodelchecker.org/manual/PropertySpecification/Reward-basedProperties

[5] PRISM Manual, Experiments. https://www.prismmodelchecker.org/manual/RunningPRISM/Experiments