# Human review packet: Naor1969QueueTolls

Generated from byte-pinned source excerpts, the expanded PaperInterface specifications, and hash-bound raw-source-to-Spec screening records.

**Paper:** Naor1969QueueTolls
**Paper id:** Naor1969QueueTolls
**Source version:** Econometrica 37(1), January 1969, pp. 15–24 (JSTOR scan)
**Source record:** audit/paper_statement_map.json
**Rows:** 6
**Generated:** 2026-09-06
**Source URL:** official source

## How to use this packet

For each source claim, compare the exact source excerpts with the expanded PaperInterface specification. Mark up this PDF or fill the blank reviewer box in the TeX source; return the annotated file for reconciliation into the paper-local human-review log.

## Open the interactive dashboard

The interactive dashboard is an optional alternative to using this PDF; it presents the same review surface rather than an additional review requirement. After forking and cloning (or pulling) AppliedModelingLib, run the following from the repository root. The cache command is needed only the first time in a new clone.

```
cd <your-repository-checkout>
lake exe cache get
python3 scripts/review_dashboard.py --paper Naor1969QueueTolls --serve
```

Open http://127.0.0.1:8765/ in a browser and keep that terminal running while reviewing. The dashboard records saved annotations in this paper's local reviewer-owned review record.

## Contents

# Material library prerequisites

### AppliedModelingLib.Probability.Queueing.BirthDeathRates

**Source locator:** cited publication, lines 82-95

## Verbatim paper-source connection

Having posed the above fundamental conditions necessary to create a framework for a queueing system with tolls we can now proceed to a detailed
↪ description of a model and a cost structure:
(i) A stationary Poisson stream of customers with parameter A arrives at a
single service station.
(ii) The station renders service in such a way that the service times are independently, identically, and exponentially distributed with intensity parameter p.
(iii) On successful completion of service, the customer is endowed with a reward
R (expressible in monetary units). All customer rewards are equal.
(iv) The cost to a customer for staying in a queue (i.e. for queueing) is C monetary units in unit time. All customer costs are equal.
(v) The newly arrived customer is required to choose one of two alternatives:
either (a) he joins the queue, incurs the losses associated with spending some of
his time in it, and finally obtains the reward; or (b) he refuses to join the queue an
action which does not bring about any gain or loss. The choice of one of these two
alternatives will be made by the customer on comparing the net gains associated
with each of them. To avoid ambiguity it will be stipulated that, in the case of a tie,
the customer will join the queue.

[Next verbatim source excerpt]

2. SOME PROPERTIES OF THE MODEL
Under the model assumptions given in the previous section, it is clear that all
reasonable strategies will be of the following nature. A newly arrived customer
will observe the queue size, i say, at that instant. This quantity is a random variable
whose distribution is partly determined by the strategy pursued. Now if the observed value of this random variable falls short of a constant n (the selected
↪ strategy),
the newly arrived customer will join the queue; if the observed value i is equal to n
the new customer is diverted and does not join the queue. The observed value i
can never exceed n in this model.
Clearly we are confronted with a system that is identical with a queueing model
in which finite waiting space only is available to queueing customers. If we define
(2)

it'

P

we obtain the following steady state equations:
(3)

pip) = pi.

(0n<i< n),

[form-feed in source extraction]
18

P. NAOR

the solution3 of which is

(4)

Pi

(Oi

IP

p

1 +p + +pp

n).

## Lean-expanded library semantic target

type:
Type

constructor AppliedModelingLib.Probability.Queueing.BirthDeathRates.mk:
(ℕ → ℝ) → (death : ℕ → ℝ) → death 0 = 0 → AppliedModelingLib.Probability.Queueing.BirthDeathRates

## Exact Lean library declaration

```
/-- Birth and death rates for a queue-length process on `ℕ`. -/
structure BirthDeathRates where
  birth : ℕ → ℝ
  death : ℕ → ℝ
  death_zero : death 0 = 0
```

**Recorded source-to-library screening** **Verdict:** matches

**Reason:** The source Poisson arrivals and exponential services induce queue-length birth and death rates, and death_zero states the impossibility of a service departure from an empty queue. This is a generic carrier: the constant rates, positivity, and capacity cutoff are supplied by the concrete source-model instantiation rather than omitted from a claimed source theorem.

**Reviewer check:** ☐ Matches source input

*If left unchecked, explain the discrepancy or uncertainty below.*

**Reviewer annotation**

**AppliedModelingLib.Probability.Queueing.GeneratorStationaryLaw**

**Source locator:** cited publication, lines 82-95

## Verbatim paper-source connection

Having posed the above fundamental conditions necessary to create a framework for a queueing system with tolls we can now proceed to a detailed
↪ description of a model and a cost structure:
(i) A stationary Poisson stream of customers with parameter A arrives at a
single service station.
(ii) The station renders service in such a way that the service times are independently, identically, and exponentially distributed with intensity parameter p.
(iii) On successful completion of service, the customer is endowed with a reward
R (expressible in monetary units). All customer rewards are equal.
(iv) The cost to a customer for staying in a queue (i.e. for queueing) is C monetary units in unit time. All customer costs are equal.
(v) The newly arrived customer is required to choose one of two alternatives:
either (a) he joins the queue, incurs the losses associated with spending some of
his time in it, and finally obtains the reward; or (b) he refuses to join the queue an
action which does not bring about any gain or loss. The choice of one of these two
alternatives will be made by the customer on comparing the net gains associated
with each of them. To avoid ambiguity it will be stipulated that, in the case of a tie,
the customer will join the queue.

[Next verbatim source excerpt]

2. SOME PROPERTIES OF THE MODEL
Under the model assumptions given in the previous section, it is clear that all
reasonable strategies will be of the following nature. A newly arrived customer
will observe the queue size, i say, at that instant. This quantity is a random variable
whose distribution is partly determined by the strategy pursued. Now if the observed value of this random variable falls short of a constant n (the selected
↪ strategy),
the newly arrived customer will join the queue; if the observed value i is equal to n
the new customer is diverted and does not join the queue. The observed value i
can never exceed n in this model.
Clearly we are confronted with a system that is identical with a queueing model
in which finite waiting space only is available to queueing customers. If we define
(2)

it'

P

we obtain the following steady state equations:
(3)

pip) = pi.

(0n<i< n),

[form-feed in source extraction]
18

P. NAOR

the solution3 of which is

(4)

Pi

(Oi

IP

p

1 +p + +pp

n).

## Lean-expanded library semantic target

type:
AppliedModelingLib.Probability.Queueing.BirthDeathRates → Type

constructor AppliedModelingLib.Probability.Queueing.GeneratorStationaryLaw.mk:
{G : AppliedModelingLib.Probability.Queueing.BirthDeathRates} →
  (mass : ℕ → ℝ) →
    (∀ (n : ℕ), 0 ≤ mass n) →
      HasSum mass 1 →
        (∀ (n : ℕ),
            mass n * (G.1 n + G.2 n) = (if n = 0 then 0 else mass (n - 1) * G.1 (n - 1)) + mass (n + 1) * G.2 (n + 1)) →
          AppliedModelingLib.Probability.Queueing.GeneratorStationaryLaw G

## Exact Lean library declaration

```
/-- A normalized nonnegative solution of the birth--death generator equations. -/
structure GeneratorStationaryLaw (G : BirthDeathRates) where
  mass : ℕ → ℝ
  nonneg : ∀ n, 0 ≤ mass n
  hasSum_one : HasSum mass 1
  generator_balance : StationaryGeneratorBalance G mass
```

**Recorded source-to-library screening** **Verdict:** matches

**Reason:** The declaration requires a nonnegative normalized mass satisfying the birth-death generator balance at every state, the generic form of the source finite-capacity steady-state equations. Finite capacity, constant rates, and zero mass beyond the threshold belong to the instantiated rates and mass; HasSum is compatible with that finite support.

**Reviewer check:** ☐ Matches source input

*If left unchecked, explain the discrepancy or uncertainty below.*

**Reviewer annotation**

# Source claims

## Review row 1

**Source locator:** cited publication, lines 82-95

### Verbatim source input

```
Having posed the above fundamental conditions necessary to create a framework for a queueing system with tolls we can now proceed to a detailed description of a model and a cost structure:
(i) A stationary Poisson stream of customers with parameter A arrives at a
single service station.
(ii) The station renders service in such a way that the service times are independently, identically, and exponentially distributed with intensity parameter p.
(iii) On successful completion of service, the customer is endowed with a reward
R (expressible in monetary units). All customer rewards are equal.
(iv) The cost to a customer for staying in a queue (i.e. for queueing) is C monetary units in unit time. All customer costs are equal.
(v) The newly arrived customer is required to choose one of two alternatives:
either (a) he joins the queue, incurs the losses associated with spending some of
his time in it, and finally obtains the reward; or (b) he refuses to join the queue an
action which does not bring about any gain or loss. The choice of one of these two
alternatives will be made by the customer on comparing the net gains associated
with each of them. To avoid ambiguity it will be stipulated that, in the case of a tie,
the customer will join the queue.

[Next verbatim source excerpt]

2. SOME PROPERTIES OF THE MODEL
Under the model assumptions given in the previous section, it is clear that all
reasonable strategies will be of the following nature. A newly arrived customer
will observe the queue size, i say, at that instant. This quantity is a random variable
whose distribution is partly determined by the strategy pursued. Now if the observed value of this random variable falls short of a constant n (the selected strategy),
the newly arrived customer will join the queue; if the observed value i is equal to n
the new customer is diverted and does not join the queue. The observed value i
can never exceed n in this model.
Clearly we are confronted with a system that is identical with a queueing model
in which finite waiting space only is available to queueing customers. If we define
(2)

it'

P

we obtain the following steady state equations:
(3)

pip) = pi.

(0n<i< n),


[form-feed in source extraction]
18

P. NAOR

the solution3 of which is

(4)

Pi

(Oi

IP

p

1 +p + +pp

n).
```

### Expanded PaperInterface specification

```
∀ (rho serviceRate : ℝ) (capacity : ℕ),
  0 < rho →
    0 < serviceRate →
      ∃ (law :
        AppliedModelingLib.Probability.Queueing.GeneratorStationaryLaw
          { birth := fun (n : ℕ) => if n < capacity then rho * serviceRate else 0,
            death := fun (n : ℕ) => if n = 0 then 0 else serviceRate, death_zero := ⋯ }),
        law.1 = fun (state : ℕ) =>
          if state ≤ capacity then rho ^ state / ∑ i ∈ Finset.range (capacity + 1), rho ^ i else 0
```

### Lean proof endpoint (built separately)

```
Naor1969QueueTolls.finiteCapacityStationaryLaw
```

**Recorded source-to-Spec screening** **Verdict:** matches

**Reason:** The expanded generator has arrivals at rho times mu below capacity, service at mu above zero, and mass rho^i divided by the finite normalizer on i at most capacity. The positive-rate premises are explicit, and the finite sum yields the intended uniform finite-capacity law at rho = 1.

**Reviewer check:** ☐ Matches source input

*If left unchecked, explain the discrepancy or uncertainty below.*

**Reviewer annotation**

| |
|---|
| |

# Review row 2

**Source locator:** cited publication, lines 175-281

## Verbatim source input

The generating function is derived as

(5)

(5)

n

-p P
~~I

g(z) =

pIz

p 1

I

-(pZ)n+
1 -pz

Pn+

'

The expected value, q, of the random variable equals
p[1-(n+

1) pn +npn+ 1]

P

(n +1) pn

I -P
pn+)
1pn+p
(1-p)(I-_
q=E{i} =
The expected number of customers, C say, diverted from the service station in
unit time is given by

(6)

(7)

-

= iPn

pI(n+1

We mention, in passing, that the busy fraction b, i.e., the degree of utilization of
the service station, is, of course, not equal to p (as in the "usual"models) but rather
nP

(8)

b=

Pi

-Po

_ p(n~f)
-

P

i=I

n1

The expected number of customers joining the queue in unit time equals
(9)(9)

,1-4

= pn~X(-~=1
(-Pn)

=ip

p___

n+1t)

Pn__

n+1t

The expected number of customers leaving the service station in unit time equals
(10)

jbb=u(l-po)=u

-lij

;Pt;]1

These two quantities must be identical under steady state conditions and, indeed,
it is easy to verify that
(11)

p = 1- Po
l-pn

## Expanded PaperInterface specification

```
∀ (rho serviceRate z : ℝ) (capacity : ℕ),
  0 < rho →
    0 < serviceRate →
      (∑ state ∈ Finset.range (capacity + 1), ↑state * rho ^ state) /
            ∑ state ∈ Finset.range (capacity + 1), rho ^ state =
          (∑ n ∈ Finset.range (capacity + 1), ↑n * rho ^ n) / ∑ i ∈ Finset.range (capacity + 1), rho ^ i ∧
        ∑ state ∈ Finset.range (capacity + 1),
              (if state ≤ capacity then rho ^ state / ∑ i ∈ Finset.range (capacity + 1), rho ^ i else 0) * z ^ state =
            (∑ i ∈ Finset.range (capacity + 1), (rho * z) ^ i) / ∑ i ∈ Finset.range (capacity + 1), rho ^ i ∧
          (rho * serviceRate *
                if capacity ≤ capacity then rho ^ capacity / ∑ i ∈ Finset.range (capacity + 1), rho ^ i else 0) +
              rho * serviceRate *
                (1 - if capacity ≤ capacity then rho ^ capacity / ∑ i ∈ Finset.range (capacity + 1), rho ^ i else 0) =
            rho * serviceRate ∧
            rho * serviceRate *
                (1 - if capacity ≤ capacity then rho ^ capacity / ∑ i ∈ Finset.range (capacity + 1), rho ^ i else 0) =
              serviceRate * (1 - if 0 ≤ capacity then rho ^ 0 / ∑ i ∈ Finset.range (capacity + 1), rho ^ i else 0) ∧
            (0 < capacity →
              (1 - if 0 ≤ capacity then rho ^ 0 / ∑ i ∈ Finset.range (capacity + 1), rho ^ i else 0) /
                  (1 -
                    if capacity ≤ capacity then rho ^ capacity / ∑ i ∈ Finset.range (capacity + 1), rho ^ i else 0) =
                rho)
```

## Lean proof endpoint (built separately)

Naor1969QueueTolls.stationaryPerformance

## Recorded source-to-Spec screening **Verdict:** matches

**Reason:** The target contains the finite-sum mean and generating function, rejected and admitted flows, admitted flow equal to service-completion flow, and the source ratio identity. Positive rho and mu and a positive-capacity guard for the final denominator are explicit; the finite-sum forms supply the intended values at rho = 1.

**Reviewer check:** ☐ Matches source input

*If left unchecked, explain the discrepancy or uncertainty below.*

**Reviewer annotation**

# Review row 3

**Source locator:** cited publication, lines 120-131

## Verbatim source input

Secondly, for our model to make sense it is required that under favorable circumstances a customer will desire to queue up for (completion of) service.
↪ Hence if a
newly arrived customer encounters a completely empty service station he should
make the "pro-queueing" choice. His expected loss is given by CV' whereas his
reward to be collected at the end of service equals R. If the dimensionless quantity
(Rp/C) is denoted by v, it is clear that in meaningful models the following inequality
must hold
(1)

VS= >C ) 1.

If inequality (1) does not hold the optimal policy is to disband the service station
and divert the customer stream altogether.

[Next verbatim source excerpt]

The newly arrived customer weighs the two alternatives-to join or not to join
the queue-by the net gains associated with them. The net gain, in the first case,
is equal to
(t2)

Gi = R- (i+ 1)C-.

In the alternative case the net gain is zero. Hence self interest is served if a strategy
is established in the following fashion. An integer, ns,is found which satisfies simultaneously two inequalities
(13)

R-n

C-

0

and
(14)

R- (ns+ l)C-

< O.

Inequality (13) pertains to the case where the number of queueing customers
(including the one in service) encountered by the newly arrived customer falls
short of the critical number by one. The customer, of course, is supposed to join
and indeed the inequality is in his favor. Inequality (14) relates to the unfavorable
event: the critical number, ns, of customers is already in the queue. We can incorporate the two inequalities in one expression
(15)

nS <C

1

Alternatively we may express the same idea in different notation
(16)

nS-I[Vs]

where [ ] is the well known bracket function; that is, ns is the largest integer not
exceeding vs.
We note that the critical number, ns, derived by "actualizing self interest" depends on ,u,R, and C, but not on the arrival intensity A.This fact alone
↪ sufficesbefore pursuing further detailed investigation-to throw serious doubt on the

## Expanded PaperInterface specification

```
∀ (reward queueCost serviceRate : ℝ),
  0 < queueCost →
    0 < serviceRate →
      1 < reward * serviceRate / queueCost →
        (∀ queueLength < ⌊reward * serviceRate / queueCost⌋₊,
            0 ≤ reward - ↑(queueLength + 1) * queueCost / serviceRate) ∧
          ∀ (queueLength : ℕ),
            ⌊reward * serviceRate / queueCost⌋₊ ≤ queueLength →
              reward - ↑(queueLength + 1) * queueCost / serviceRate < 0
```

## Lean proof endpoint (built separately)

Naor1969QueueTolls.selfOptimizingThreshold

## Recorded source-to-Spec screening **Verdict:** matches

**Reason:** The target expands v_s to R mu divided by C and the joining gain to R minus (i + 1) C divided by mu; it joins below floor v_s with weak gain and rejects at or above it with strict loss. The source tie-to-join convention and the positive C, mu, and meaningful-model premises are present.
**Reviewer check:** ☐ Matches source input

*If left unchecked, explain the discrepancy or uncertainty below.*
**Reviewer annotation**

# Review row 4

**Source locator:** cited publication, lines 120-131

## Verbatim source input

Secondly, for our model to make sense it is required that under favorable circumstances a customer will desire to queue up for (completion of) service.
↪ Hence if a
newly arrived customer encounters a completely empty service station he should
make the "pro-queueing" choice. His expected loss is given by CV' whereas his
reward to be collected at the end of service equals R. If the dimensionless quantity
(Rp/C) is denoted by v, it is clear that in meaningful models the following inequality
must hold
(1)

VS= >C ) 1.

If inequality (1) does not hold the optimal policy is to disband the service station
and divert the customer stream altogether.

[Next verbatim source excerpt]

If the viewpoint is taken that the expected sum of the net gains accruing to
customers in unit time is the public good which should be optimized, we must
proceed in a different mode from that outlined in the previous section. We note
that expected total net gain, P, under some strategy n is given by
(17)

P =(-C)R=-RCE {i} =

(l -Pn)-Cq

p
-AR l
pna cL1P

n pn+l

[form-feed in source extraction]
20

P. NAOR

By some elementary (but lengthy) considerations it can be shown that the
function P in its dependence on n is "discretely unimodal" or, in other words, a
local maximum is a global maximum. Hence we seek that strategy, no say, which is
associated with two inequalities

[Next verbatim source excerpt]

To deal with (22) it will be convenient to investigate a function,

(23)

Vs= [vo( (-p)-p

(' -P vo)] (1 _p)-

2

of two independent variables p (> 0) and vo() 1). We note, in passing, that no true
singularity exists for this function if p = 1; rather the function is well behaved and
a nonzero and finite function value exists at that value of p, to wit: vs= (vo(vo+ 1)/2).
Next we study a setting in which the value of p is arbitrary (positive) but fixed; we
note that vs is a boundlessly increasing function of vo. Hence the integers between
which vo lies (viewed now as a function of v, and p) will obey the inequalities associated with (22). As a result we arrive at
(24)

no = [vo],

an expression which is completely analogous to (18). Further manipulations lead
to the inequality
(25)

VO< Vs

where the equality sign holds only4 if vs equals unity.

## Expanded PaperInterface specification

```
∀ (rho reward queueCost serviceRate : ℝ),
  0 < rho →
    0 < queueCost →
      0 < serviceRate →
        1 < reward * serviceRate / queueCost →
          ∃ (socialCapacity : ℕ),
            (∀ (capacity : ℕ),
                reward * (rho * serviceRate) *
                      (1 -
                        if capacity ≤ capacity then rho ^ capacity / ∑ i ∈ Finset.range (capacity + 1), rho ^ i
                        else 0) -
                    queueCost *
```

```
        ((∑ state ∈ Finset.range (capacity + 1), ↑state * rho ^ state) /
          ∑ state ∈ Finset.range (capacity + 1), rho ^ state) ≤
    reward * (rho * serviceRate) *
        (1 -
          if socialCapacity ≤ socialCapacity then
            rho ^ socialCapacity / ∑ i ∈ Finset.range (socialCapacity + 1), rho ^ i
          else 0) -
      queueCost *
        ((∑ state ∈ Finset.range (socialCapacity + 1), ↑state * rho ^ state) /
          ∑ state ∈ Finset.range (socialCapacity + 1), rho ^ state)) ∧
  socialCapacity ≤ ⌊reward * serviceRate / queueCost⌋₊
```

## Lean proof endpoint (built separately)

Naor1969QueueTolls.socialOptimalThreshold

## Recorded source-to-Spec screening **Verdict:** matches

**Reason:** The target states existence of a global maximizer of the source aggregate reward-minus-queue-cost objective and selects it no above floor v_s, the source consequence of discrete unimodality and v_o less than v_s. The rate, cost, and meaningful-model premises are explicit, including the finite-sum rho = 1 case.

**Reviewer check:** ☐ Matches source input

*If left unchecked, explain the discrepancy or uncertainty below.*

**Reviewer annotation**

# Review row 5

**Source locator:** cited publication, lines 296-338

## Verbatim source input

The newly arrived customer weighs the two alternatives-to join or not to join
the queue-by the net gains associated with them. The net gain, in the first case,
is equal to
(t2)

Gi = R- (i+ 1)C-.

In the alternative case the net gain is zero. Hence self interest is served if a strategy
is established in the following fashion. An integer, ns,is found which satisfies simultaneously two inequalities
(13)

R-n

C-

0

and
(14)

R- (ns+ l)C-

< O.

Inequality (13) pertains to the case where the number of queueing customers
(including the one in service) encountered by the newly arrived customer falls
short of the critical number by one. The customer, of course, is supposed to join
and indeed the inequality is in his favor. Inequality (14) relates to the unfavorable
event: the critical number, ns, of customers is already in the queue. We can incorporate the two inequalities in one expression
(15)

nS <C

1

Alternatively we may express the same idea in different notation
(16)

nS-I[Vs]

where [ ] is the well known bracket function; that is, ns is the largest integer not
exceeding vs.
We note that the critical number, ns, derived by "actualizing self interest" depends on ,u,R, and C, but not on the arrival intensity A.This fact alone sufficesbefore pursuing further detailed investigation-to throw serious doubt on the

[Next verbatim source excerpt]

narrow self interest-the facilities of the system are overly congested. To arrive at
an ameliorated state of affairs it is necessary to reduce the strategy n from n, to no.
This can be done in two distinct ways: either through an administrative rule to
the effect that the maximally permissible queue size should be smaller5 than a
prima facie admissible number n8; or, alternatively, a toll 0 is imposed on customers
joining the queue and their (individually) expected net gain is reduced in such a way
that no is the current criterion of newly arrived customers based on their present
comparison of alternatives.
What is the optimal value, 0*, or rather the optimal range, of the toll? Clearly
this is given by

(26)

-(v5-no-1)

(

=R-

1

< 0*

R - Cn = At (V5-no)

If a toll taken from this range is levied on customers joining the queue, the combined income (in unit time) of customers and the revenue agency is maximized.
We might explicitly mention that expenditure incurred in toll collection and in
information processing is considered negligible in this presentation.
Clearly, if the toll revenue may be used for redistribution of income among the
population or for socially useful purposes the proposed imposition of tolls is an
optimal procedure.

[Next verbatim source excerpt]

Secondly, for our model to make sense it is required that under favorable circumstances a customer will desire to queue up for (completion of) service. Hence if a
newly arrived customer encounters a completely empty service station he should
make the "pro-queueing" choice. His expected loss is given by CV' whereas his
reward to be collected at the end of service equals R. If the dimensionless quantity
(Rp/C) is denoted by v, it is clear that in meaningful models the following inequality

must hold
(1)

VS= >C ) 1.

If inequality (1) does not hold the optimal policy is to disband the service station and divert the customer stream altogether.

[Next verbatim source excerpt]

If the viewpoint is taken that the expected sum of the net gains accruing to customers in unit time is the public good which should be optimized, we must proceed in a different mode from that outlined in the previous section. We note that expected total net gain, P, under some strategy n is given by
(17)

P =(-C)R=-RCE {i} =

(l -Pn)-Cq

p
-AR l
pna cL1P

n pn+l

[form-feed in source extraction]
20

P. NAOR

By some elementary (but lengthy) considerations it can be shown that the function P in its dependence on n is "discretely unimodal" or, in other words, a local maximum is a global maximum. Hence we seek that strategy, no say, which is associated with two inequalities

[Next verbatim source excerpt]

To deal with (22) it will be convenient to investigate a function,

(23)

Vs= [vo( (-p)-p

(' -P vo)] (1 _p)-

2

of two independent variables p (> 0) and vo() 1). We note, in passing, that no true singularity exists for this function if p = 1; rather the function is well behaved and a nonzero and finite function value exists at that value of p, to wit: vs= (vo(vo+ 1)/2).
Next we study a setting in which the value of p is arbitrary (positive) but fixed; we note that vs is a boundlessly increasing function of vo. Hence the integers between which vo lies (viewed now as a function of v, and p) will obey the inequalities associated with (22). As a result we arrive at
(24)

no = [vo],

an expression which is completely analogous to (18). Further manipulations lead to the inequality
(25)

VO< Vs

where the equality sign holds only4 if vs equals unity.

## Expanded PaperInterface specification

```
∀ (rho reward queueCost serviceRate toll : ℝ) (socialCapacity : ℕ),
  0 < rho →
    0 < queueCost →
      0 < serviceRate →
        1 < reward * serviceRate / queueCost →
          (∀ (capacity : ℕ),
              reward * (rho * serviceRate) *
                    (1 -
                      if capacity ≤ capacity then rho ^ capacity / ∑ i ∈ Finset.range (capacity + 1), rho ^ i else 0) -
                  queueCost *
                    ((∑ state ∈ Finset.range (capacity + 1), ↑state * rho ^ state) /
                      ∑ state ∈ Finset.range (capacity + 1), rho ^ state) ≤
                reward * (rho * serviceRate) *
                    (1 -
                      if socialCapacity ≤ socialCapacity then
                        rho ^ socialCapacity / ∑ i ∈ Finset.range (socialCapacity + 1), rho ^ i
                      else 0) -
                  queueCost *
                    ((∑ state ∈ Finset.range (socialCapacity + 1), ↑state * rho ^ state) /
                      ∑ state ∈ Finset.range (socialCapacity + 1), rho ^ state)) →
            reward - ↑(socialCapacity + 1) * queueCost / serviceRate < toll →
              toll ≤ reward - ↑socialCapacity * queueCost / serviceRate →
                (∀ (capacity : ℕ),
                    reward * (rho * serviceRate) *
                          (1 -
                            if capacity ≤ capacity then rho ^ capacity / ∑ i ∈ Finset.range (capacity + 1), rho ^ i
                            else 0) -
```

```
          queueCost *
            ((∑ state ∈ Finset.range (capacity + 1), ↑state * rho ^ state) /
              ∑ state ∈ Finset.range (capacity + 1), rho ^ state) -
        toll *
          (rho * serviceRate *
            (1 -
              if capacity ≤ capacity then rho ^ capacity / ∑ i ∈ Finset.range (capacity + 1), rho ^ i
              else 0)) +
      toll *
        (rho * serviceRate *
          (1 -
            if capacity ≤ capacity then rho ^ capacity / ∑ i ∈ Finset.range (capacity + 1), rho ^ i
            else 0)) ≤
    reward * (rho * serviceRate) *
            (1 -
              if socialCapacity ≤ socialCapacity then
                rho ^ socialCapacity / ∑ i ∈ Finset.range (socialCapacity + 1), rho ^ i
              else 0) -
          queueCost *
            ((∑ state ∈ Finset.range (socialCapacity + 1), ↑state * rho ^ state) /
              ∑ state ∈ Finset.range (socialCapacity + 1), rho ^ state) -
        toll *
          (rho * serviceRate *
            (1 -
              if socialCapacity ≤ socialCapacity then
                rho ^ socialCapacity / ∑ i ∈ Finset.range (socialCapacity + 1), rho ^ i
              else 0)) +
      toll *
        (rho * serviceRate *
          (1 -
            if socialCapacity ≤ socialCapacity then
              rho ^ socialCapacity / ∑ i ∈ Finset.range (socialCapacity + 1), rho ^ i
            else 0))) ∧
  ⌊(reward - toll) * serviceRate / queueCost⌋₊ = socialCapacity
```

### Lean proof endpoint (built separately)

Naor1969QueueTolls.socialTollImplementsOptimalThreshold

### Recorded source-to-Spec screening **Verdict:** matches

**Reason:** The inequalities expand to R minus (n + 1) C divided by mu less than toll at most R minus n C divided by mu, the source implementing interval. They imply the intended floor threshold with strict and weak endpoints; with n globally welfare-maximizing, customer income plus toll revenue has the same objective because the transfer cancels. All mathematical premises are explicit.

**Reviewer check:** ☐ Matches source input

*If left unchecked, explain the discrepancy or uncertainty below.*

**Reviewer annotation**

# Review row 6

**Source locator:** cited publication, lines 768-834

## Verbatim source input

The objective function of the toll collector is given by

(27)

M = (x-)

=

Pn

(R-

A ) =

-Pn+

1

V

The maximization of M (which is considered a function of feasible n-s) is
brought about by techniques similar to those used in previous sections. Let the
appropriate value of n be designated by nr, It is then possible to manipulate the
inequalities associated with the maximum value of M in (27) in such a fashion that
a convenient quantity vr-analogous to vo in (23)-should be defined by
1
28~ ~~( ( P, 1)(1 -

(28)

Vr

pv- 1 (1

p)2

Vs

The integer nr which maximizes toll revenue is derived (as analogous integers
before) by applying the bracket function on Vr:
' Such a measure would have to be explained very carefully to the participants since it is in apparent
contradiction with "common sense."

[form-feed in source extraction]
REGULATION

(29)

OF QUEUES

23

nr = [Vrl.

Further (rather tedious) manipulation yields
(30)

Vr<

(given vs > 1),

v0< vS

the approximative meaning of which is the following. Some toll collection may be
beneficial to a queueing system if an appropriate objective function (representing
public good) is chosen. If the toll collecting agency is a decision maker tending to
maximize its own revenue, however, the entrance fees Or, levied on joining customers,
will be too high and social optimality will (frequently) not be attained:

[Next verbatim source excerpt]

(31)

Or =

R

- Cnlr
r-

C
-

(Vs-nr)

[Next verbatim source excerpt]

Secondly, for our model to make sense it is required that under favorable circumstances a customer will desire to queue up for (completion of) service.
↪ Hence if a

newly arrived customer encounters a completely empty service station he should make the "pro-queueing" choice. His expected loss is given by CV' whereas his reward to be collected at the end of service equals R. If the dimensionless quantity (Rp/C) is denoted by v, it is clear that in meaningful models the following inequality must hold
(1)

VS= >C ) 1.

If inequality (1) does not hold the optimal policy is to disband the service station and divert the customer stream altogether.

[Next verbatim source excerpt]

If the viewpoint is taken that the expected sum of the net gains accruing to customers in unit time is the public good which should be optimized, we must proceed in a different mode from that outlined in the previous section. We note that expected total net gain, P, under some strategy n is given by
(17)

P =(-C)R=-RCE {i} =

(l -Pn)-Cq

p
-AR l
pna cL1P

n pn+l

[form-feed in source extraction]
20

P. NAOR

By some elementary (but lengthy) considerations it can be shown that the function P in its dependence on n is "discretely unimodal" or, in other words, a local maximum is a global maximum. Hence we seek that strategy, no say, which is associated with two inequalities

[Next verbatim source excerpt]

To deal with (22) it will be convenient to investigate a function,

(23)

Vs= [vo( (-p)-p

(' -P vo)] (1 _p)-

2

of two independent variables p (> 0) and vo() 1). We note, in passing, that no true singularity exists for this function if p = 1; rather the function is well behaved and a nonzero and finite function value exists at that value of p, to wit: vs= (vo(vo+ 1)/2).
Next we study a setting in which the value of p is arbitrary (positive) but fixed; we note that vs is a boundlessly increasing function of vo. Hence the integers between which vo lies (viewed now as a function of v, and p) will obey the inequalities associated with (22). As a result we arrive at
(24)

no = [vo],

an expression which is completely analogous to (18). Further manipulations lead to the inequality
(25)

VO< Vs

where the equality sign holds only4 if vs equals unity.

## Expanded PaperInterface specification

```
∀ (rho reward queueCost serviceRate : ℝ),
  0 < rho →
    0 < queueCost →
      0 < serviceRate →
        1 < reward * serviceRate / queueCost →
          ∃ (revenueCapacity : ℕ) (socialCapacity : ℕ),
            (∀ (capacity : ℕ),
                ↑capacity ≤ reward * serviceRate / queueCost →
                  rho * serviceRate *
                        (1 -
                          if capacity ≤ capacity then rho ^ capacity / ∑ i ∈ Finset.range (capacity + 1), rho ^ i
                          else 0) *
                      (reward - ↑capacity * queueCost / serviceRate) ≤
                    rho * serviceRate *
                        (1 -
                          if revenueCapacity ≤ revenueCapacity then
                            rho ^ revenueCapacity / ∑ i ∈ Finset.range (revenueCapacity + 1), rho ^ i
                          else 0) *
                      (reward - ↑revenueCapacity * queueCost / serviceRate)) ∧
              (∀ (capacity : ℕ),
                  reward * (rho * serviceRate) *
                        (1 -
                          if capacity ≤ capacity then rho ^ capacity / ∑ i ∈ Finset.range (capacity + 1), rho ^ i
```

```
          else 0) -
        queueCost *
          ((∑ state ∈ Finset.range (capacity + 1), ↑state * rho ^ state) /
            ∑ state ∈ Finset.range (capacity + 1), rho ^ state) ≤
      reward * (rho * serviceRate) *
          (1 -
            if socialCapacity ≤ socialCapacity then
              rho ^ socialCapacity / ∑ i ∈ Finset.range (socialCapacity + 1), rho ^ i
            else 0) -
        queueCost *
          ((∑ state ∈ Finset.range (socialCapacity + 1), ↑state * rho ^ state) /
            ∑ state ∈ Finset.range (socialCapacity + 1), rho ^ state)) ∧
    revenueCapacity ≤ socialCapacity ∧ socialCapacity ≤ ⌊reward * serviceRate / queueCost⌋₊
```

## Lean proof endpoint (built separately)

Naor1969QueueTolls.socialAndRevenueThresholdOrder

## Recorded source-to-Spec screening **Verdict:** matches

**Reason:** The target uses the source welfare objective (admitted reward flow minus expected queueing cost), defines revenue from admitted flow and the fee, and asserts feasible revenue and global welfare maximizers with n_r at most n_o at most floor v_s. Positivity and meaningful-model premises are explicit, and the finite sums remain defined at rho = 1.

**Reviewer check:** ☐ Matches source input

*If left unchecked, explain the discrepancy or uncertainty below.*

**Reviewer annotation**