Stochastic process algebras for quantitative analysis: Lecture 1 — Introduction

Jane Hillston

LFCS, School of Informatics
The University of Edinburgh
Scotland

4th March 2013

Outline

Outline

Quantitative Analysis

Quantitative analysis analysis seeks to associate behaviours of systems with quantified values.

Examples include performance analysis, dependability analysis, availability analysis, etc. of computer and communication systems, and dynamic study of biological processes in systems biology, capacity planning in manufacturing systems, ...

It is sometimes termed non-functional analysis to emphasise that, unlike functional analysis ensuring the correct behaviour of the system is not the primary goal.

Quantitative Analysis

Quantitative analysis analysis seeks to associate behaviours of systems with quantified values.

Examples include performance analysis, dependability analysis, availability analysis, etc. of computer and communication systems, and dynamic study of biological processes in systems biology, capacity planning in manufacturing systems, ...

It is sometimes termed non-functional analysis to emphasise that, unlike functional analysis ensuring the correct behaviour of the system is not the primary goal.

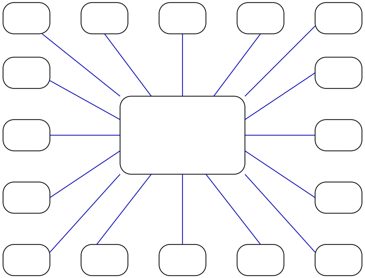
Quantitative Analysis

Quantitative analysis analysis seeks to associate behaviours of systems with quantified values.

Examples include performance analysis, dependability analysis, availability analysis, etc. of computer and communication systems, and dynamic study of biological processes in systems biology, capacity planning in manufacturing systems, ...

It is sometimes termed non-functional analysis to emphasise that, unlike functional analysis ensuring the correct behaviour of the system is not the primary goal.

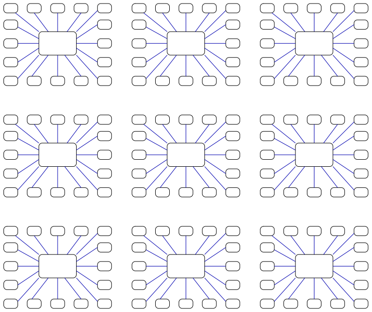
Quantitative Modelling: Motivation



Quality of Service issues

- Can the server maintain reasonable response times?

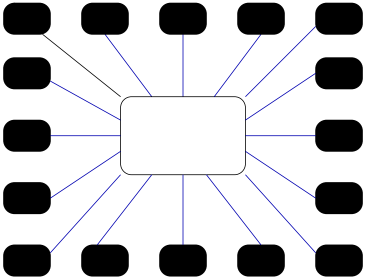
Quantitative Modelling: Motivation



Scalability issues

- How many times do we have to replicate this service to support all of the subscribers?

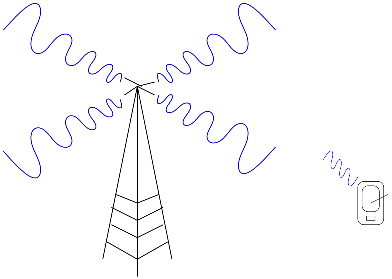
Quantitative Modelling: Motivation



Scalability issues

- Will the server withstand a distributed denial of service attack?

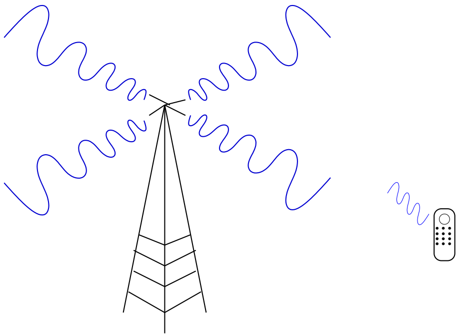
Quantitative Modelling: Motivation



Service-level agreements

- What percentage of downloads do complete within the time we advertised?

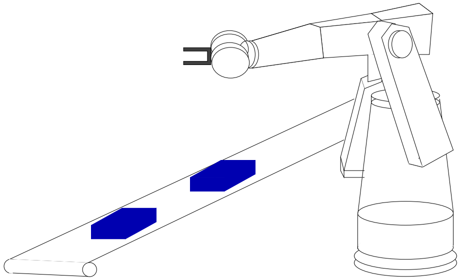
Quantitative Modelling: Motivation



System Configuration

- How many frequencies do you need to keep blocking probabilities low?

Quantitative Modelling: Motivation



System Tuning

- What speed of conveyor belt will minimize robot idle time and maximize throughput whilst avoiding lost widgets?

Performance Analysis

Performance modelling is concerned with the dynamic behaviour of systems and quantified assessment of that behaviour.

There are often conflicting interests at play:

- Users typically want to optimise external measurements of the dynamics such as response time (as small as possible), throughput (as high as possible) or blocking probability (preferably zero);
- In contrast, system managers may seek to optimize internal measurements of the dynamics such as utilisation (reasonably high, but not too high), idle time (as small as possible) or failure rates (as low as possible).

Performance Analysis

Performance modelling is concerned with the dynamic behaviour of systems and quantified assessment of that behaviour.

There are often conflicting interests at play:

- Users typically want to optimise external measurements of the dynamics such as response time (as small as possible), throughput (as high as possible) or blocking probability (preferably zero);
- In contrast, system managers may seek to optimize internal measurements of the dynamics such as utilisation (reasonably high, but not too high), idle time (as small as possible) or failure rates (as low as possible).

Performance Analysis

Performance modelling is concerned with the dynamic behaviour of systems and quantified assessment of that behaviour.

There are often conflicting interests at play:

- Users typically want to optimise external measurements of the dynamics such as response time (as small as possible), throughput (as high as possible) or blocking probability (preferably zero);
- In contrast, system managers may seek to optimize internal measurements of the dynamics such as utilisation (reasonably high, but not too high), idle time (as small as possible) or failure rates (as low as possible).

Key notions

A model can be constructed to represent some aspect of the dynamic behaviour of a system.

Once constructed, such a model becomes a tool with which we can investigate the behaviour of the system.

Key notions

A model can be constructed to represent some aspect of the dynamic behaviour of a system.

Once constructed, such a model becomes a tool with which we can investigate the behaviour of the system.

Modelling computer systems: the challenges

- Physical distance
  - Network latency
- Partial failures
  - Server may be down
  - Routers may be down
- Scale
  - Workload characterisation
- Resource sharing
  - Network contention
  - CPU load

Modelling computer systems: the challenges

- Physical distance
  - Network latency
- Partial failures
  - Server may be down
  - Routers may be down
- Scale
  - Workload characterisation
- Resource sharing
  - Network contention
  - CPU load

Modelling computer systems: the challenges

- Physical distance
  - Network latency
- Partial failures
  - Server may be down
  - Routers may be down
- Scale
  - Workload characterisation
- Resource sharing
  - Network contention
  - CPU load

Modelling computer systems: the challenges

- Physical distance
  - Network latency
- Partial failures
  - Server may be down
  - Routers may be down
- Scale
  - Workload characterisation
- Resource sharing
  - Network contention
  - CPU load

Modelling computer systems: the challenges

- Physical distance — need to represent time
  - Network latency
- Partial failures
  - Server may be down
  - Routers may be down
- Scale
  - Workload characterisation
- Resource sharing
  - Network contention
  - CPU load

Modelling computer systems: the challenges

- Physical distance — need to represent time
  - Network latency
- Partial failures — randomness and probability
  - Server may be down
  - Routers may be down
- Scale
  - Workload characterisation
- Resource sharing
  - Network contention
  - CPU load

Modelling computer systems: the challenges

- Physical distance — need to represent time
  - Network latency
- Partial failures — randomness and probability
  - Server may be down
  - Routers may be down
- Scale — need to quantify population sizes
  - Workload characterisation
- Resource sharing
  - Network contention
  - CPU load

Modelling computer systems: the challenges

- Physical distance — need to represent time
  - Network latency
- Partial failures — randomness and probability
  - Server may be down
  - Routers may be down
- Scale — need to quantify population sizes
  - Workload characterisation
- Resource sharing — need to express percentages
  - Network contention
  - CPU load

Modelling computer systems: the challenges

Time	What representation of time will we use?
Randomness	What kind of random number distributions will we use?
Probability	How can we have probabilities in the model without uncertainty in the results?
Scale	How can we escape the state-space explosion problem?
Percentages	What can it mean to have a fraction of a process?

Outline

Operational Laws

- Operational laws are simple equations may be regarded as a very abstract model of the average behaviour of almost any system, based on the operational view of the system.
- The laws are very general and make almost no assumptions about the behaviour of the random variables characterising the system.
- Another advantage of the laws is their simplicity: this means that they can be applied quickly without detailed knowledge. We will use them sometimes to derive further data from the output observed from models.

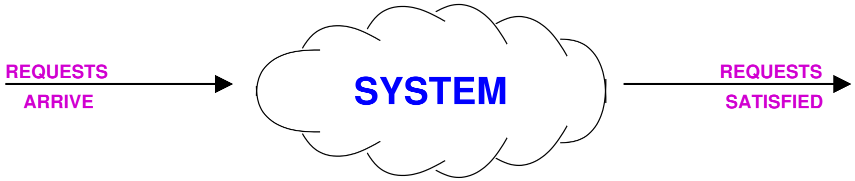
Operational Laws

- Operational laws are simple equations may be regarded as a very abstract model of the average behaviour of almost any system, based on the operational view of the system.
- The laws are very general and make almost no assumptions about the behaviour of the random variables characterising the system.
- Another advantage of the laws is their simplicity: this means that they can be applied quickly without detailed knowledge. We will use them sometimes to derive further data from the output observed from models.

Operational Laws

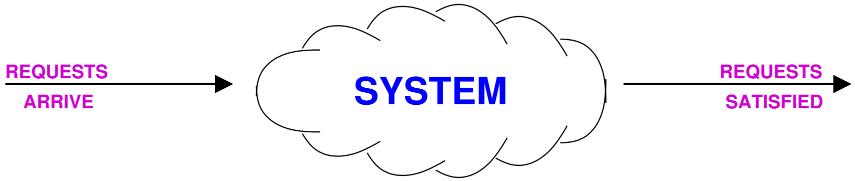
- Operational laws are simple equations may be regarded as a very abstract model of the average behaviour of almost any system, based on the operational view of the system.
- The laws are very general and make almost no assumptions about the behaviour of the random variables characterising the system.
- Another advantage of the laws is their simplicity: this means that they can be applied quickly without detailed knowledge. We will use them sometimes to derive further data from the output observed from models.

Observable variables



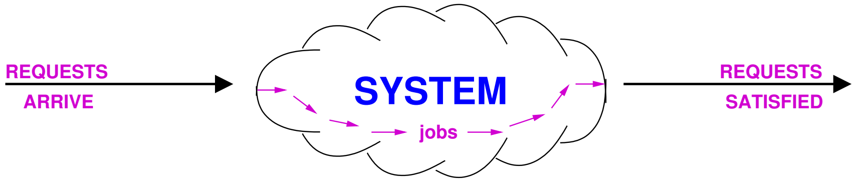
- Operational laws are based on observable variables — values which we could derive from watching a system over a finite period of time.
- We assume that the system receives requests from its environment.
- Each request generates a job or customer within the system.
- When the job has been processed the system responds to the environment with the completion of the corresponding request.

Observable variables



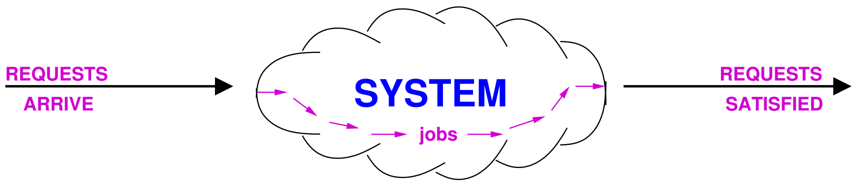
- Operational laws are based on observable variables — values which we could derive from watching a system over a finite period of time.
- We assume that the system receives requests from its environment.
- Each request generates a job or customer within the system.
- When the job has been processed the system responds to the environment with the completion of the corresponding request.

Observable variables



- Operational laws are based on observable variables — values which we could derive from watching a system over a finite period of time.
- We assume that the system receives requests from its environment.
- Each request generates a job or customer within the system.
- When the job has been processed the system responds to the environment with the completion of the corresponding request.

Observable variables



- Operational laws are based on observable variables — values which we could derive from watching a system over a finite period of time.
- We assume that the system receives requests from its environment.
- Each request generates a job or customer within the system.
- When the job has been processed the system responds to the environment with the completion of the corresponding request.

Observations and measurements

If we observed such an abstract system we might measure the following quantities:

- T, the length of time we observe the system;
- A, the number of request arrivals we observe;
- C, the number of request completions we observe;
- B, the total amount of time during which the system is busy ($B \leq T$);
- N, the average number of jobs in the system.

Observations and measurements

If we observed such an abstract system we might measure the following quantities:

- T, the length of time we observe the system;
- A, the number of request arrivals we observe;
- C, the number of request completions we observe;
- B, the total amount of time during which the system is busy ($B \leq T$);
- N, the average number of jobs in the system.

Observations and measurements

If we observed such an abstract system we might measure the following quantities:

- T, the length of time we observe the system;
- A, the number of request arrivals we observe;
- C, the number of request completions we observe;
- B, the total amount of time during which the system is busy ($B \leq T$);
- N, the average number of jobs in the system.

Observations and measurements

If we observed such an abstract system we might measure the following quantities:

- T, the length of time we observe the system;
- A, the number of request arrivals we observe;
- C, the number of request completions we observe;
- B, the total amount of time during which the system is busy ($B \leq T$);
- N, the average number of jobs in the system.

Observations and measurements

If we observed such an abstract system we might measure the following quantities:

- T, the length of time we observe the system;
- A, the number of request arrivals we observe;
- C, the number of request completions we observe;
- B, the total amount of time during which the system is busy ($B \leq T$);
- N, the average number of jobs in the system.

Four important quantities

From these observed values we can derive the following four important quantities:

$\lambda = A/T$, the arrival rate;

$X = C/T$, the throughput or completion rate,

$U = B/T$, the utilisation;

$S = B/C$, the mean service time per completed job.

Four important quantities

From these observed values we can derive the following four important quantities:

$\lambda = A/T$, the arrival rate;

$X = C/T$, the throughput or completion rate,

$U = B/T$, the utilisation;

$S = B/C$, the mean service time per completed job.

Four important quantities

From these observed values we can derive the following four important quantities:

$$\begin{aligned} \lambda &= A/T, && \text{the arrival rate;} \\ X &= C/T, && \text{the throughput or completion rate,} \\ U &= B/T, && \text{the utilisation;} \\ S &= B/C, && \text{the mean service time per completed job.} \end{aligned}$$

Four important quantities

From these observed values we can derive the following four important quantities:

$$\begin{aligned} \lambda &= A/T, && \text{the arrival rate;} \\ X &= C/T, && \text{the throughput or completion rate,} \\ U &= B/T, && \text{the utilisation;} \\ S &= B/C, && \text{the mean service time per completed job.} \end{aligned}$$

Job flow balance

- We will assume that the system is job flow balanced. This means that the number of arrivals is equal to the number of completions during an observation period, i.e. $A = C$.
- This is a testable assumption because an analyst can always test whether the assumption holds.
- Note that if the system is job flow balanced the arrival rate will be the same as the completion rate, that is, $\lambda = X$.

Job flow balance

- We will assume that the system is job flow balanced. This means that the number of arrivals is equal to the number of completions during an observation period, i.e. $A = C$.
- This is a testable assumption because an analyst can always test whether the assumption holds.
- Note that if the system is job flow balanced the arrival rate will be the same as the completion rate, that is, $\lambda = X$.

Job flow balance

- We will assume that the system is job flow balanced. This means that the number of arrivals is equal to the number of completions during an observation period, i.e. $A = C$.
- This is a testable assumption because an analyst can always test whether the assumption holds.
- Note that if the system is job flow balanced the arrival rate will be the same as the completion rate, that is, $\lambda = X$.

Little's Law

Little's Law

$$N = XW$$

The average number of jobs in a system is equal to the product of the throughput of the system and the average time spent in that system by a job.

Example

Consider a disk that serves 40 requests/second ($X = 40$) and suppose that on average there are 4 requests present in the disk system (waiting to be served or in service) ($N = 4$).

Little's law tells us that the average time spent at the disk by a request must be $4/40 = 0.1$ seconds.

If we know that each request requires 0.0225 seconds of disk service we can then deduce that the average queueing time is 0.0775 seconds.

Example

Consider a disk that serves 40 requests/second ($X = 40$) and suppose that on average there are 4 requests present in the disk system (waiting to be served or in service) ($N = 4$).

Little's law tells us that the average time spent at the disk by a request must be $4/40 = 0.1$ seconds.

If we know that each request requires 0.0225 seconds of disk service we can then deduce that the average queueing time is 0.0775 seconds.

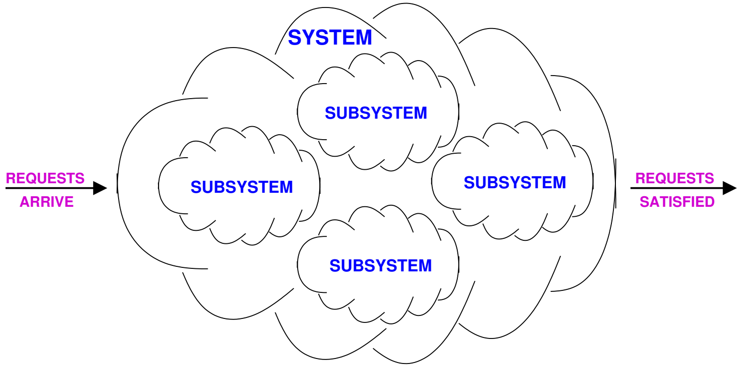
Example

Consider a disk that serves 40 requests/second ($X = 40$) and suppose that on average there are 4 requests present in the disk system (waiting to be served or in service) ($N = 4$).

Little's law tells us that the average time spent at the disk by a request must be $4/40 = 0.1$ seconds.

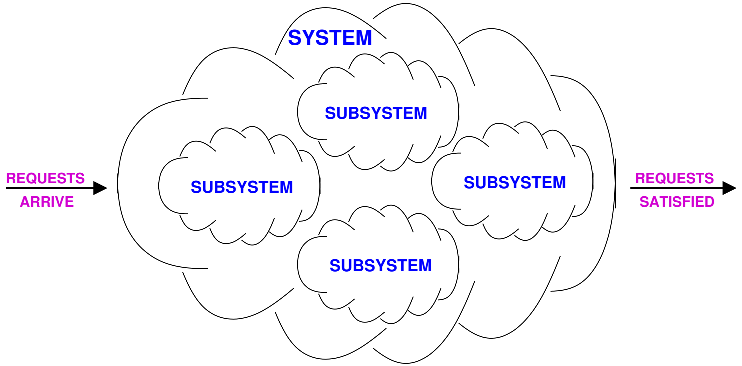
If we know that each request requires 0.0225 seconds of disk service we can then deduce that the average queueing time is 0.0775 seconds.

Subsystems within Systems



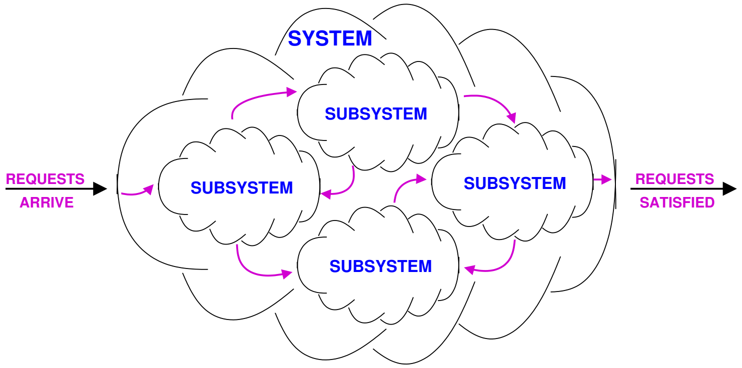
- A system may be regarded as being made up of a number of devices or resources.
- Each of these may be treated as a system in its own right from the perspective of operational laws.

Subsystems within Systems



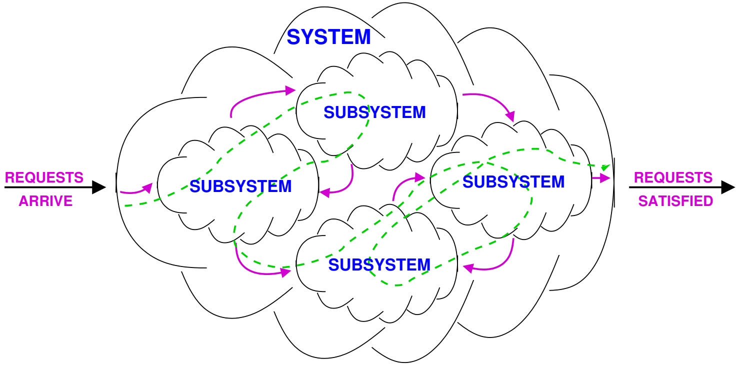
- A system may be regarded as being made up of a number of devices or resources.
- Each of these may be treated as a system in its own right from the perspective of operational laws.

Subsystems within Systems



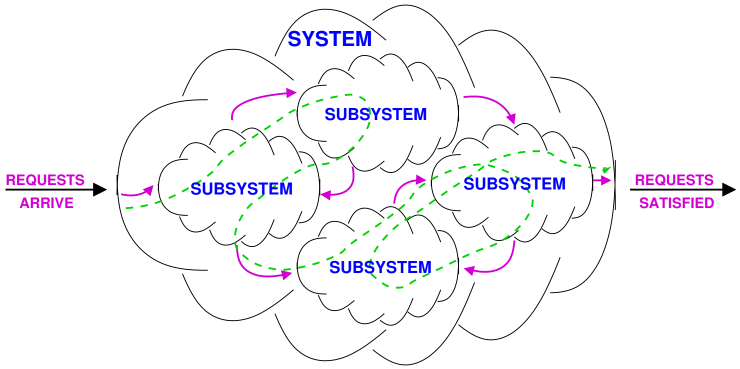
An external request generates a job within the system; this job may then circulate between the resources until all necessary processing has been done; as it arrives at each resource it is treated as a request, generating a job internal to that resource.

Subsystems within Systems



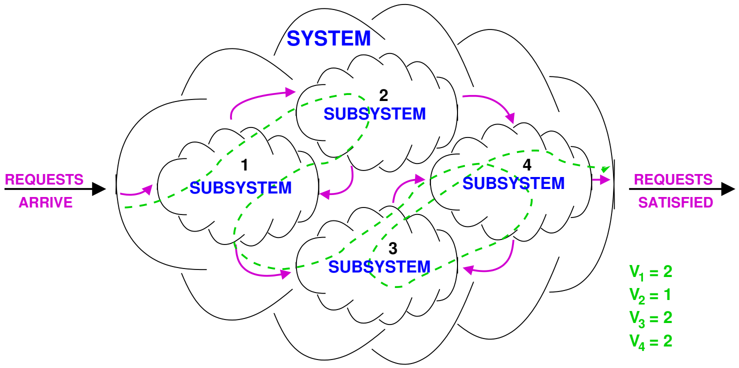
An external request generates a job within the system; this job may then circulate between the resources until all necessary processing has been done; as it arrives at each resource it is treated as a request, generating a job internal to that resource.

Visit count



In an observation interval we can count not only completions external to the system, but also the number of completions at each resource within the system.

Visit count



We define the visit count, V_i, of the ith resource to be the ratio of the number of completions at that resource to the number of system completions $V_i \equiv C_i / C$.

Visit count: example

For example, if, during an observation interval, we measure 10 system completions and 150 completions at a specific disk, then on the average each system-level request requires 15 disk operations.

Forced Flow Law

The forced flow law captures the relationship between the different components within a system. It states that the throughputs or flows, in all parts of a system must be proportional to one another.

Forced Flow Law

$$X_i = XV_i$$

The throughput at the ith resource is equal to the product of the throughput of the system and the visit count at that resource.

Forced Flow Law

The forced flow law captures the relationship between the different components within a system. It states that the throughputs or flows, in all parts of a system must be proportional to one another.

Forced Flow Law

$$X_i = XV_i$$

The throughput at the ith resource is equal to the product of the throughput of the system and the visit count at that resource.

Example

- Consider a robotic workcell within a computerised manufacturing system which processes widgets.
- Suppose that processing each widget requires 4 accesses to the lathe and 2 accesses to the press.
- We know that the lathe processes 8 widgets in a minute and we want to know the throughput of the press.
- The throughput of the workcell will be proportional to the lathe throughput, i.e. $X = X_{lathe}/V_{lathe} = 8/4 = 2$.
- The throughput of the press will be $X_{press} = X \times V_{press} = 2 \times 2 = 4$.
- Thus the press throughput is 4 widgets per minute.

Example

- Consider a robotic workcell within a computerised manufacturing system which processes widgets.
- Suppose that processing each widget requires 4 accesses to the lathe and 2 accesses to the press.
- We know that the lathe processes 8 widgets in a minute and we want to know the throughput of the press.
- The throughput of the workcell will be proportional to the lathe throughput, i.e. $X = X_{lathe}/V_{lathe} = 8/4 = 2$.
- The throughput of the press will be $X_{press} = X \times V_{press} = 2 \times 2 = 4$.
- Thus the press throughput is 4 widgets per minute.

Example

- Consider a robotic workcell within a computerised manufacturing system which processes widgets.
- Suppose that processing each widget requires 4 accesses to the lathe and 2 accesses to the press.
- We know that the lathe processes 8 widgets in a minute and we want to know the throughput of the press.
- The throughput of the workcell will be proportional to the lathe throughput, i.e. $X = X_{lathe}/V_{lathe} = 8/4 = 2$.
- The throughput of the press will be $X_{press} = X \times V_{press} = 2 \times 2 = 4$.
- Thus the press throughput is 4 widgets per minute.

Example

- Consider a robotic workcell within a computerised manufacturing system which processes widgets.
- Suppose that processing each widget requires 4 accesses to the lathe and 2 accesses to the press.
- We know that the lathe processes 8 widgets in a minute and we want to know the throughput of the press.
- The throughput of the workcell will be proportional to the lathe throughput, i.e. $X = X_{lathe}/V_{lathe} = 8/4 = 2$.
- The throughput of the press will be $X_{press} = X \times V_{press} = 2 \times 2 = 4$.
- Thus the press throughput is 4 widgets per minute.

Example

- Consider a robotic workcell within a computerised manufacturing system which processes widgets.
- Suppose that processing each widget requires 4 accesses to the lathe and 2 accesses to the press.
- We know that the lathe processes 8 widgets in a minute and we want to know the throughput of the press.
- The throughput of the workcell will be proportional to the lathe throughput, i.e. $X = X_{lathe}/V_{lathe} = 8/4 = 2$.
- The throughput of the press will be $X_{press} = X \times V_{press} = 2 \times 2 = 4$.
- Thus the press throughput is 4 widgets per minute.

Example

- Consider a robotic workcell within a computerised manufacturing system which processes widgets.
- Suppose that processing each widget requires 4 accesses to the lathe and 2 accesses to the press.
- We know that the lathe processes 8 widgets in a minute and we want to know the throughput of the press.
- The throughput of the workcell will be proportional to the lathe throughput, i.e. $X = X_{lathe}/V_{lathe} = 8/4 = 2$.
- The throughput of the press will be $X_{press} = X \times V_{press} = 2 \times 2 = 4$.
- Thus the press throughput is 4 widgets per minute.

Interactive Response Time Law

- Back when most processing was done on shared mainframes, think time, Z, was quite literally the length of time that a programmer spent thinking before submitting another job.
- More generally in interactive systems, jobs spend time in the system not engaged in processing, or waiting for processing: this may be because of interaction with a human user, or may be for some other reason.
- The key feature of such a system is that the residence time can no longer be taken as a true reflection of the response time of the system.

Interactive Response Time Law

- Back when most processing was done on shared mainframes, think time, Z, was quite literally the length of time that a programmer spent thinking before submitting another job.
- More generally in interactive systems, jobs spend time in the system not engaged in processing, or waiting for processing: this may be because of interaction with a human user, or may be for some other reason.
- The key feature of such a system is that the residence time can no longer be taken as a true reflection of the response time of the system.

Interactive Response Time Law

- Back when most processing was done on shared mainframes, think time, Z, was quite literally the length of time that a programmer spent thinking before submitting another job.
- More generally in interactive systems, jobs spend time in the system not engaged in processing, or waiting for processing: this may be because of interaction with a human user, or may be for some other reason.
- The key feature of such a system is that the residence time can no longer be taken as a true reflection of the response time of the system.

Example

- For example, if we are studying a cluster of workstations with a central file server to investigate the load on the file server, the think time might represent the average time that each workstation spends processing locally without access to the file server.
- At the end of this non-processing period the job generates a fresh request.

Example

- For example, if we are studying a cluster of workstations with a central file server to investigate the load on the file server, the think time might represent the average time that each workstation spends processing locally without access to the file server.
- At the end of this non-processing period the job generates a fresh request.

Think time, residence time, response time

- The think time represents the time between processing being completed and the job becoming available as a request again.
- Thus the residence time of the job, as calculated by Little's Law as the time from arrival to completion, is greater than the system's response time.

Think time, residence time, response time

- The think time represents the time between processing being completed and the job becoming available as a request again.
- Thus the residence time of the job, as calculated by Little's Law as the time from arrival to completion, is greater than the system's response time.

Interactive Response Time Law

The interactive response time law reflects this: it calculates the response time, R as follows:

Interactive Response Time Law

$$R = N/X - Z$$

The response time in an interactive system is the residence time minus the think time.

Note that if the think time is zero, $Z = 0$ and $R = W$, then the interactive response time law simply becomes Little's Law.

Interactive Response Time Law

The interactive response time law reflects this: it calculates the response time, R as follows:

Interactive Response Time Law

$$R = N/X - Z$$

The response time in an interactive system is the residence time minus the think time.

Note that if the think time is zero, $Z = 0$ and $R = W$, then the interactive response time law simply becomes Little's Law.

Interactive Response Time Law: Example

- Suppose that the library catalogue system has 64 interactive users connected via Browsers, that the average think time is 30 seconds, and that system throughput is 2 interactions/second.
- Then the interactive response time law tells us that the response time must be $64/2 - 30 = 2$ seconds.

Interactive Response Time Law: Example

- Suppose that the library catalogue system has 64 interactive users connected via Browsers, that the average think time is 30 seconds, and that system throughput is 2 interactions/second.
- Then the interactive response time law tells us that the response time must be $64/2 - 30 = 2$ seconds.

Outline

The discrete event view

In general we wish to have a more detailed (mechanistic) view of the system under study than that presented by the operational laws. In this course we will consider discrete event systems.

The state of the system is characterised by variables which take distinct values and which change by discrete events, i.e. at a distinct time something happens within the system which results in a change in one or more of the state variables.

The discrete event view

In general we wish to have a more detailed (mechanistic) view of the system under study than that presented by the operational laws. In this course we will consider discrete event systems.

The state of the system is characterised by variables which take distinct values and which change by discrete events, i.e. at a distinct time something happens within the system which results in a change in one or more of the state variables.

The discrete event view: example

We might be interested in the number of nodes in a communication network which are currently waiting to send a message, N.

- If a node, which was not previously waiting, generates a message and is now waiting to send then $N \to N + 1$, or
- If a node, which was previously waiting, successfully transmits its message then $N \to N - 1$.

Discrete time vs. Continuous time

Within discrete event systems there is a distinction between a discrete time representation and a continuous time representation:

Discrete time: such models only consider the system at predetermined moments in time, which are typically evenly spaced, eg. at each clock "tick".

Continuous time: such models consider the system at the time of each event so the time parameter in such models is conceptually continuous.

At levels of abstraction above the hardware clock continuous time models are generally appropriate for computer and communication systems.

Discrete time vs. Continuous time

Within discrete event systems there is a distinction between a discrete time representation and a continuous time representation:

Discrete time: such models only consider the system at predetermined moments in time, which are typically evenly spaced, eg. at each clock "tick".

Continuous time: such models consider the system at the time of each event so the time parameter in such models is conceptually continuous.

At levels of abstraction above the hardware clock continuous time models are generally appropriate for computer and communication systems.

Discrete time vs. Continuous time

Within discrete event systems there is a distinction between a discrete time representation and a continuous time representation:

Discrete time: such models only consider the system at predetermined moments in time, which are typically evenly spaced, eg. at each clock "tick".

Continuous time: such models consider the system at the time of each event so the time parameter in such models is conceptually continuous.

At levels of abstraction above the hardware clock continuous time models are generally appropriate for computer and communication systems.

Quantitative modelling

When systems are modelled to verify their functional behaviour (correctness), all definite values are abstracted away — qualitative modelling.

In contrast, performance modelling is quantitative modelling as we must take into account explicit values for time (latency, service time etc.) and probability (choices, alternative outcomes, mixed workload).

Probability will be used to represent randomness (e.g. from human users) but also as an abstraction over unknown values (e.g. service times).

Quantitative modelling

When systems are modelled to verify their functional behaviour (correctness), all definite values are abstracted away — qualitative modelling.

In contrast, performance modelling is quantitative modelling as we must take into account explicit values for time (latency, service time etc.) and probability (choices, alternative outcomes, mixed workload).

Probability will be used to represent randomness (e.g. from human users) but also as an abstraction over unknown values (e.g. service times).

Quantitative modelling

When systems are modelled to verify their functional behaviour (correctness), all definite values are abstracted away — qualitative modelling.

In contrast, performance modelling is quantitative modelling as we must take into account explicit values for time (latency, service time etc.) and probability (choices, alternative outcomes, mixed workload).

Probability will be used to represent randomness (e.g. from human users) but also as an abstraction over unknown values (e.g. service times).

Outline

Random experiments and events

- To apply probability theory to the process under study, we view it as a random experiment.
- The sample space of a random experiment is the set of all individual outcomes of the experiment.
- These individual outcomes are also called sample points or elementary events.
- An event is a subset of a sample space.

Random experiments and events

- To apply probability theory to the process under study, we view it as a random experiment.
- The sample space of a random experiment is the set of all individual outcomes of the experiment.
- These individual outcomes are also called sample points or elementary events.
- An event is a subset of a sample space.

Random experiments and events

- To apply probability theory to the process under study, we view it as a random experiment.
- The sample space of a random experiment is the set of all individual outcomes of the experiment.
- These individual outcomes are also called sample points or elementary events.
- An event is a subset of a sample space.

Random experiments and events

- To apply probability theory to the process under study, we view it as a random experiment.
- The sample space of a random experiment is the set of all individual outcomes of the experiment.
- These individual outcomes are also called sample points or elementary events.
- An event is a subset of a sample space.

Random variables

We are interested in the dynamics of a system as events happen over time.

A function which associates a (real-valued) number with the outcome of an experiment is known as a random variable.

Formally, a random variable X is a real-valued function defined on a sample space Ω.

Measurable functions

If X is a random variable, and x is a real number, we write $X \leq x$ for the event

$$\{\, \omega : \omega \in \Omega \text{ and } X(\omega) \leq x \,\}$$

and we write $X = x$ for the event

$$\{\, \omega : \omega \in \Omega \text{ and } X(\omega) = x \,\}$$

We require that for a random variable X, the set $X \leq x$ is an event for each real x. This is necessary so that probability calculations can be made. A function having this property is said to be a measurable function or measurable in the Borel sense.

Measurable functions

If X is a random variable, and x is a real number, we write $X \leq x$ for the event

$$\{\, \omega : \omega \in \Omega \text{ and } X(\omega) \leq x \,\}$$

and we write $X = x$ for the event

$$\{\, \omega : \omega \in \Omega \text{ and } X(\omega) = x \,\}$$

We require that for a random variable X, the set $X \leq x$ is an event for each real x. This is necessary so that probability calculations can be made. A function having this property is said to be a measurable function or measurable in the Borel sense.

Measurable functions

If X is a random variable, and x is a real number, we write $X \leq x$ for the event

$$\{\, \omega : \omega \in \Omega \text{ and } X(\omega) \leq x \,\}$$

and we write $X = x$ for the event

$$\{\, \omega : \omega \in \Omega \text{ and } X(\omega) = x \,\}$$

We require that for a random variable X, the set $X \leq x$ is an event for each real x. This is necessary so that probability calculations can be made. A function having this property is said to be a measurable function or measurable in the Borel sense.

Distribution function

For each random variable X we define its distribution function F for each real x by

$$F(x) = \Pr[X \leq x]$$

We associate another function $p(\cdot)$, called the probability mass function, with X (pmf), for each x:

$$p(x) = \Pr[X = x]$$

A random variable X is continuous if $p(x) = 0$ for all real x.

(If X is a continuous random variable, then X can assume infinitely many values, and so it is reasonable that the probability of its assuming any specific value we choose beforehand is zero.)

Distribution function

For each random variable X we define its distribution function F for each real x by

$$F(x) = \Pr[X \leq x]$$

We associate another function $p(\cdot)$, called the probability mass function, with X (pmf), for each x:

$$p(x) = \Pr[X = x]$$

A random variable X is continuous if $p(x) = 0$ for all real x.

(If X is a continuous random variable, then X can assume infinitely many values, and so it is reasonable that the probability of its assuming any specific value we choose beforehand is zero.)

Distribution function

For each random variable X we define its distribution function F for each real x by

$$F(x) = \Pr[X \leq x]$$

We associate another function $p(\cdot)$, called the probability mass function, with X (pmf), for each x:

$$p(x) = \Pr[X = x]$$

A random variable X is continuous if $p(x) = 0$ for all real x.

(If X is a continuous random variable, then X can assume infinitely many values, and so it is reasonable that the probability of its assuming any specific value we choose beforehand is zero.)

Distribution function

For each random variable X we define its distribution function F for each real x by

$$F(x) = \Pr[X \leq x]$$

We associate another function $p(\cdot)$, called the probability mass function, with X (pmf), for each x:

$$p(x) = \Pr[X = x]$$

A random variable X is continuous if $p(x) = 0$ for all real x.

(If X is a continuous random variable, then X can assume infinitely many values, and so it is reasonable that the probability of its assuming any specific value we choose beforehand is zero.)

Exponential random variables, distribution function

The random variable X is said to be an exponential random variable with parameter λ $(\lambda > 0)$ or to have an exponential distribution with parameter λ if it has the distribution function

$$F(x) = \begin{cases} 1 - e^{-\lambda x} & \text{for } x > 0 \\ 0 & \text{for } x \leq 0 \end{cases}$$

Some authors call this distribution the negative exponential distribution.

Exponential random variables, distribution function

The random variable X is said to be an exponential random variable with parameter λ ($\lambda > 0$) or to have an exponential distribution with parameter λ if it has the distribution function

$$F(x) = \begin{cases} 1 - e^{-\lambda x} & \text{for } x > 0 \\ 0 & \text{for } x \leq 0 \end{cases}$$

Some authors call this distribution the negative exponential distribution.

Exponential random variables, density function

The density function $f = dF/dx$ is given by

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

Mean, or expected value

If X is a continuous random variable with density function $f(\cdot)$, we define the mean or expected value of X, $\mu = E[X]$ by

$$\mu = E[X] = \int_{-\infty}^{\infty} xf(x)dx$$

If X is a discrete random variable with probability mass function $p(\cdot)$, we define the mean or expected value of $X \in S$, $\mu = E[X]$ by

$$E(X) = \sum_{x \in S} xp(x)$$

Mean, or expected value

If X is a continuous random variable with density function $f(\cdot)$, we define the mean or expected value of X, $\mu = E[X]$ by

$$\mu = E[X] = \int_{-\infty}^{\infty} x f(x) dx$$

If X is a discrete random variable with probability mass function $p(\cdot)$, we define the mean or expected value of $X \in S$, $\mu = E[X]$ by

$$E(X) = \sum_{x \in S} x p(x)$$

Mean, or expected value, of the exponential distribution

Suppose X has an exponential distribution with parameter $\lambda > 0$.

Then

$$\mu = E[X] = \int_{-\infty}^{\infty} x\lambda e^{-\lambda x} dx = \frac{1}{\lambda}$$

Exponential inter-event time distribution

The time interval between successive events can also be deduced.

Let $F(t)$ be the distribution function of T, the time between events. Consider $\Pr(T > t) = 1 - F(t)$:

$$\begin{aligned}
\Pr(T > t) &= \Pr(\text{No events in an interval of length } t) \\
&= 1 - F(t) \\
&= 1 - (1 - e^{-\lambda t}) \\
&= e^{-\lambda t}
\end{aligned}$$

Memoryless property of the exponential distribution

The memoryless property of the exponential distribution is so called because the time to the next event is independent of when the last event occurred.

Memoryless property of the exponential distribution

Suppose that the last event was at time 0. What is the probability that the next event will be after $t + s$, given that time t has elapsed since the last event, and no events have occurred?

$$\begin{aligned}
\Pr(T > t + s \mid T > t) &= \frac{\Pr(T > t + s \text{ and } T > t)}{\Pr(T > t)} \\
&= \frac{e^{-\lambda(t+s)}}{e^{-\lambda t}} \\
&= e^{-\lambda s}
\end{aligned}$$

This value is independent of t (and so the time already spent has not been remembered).

Memoryless property of the exponential distribution

Suppose that the last event was at time 0. What is the probability that the next event will be after $t + s$, given that time t has elapsed since the last event, and no events have occurred?

$$\begin{aligned}
\Pr(T > t + s \mid T > t) &= \frac{\Pr(T > t + s \text{ and } T > t)}{\Pr(T > t)} \\
&= \frac{e^{-\lambda(t+s)}}{e^{-\lambda t}} \\
&= e^{-\lambda s}
\end{aligned}$$

This value is independent of t (and so the time already spent has not been remembered).

Memoryless property of the exponential distribution

Suppose that the last event was at time 0. What is the probability that the next event will be after $t+s$, given that time t has elapsed since the last event, and no events have occurred?

$$\begin{aligned}\Pr(T > t+s \mid T > t) &= \frac{\Pr(T > t+s \text{ and } T > t)}{\Pr(T > t)} \\ &= \frac{e^{-\lambda(t+s)}}{e^{-\lambda t}} \\ &= e^{-\lambda s}\end{aligned}$$

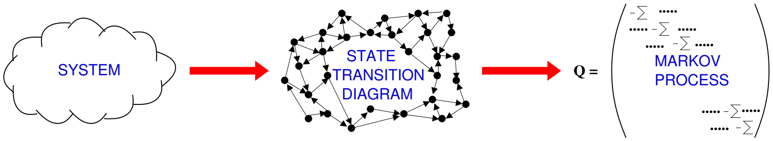
This value is independent of t (and so the time already spent has not been remembered).

Continuous-Time Markov Chains (CTMCs)

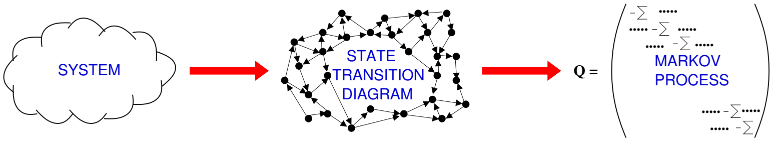
A Markov process with discrete state space and discrete index set is called a Markov chain. The future behaviour of a Markov chain depends only on its current state, and not on how that state was reached. This is the Markov, or memoryless, property.

$$\begin{aligned} & \Pr(X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n, \ldots, X(t_0) = x_0) \\ = \; & \Pr(X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n) \end{aligned}$$

Performance Modelling using CTMC



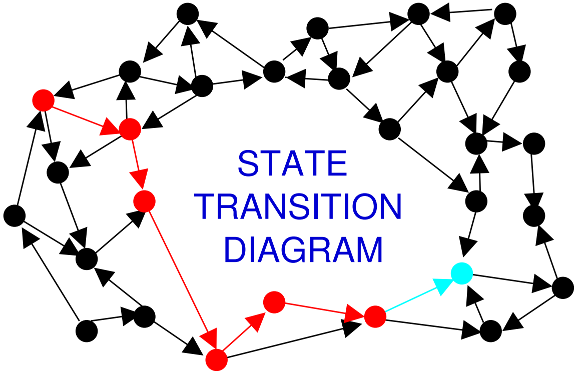
Performance Modelling using CTMC



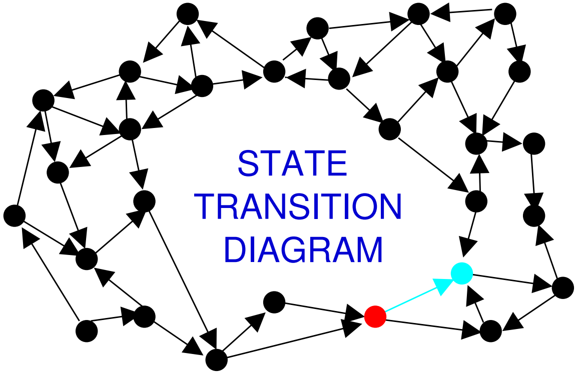
A stochastic process $X(t)$ is a Markov process iff for all $t_0 < t_1 < .... < t_n < t_{n+1}$, the joint probability distribution of $(X(t_0), X(t_1), ..., X(t_n), X(t_{n+1}))$ is such that

$$Pr(X(t_{n+1}) = s_{i_{n+1}} \mid X(t_0) = s_{i_0}, ..., X(t_n) = s_{i_n}) = Pr(X(t_{n+1}) = s_{i_{n+1}} \mid X(t_n) = s_{i_n})$$

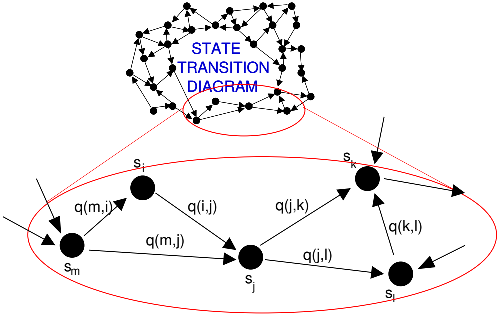
Performance Modelling using CTMC



Performance Modelling using CTMC

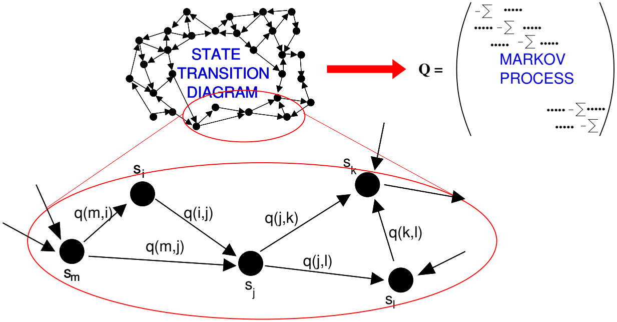


Performance Modelling using CTMC



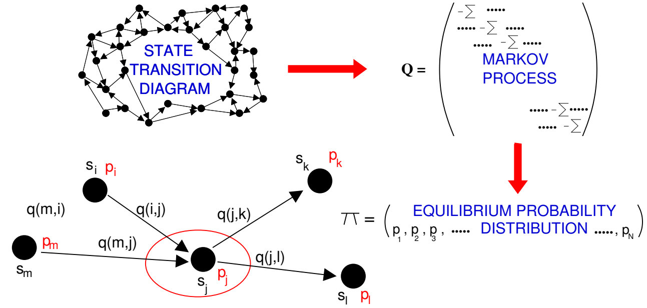
A negative exponentially distributed duration is associated with each transition.

Performance Modelling using CTMC



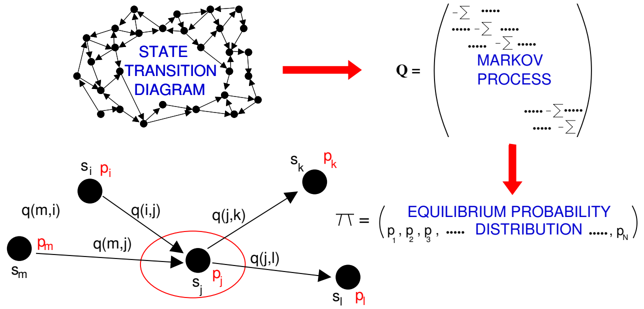
these parameters form the entries of the infinitesimal generator matrix Q

Performance Modelling using CTMC



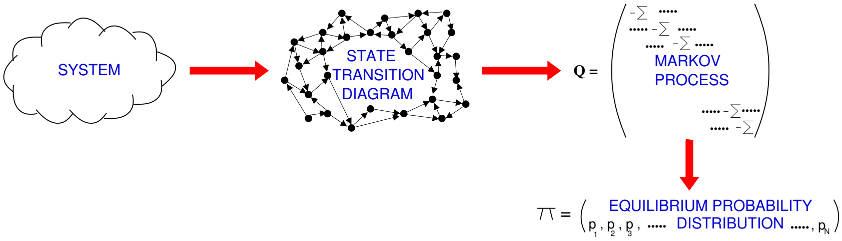
In steady state the probability flux out of a state is balanced by the flux in.

Performance Modelling using CTMC

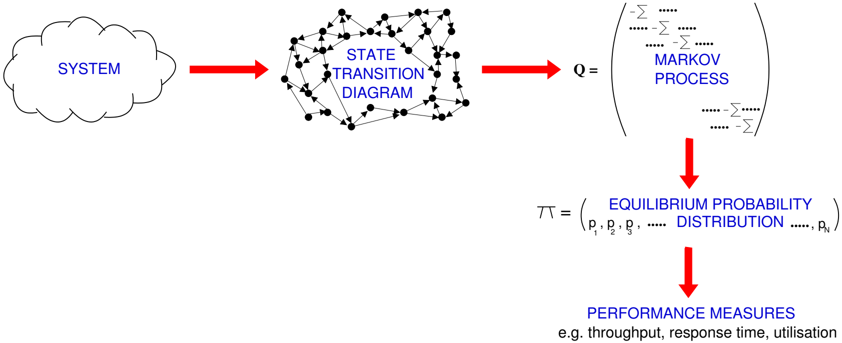


"Global balance equations" captured by $\pi Q = 0$ solved by linear algebra

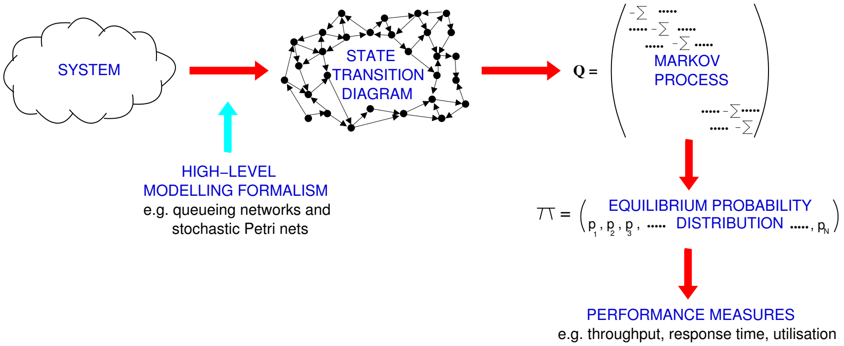
Performance Modelling using CTMC



Performance Modelling using CTMC


SYSTEM
STATE TRANSITION DIAGRAM
Q =
MARKOV PROCESS
π = (p1 , p2 , p3 , ... , pN)
EQUILIBRIUM PROBABILITY DISTRIBUTION
PERFORMANCE MEASURES
e.g. throughput, response time, utilisation

Performance Modelling using CTMC


SYSTEM
STATE TRANSITION DIAGRAM
HIGH–LEVEL MODELLING FORMALISM
e.g. queueing networks and stochastic Petri nets
Q =
$-\sum$ .....
..... $-\sum$ .....
..... $-\sum$ .....
MARKOV PROCESS
..... $-\sum$ .....
..... $-\sum$
π = (p_1, p_2, p_3, ... EQUILIBRIUM PROBABILITY DISTRIBUTION ....., p_N)
PERFORMANCE MEASURES
e.g. throughput, response time, utilisation

Outline

The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov processes.
- Process algebras offered a compositional description technique supported by apparatus for formal reasoning.
- Performance Evaluation Process Algebra (PEPA) sought to address these problems by the introduction of a suitable process algebra.
- We have sought to investigate and exploit the interplay between the process algebra and the continuous time Markov chain (CTMC).

The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov processes.
- Process algebras offered a compositional description technique supported by apparatus for formal reasoning.
- Performance Evaluation Process Algebra (PEPA) sought to address these problems by the introduction of a suitable process algebra.
- We have sought to investigate and exploit the interplay between the process algebra and the continuous time Markov chain (CTMC).

The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov processes.
- Process algebras offered a compositional description technique supported by apparatus for formal reasoning.
- Performance Evaluation Process Algebra (PEPA) sought to address these problems by the introduction of a suitable process algebra.
- We have sought to investigate and exploit the interplay between the process algebra and the continuous time Markov chain (CTMC).

The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov processes.
- Process algebras offered a compositional description technique supported by apparatus for formal reasoning.
- Performance Evaluation Process Algebra (PEPA) sought to address these problems by the introduction of a suitable process algebra.
- We have sought to investigate and exploit the interplay between the process algebra and the continuous time Markov chain (CTMC).

The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov processes.
- Process algebras offered a compositional description technique supported by apparatus for formal reasoning.
- Performance Evaluation Process Algebra (PEPA) sought to address these problems by the introduction of a suitable process algebra.
- We have sought to investigate and exploit the interplay between the process algebra and the continuous time Markov chain (CTMC).

Performance Evaluation Process Algebra

- PEPA (Performance Evaluation Process Algebra) is a high-level modelling language for distributed systems. It can be used to develop models of existing systems (abstraction) or designs for proposed ones (specification).
- PEPA can capture performance information in a process algebra setting. It is a stochastic process algebra.
- For technical details the definitive reference is *A Compositional Approach to Performance Modelling*, Hillston, Cambridge University Press, 1996.

Strengths of stochastic process algebras

SPAs have strengths in the areas of semantic definition, inherent compositionality and the existence of important equivalence relations (including bisimulation). This relation provides the basis for aggregation of PEPA models.

Terminology

The components in a PEPA model engage, cooperatively or individually, in activities.

Each activity has an action type which corresponds to the actions of the system being modelled.

To represent unimportant or unknown actions there is a distinguished action type, τ.

Terminology

The components in a PEPA model engage, cooperatively or individually, in activities.

Each activity has an action type which corresponds to the actions of the system being modelled.

To represent unimportant or unknown actions there is a distinguished action type, τ.

Terminology

The components in a PEPA model engage, cooperatively or individually, in activities.

Each activity has an action type which corresponds to the actions of the system being modelled.

To represent unimportant or unknown actions there is a distinguished action type, τ.

Quantitative aspects

Every activity in PEPA has an associated activity rate which may be any positive real number, or the distinguished symbol "$\top$", meaning unspecified, read as 'top'.

Components and activities are primitives. PEPA also provides a small set of combinators.

Quantitative aspects

Every activity in PEPA has an associated activity rate which may be any positive real number, or the distinguished symbol “$\top$”, meaning unspecified, read as ‘top’.

Components and activities are primitives. PEPA also provides a small set of combinators.

PEPA syntax

$$
\begin{array}{rclr}
S & ::= & (\alpha, r).S & \text{(prefix)} \\
  & \mid & S_1 + S_2 & \text{(choice)} \\
  & \mid & X & \text{(variable)} \\
C & ::= & C_1 \underset{L}{\bowtie} C_2 & \text{(cooperation)} \\
  & \mid & C \,/\, L & \text{(hiding)} \\
  & \mid & S & \text{(sequential)}
\end{array}
$$

PEPA: informal semantics (sequential sublanguage)

$(\alpha, r).S$

The activity (α, r) takes time Δt (drawn from the exponential distribution with parameter r).

$S_1 + S_2$

In this choice either S_1 or S_2 will complete an activity first. The other is discarded.

PEPA: informal semantics (combinators)

$C_1 \underset{L}{\bowtie} C_2$

All activities of C_1 and C_2 with types in L are shared: others remain individual.
NOTATION: write $C_1 \parallel C_2$ if L is empty.

$C \ / \ L$

Activities of C with types in L are hidden (τ type activities) to be thought of as internal delays.

Expansion Law

$$P \underset{L}{\bowtie} Q =$$

$$\sum\{(\alpha, r).(P' \underset{L}{\bowtie} Q) : P \xrightarrow{(\alpha,r)} P';\ \alpha \notin L\} \ +$$

$$\sum\{(\alpha, r).(P \underset{L}{\bowtie} Q') : Q \xrightarrow{(\alpha,r)} Q';\ \alpha \notin L\} \ +$$

$$\sum\{(\alpha, r).(P' \underset{L}{\bowtie} Q') : P \xrightarrow{(\alpha,r_1)} P';\ Q \xrightarrow{(\alpha,r_2)} Q';\ \alpha \in L\}$$

Expansion Law

$$P \underset{L}{\bowtie} Q =$$

$$\sum\{(\alpha, r).(P' \underset{L}{\bowtie} Q) : P \xrightarrow{(\alpha,r)} P';\ \alpha \notin L\} \ +$$

$$\sum\{(\alpha, r).(P \underset{L}{\bowtie} Q') : Q \xrightarrow{(\alpha,r)} Q';\ \alpha \notin L\} \ +$$

$$\sum\{(\alpha, r).(P' \underset{L}{\bowtie} Q') : P \xrightarrow{(\alpha,r_1)} P';\ Q \xrightarrow{(\alpha,r_2)} Q';\ \alpha \in L\}$$

Expansion Law

$$P \underset{L}{\bowtie} Q =$$

$$\sum\{(\alpha, r).(P' \underset{L}{\bowtie} Q) : P \xrightarrow{(\alpha,r)} P'; \alpha \notin L\} +$$

$$\sum\{(\alpha, r).(P \underset{L}{\bowtie} Q') : Q \xrightarrow{(\alpha,r)} Q'; \alpha \notin L\} +$$

$$\sum\{(\alpha, r).(P' \underset{L}{\bowtie} Q') : P \xrightarrow{(\alpha,r_1)} P'; Q \xrightarrow{(\alpha,r_2)} Q'; \alpha \in L\}$$

Expansion Law

$$P \underset{L}{\bowtie} Q =$$

$$\sum\{(\alpha, r).(P' \underset{L}{\bowtie} Q) : P \xrightarrow{(\alpha,r)} P';\ \alpha \notin L\} \ +$$

$$\sum\{(\alpha, r).(P \underset{L}{\bowtie} Q') : Q \xrightarrow{(\alpha,r)} Q';\ \alpha \notin L\} \ +$$

$$\sum\{(\alpha, r).(P' \underset{L}{\bowtie} Q') : P \xrightarrow{(\alpha,r_1)} P';\ Q \xrightarrow{(\alpha,r_2)} Q';\ \alpha \in L\}$$

Example: M/M/1/N/N queue

$$\begin{aligned}
Arrival_0 &\overset{def}{=} (accept, \lambda).Arrival_1 \\
Arrival_i &\overset{def}{=} (accept, \lambda).Arrival_{i+1} + (serve, \top).Arrival_{i-1} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad (0 < i < N) \\
Arrival_N &\overset{def}{=} (serve, \top).Arrival_{N-1} \\
Server &\overset{def}{=} (serve, \mu).Server
\end{aligned}$$

Example: M/M/1/N/N queue

$$\begin{tikzcd}
Queue_0 \arrow[r, bend left, "(accept, \lambda)"] & Queue_1 \arrow[l, bend left, "(serve, \mu)"] \arrow[r, bend left, "(accept, \lambda)"] & \cdots \arrow[l, bend left, "(serve, \mu)"] \arrow[r, bend left, "(accept, \lambda)"] & Queue_{N-1} \arrow[l, bend left, "(serve, \mu)"] \arrow[r, bend left, "(accept, \lambda)"] & Queue_N \arrow[l, bend left, "(serve, \mu)"]
\end{tikzcd}$$

$$Queue_i \equiv Arrival_i \underset{\{serve\}}{\bowtie} Server$$

Synchronisation

What should be the impact of synchronisation on rate? There are many possibilities.

- Restrict synchronisations to have one active partner and one passive partner.
- Choose a function which satisfies a small number of algebraic properties.
- Have the rate limited by the slowest participant in terms of apparent rate. This is the approach adopted by PEPA.

Synchronisation

What should be the impact of synchronisation on rate? There are many possibilities.

- Restrict synchronisations to have one active partner and one passive partner.
- Choose a function which satisfies a small number of algebraic properties.
- Have the rate limited by the slowest participant in terms of apparent rate. This is the approach adopted by PEPA.

Synchronisation

What should be the impact of synchronisation on rate? There are many possibilities.

- Restrict synchronisations to have one active partner and one passive partner.
- Choose a function which satisfies a small number of algebraic properties.
- Have the rate limited by the slowest participant in terms of apparent rate. This is the approach adopted by PEPA.

Bounded capacity

Within the cooperation framework, PEPA assumes bounded capacity: that is, a component cannot be made to perform an activity faster by cooperation, so the rate of a shared activity is the minimum of the apparent rates of the activity in the cooperating components.

Outline

PEPA activities and rates

When enabled, an activity, $a = (\alpha, \lambda)$, will delay for a period determined by its associated distribution function, i.e. the probability that the activity a happens within a period of time of length t is $F_a(t) = 1 - e^{-\lambda t}$.

PEPA activities and rates

- We can think of this as the activity setting a timer whenever it becomes enabled.
- The time allocated to the timer is determined by the rate of the activity.
- If several activities are enabled at the same time each will have its own associated timer.
- When the first timer finishes, that activity takes place—the activity is said to complete or succeed.
- This means that the activity is considered to "happen": an external observer will witness the event of activity of type α.
- An activity may be preempted, or aborted, if another one completes first.

PEPA activities and rates

- We can think of this as the activity setting a timer whenever it becomes enabled.
- The time allocated to the timer is determined by the rate of the activity.
- If several activities are enabled at the same time each will have its own associated timer.
- When the first timer finishes, that activity takes place—the activity is said to complete or succeed.
- This means that the activity is considered to "happen": an external observer will witness the event of activity of type α.
- An activity may be preempted, or aborted, if another one completes first.

PEPA activities and rates

- We can think of this as the activity setting a timer whenever it becomes enabled.
- The time allocated to the timer is determined by the rate of the activity.
- If several activities are enabled at the same time each will have its own associated timer.
- When the first timer finishes, that activity takes place—the activity is said to complete or succeed.
- This means that the activity is considered to "happen": an external observer will witness the event of activity of type α.
- An activity may be preempted, or aborted, if another one completes first.

PEPA activities and rates

- We can think of this as the activity setting a timer whenever it becomes enabled.
- The time allocated to the timer is determined by the rate of the activity.
- If several activities are enabled at the same time each will have its own associated timer.
- When the first timer finishes, that activity takes place—the activity is said to complete or succeed.
- This means that the activity is considered to "happen": an external observer will witness the event of activity of type α.
- An activity may be preempted, or aborted, if another one completes first.

PEPA activities and rates

- We can think of this as the activity setting a timer whenever it becomes enabled.
- The time allocated to the timer is determined by the rate of the activity.
- If several activities are enabled at the same time each will have its own associated timer.
- When the first timer finishes, that activity takes place—the activity is said to complete or succeed.
- This means that the activity is considered to "happen": an external observer will witness the event of activity of type α.
- An activity may be preempted, or aborted, if another one completes first.

PEPA activities and rates

- We can think of this as the activity setting a timer whenever it becomes enabled.
- The time allocated to the timer is determined by the rate of the activity.
- If several activities are enabled at the same time each will have its own associated timer.
- When the first timer finishes, that activity takes place—the activity is said to complete or succeed.
- This means that the activity is considered to "happen": an external observer will witness the event of activity of type α.
- An activity may be preempted, or aborted, if another one completes first.

PEPA and Markov processes

In a PEPA model if we define the stochastic process $X(t)$, such that $X(t) = C_i$ indicates that the system behaves as component C_i at time t, then $X(t)$ is a Markov process which can be characterised by a matrix, $\boldsymbol{Q}$.

A stationary or equilibrium probability distribution, $\pi(\cdot)$, exists for every time-homogeneous irreducible Markov process whose states are all positive-recurrent.

This distribution is found by solving the global balance equation

$$\pi \boldsymbol{Q} = \boldsymbol{0}$$

subject to the normalisation condition

$$\sum \pi(C_i) = 1.$$

PEPA and Markov processes

In a PEPA model if we define the stochastic process $X(t)$, such that $X(t) = C_i$ indicates that the system behaves as component C_i at time t, then $X(t)$ is a Markov process which can be characterised by a matrix, $\boldsymbol{Q}$.

A stationary or equilibrium probability distribution, $\pi(\cdot)$, exists for every time-homogeneous irreducible Markov process whose states are all positive-recurrent.

This distribution is found by solving the global balance equation

$$\pi \boldsymbol{Q} = \boldsymbol{0}$$

subject to the normalisation condition

$$\sum \pi(C_i) = 1.$$

PEPA and Markov processes

In a PEPA model if we define the stochastic process $X(t)$, such that $X(t) = C_i$ indicates that the system behaves as component C_i at time t, then $X(t)$ is a Markov process which can be characterised by a matrix, $\boldsymbol{Q}$.

A stationary or equilibrium probability distribution, $\pi(\cdot)$, exists for every time-homogeneous irreducible Markov process whose states are all positive-recurrent.

This distribution is found by solving the global balance equation

$$\boldsymbol{\pi Q} = \mathbf{0}$$

subject to the normalisation condition

$$\sum \pi(C_i) = 1.$$

PEPA and time

All PEPA models are time-homogeneous since all activities are time-homogeneous: the rate and type of activities enabled by a component are independent of time.

PEPA and irreducibility and positive-recurrence

The other conditions, irreducibility and positive-recurrence, are easily expressed in terms of the derivation graph of the PEPA model.

We only consider PEPA models with a finite number of states so if the model is irreducible then all states must be positive-recurrent i.e. the derivation graph is strongly connected.

In terms of the PEPA model this means that all behaviours of the system must be recurrent; in particular, for every choice, whichever path is chosen it must eventually return to the point where the choice can be made again, possibly with a different outcome.

PEPA and irreducibility and positive-recurrence

The other conditions, irreducibility and positive-recurrence, are easily expressed in terms of the derivation graph of the PEPA model.

We only consider PEPA models with a finite number of states so if the model is irreducible then all states must be positive-recurrent i.e. the derivation graph is strongly connected.

In terms of the PEPA model this means that all behaviours of the system must be recurrent; in particular, for every choice, whichever path is chosen it must eventually return to the point where the choice can be made again, possibly with a different outcome.

PEPA and irreducibility and positive-recurrence

The other conditions, irreducibility and positive-recurrence, are easily expressed in terms of the derivation graph of the PEPA model.

We only consider PEPA models with a finite number of states so if the model is irreducible then all states must be positive-recurrent i.e. the derivation graph is strongly connected.

In terms of the PEPA model this means that all behaviours of the system must be recurrent; in particular, for every choice, whichever path is chosen it must eventually return to the point where the choice can be made again, possibly with a different outcome.

Outline

Upgrading a PC LAN

Recall we wish to determine the mean waiting time for data packets at a PC connected to a local area network, operating as a token ring.

The transmission medium supports no more than one transmission at any given time. To resolve conflicts, a token is passed round the network from one node to another in round robin order.

A node can transmit, only whilst it holds the token.

Upgrading a PC LAN

Recall we wish to determine the mean waiting time for data packets at a PC connected to a local area network, operating as a token ring.

The transmission medium supports no more than one transmission at any given time. To resolve conflicts, a token is passed round the network from one node to another in round robin order.

A node can transmit, only whilst it holds the token.

Upgrading a PC LAN

Recall we wish to determine the mean waiting time for data packets at a PC connected to a local area network, operating as a token ring.

The transmission medium supports no more than one transmission at any given time. To resolve conflicts, a token is passed round the network from one node to another in round robin order.

A node can transmit, only whilst it holds the token.

Upgrading a PC LAN

There are currently four PCs (or similar devices) connected to the LAN in a small office, but the company has recently recruited two new employees, each of whom will have a PC. Our task is to find out how the delay experienced by data packets at each PC will be affected if another two PCs are added.

Modelling Assumptions

- Each PC can only store one data packet waiting for transmission at a time.
- When the token arrives either there is one packet waiting or no packet waiting.
- The average rate that a PC generates data packets is λ.
- The mean duration of a data packet transmission is d ($d = 1/\mu$), and the mean time for the token to pass from one PC to the next is m ($m = 1/\omega$).
- Transmission is gated: each PC can transmit at most one data packet per visit of the token.

Modelling Assumptions

- Each PC can only store one data packet waiting for transmission at a time.
- When the token arrives either there is one packet waiting or no packet waiting.
- The average rate that a PC generates data packets is λ.
- The mean duration of a data packet transmission is d ($d = 1/\mu$), and the mean time for the token to pass from one PC to the next is m ($m = 1/\omega$).
- Transmission is gated: each PC can transmit at most one data packet per visit of the token.

Modelling Assumptions

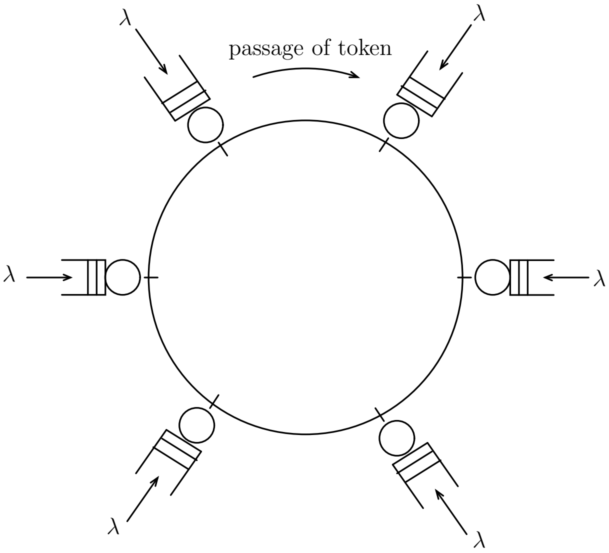
- Each PC can only store one data packet waiting for transmission at a time.
- When the token arrives either there is one packet waiting or no packet waiting.
- The average rate that a PC generates data packets is λ.
- The mean duration of a data packet transmission is d ($d = 1/\mu$), and the mean time for the token to pass from one PC to the next is m ($m = 1/\omega$).
- Transmission is gated: each PC can transmit at most one data packet per visit of the token.

Modelling Assumptions

- Each PC can only store one data packet waiting for transmission at a time.
- When the token arrives either there is one packet waiting or no packet waiting.
- The average rate that a PC generates data packets is λ.
- The mean duration of a data packet transmission is d ($d = 1/\mu$), and the mean time for the token to pass from one PC to the next is m ($m = 1/\omega$).
- Transmission is gated: each PC can transmit at most one data packet per visit of the token.

Modelling Assumptions

- Each PC can only store one data packet waiting for transmission at a time.
- When the token arrives either there is one packet waiting or no packet waiting.
- The average rate that a PC generates data packets is λ.
- The mean duration of a data packet transmission is d ($d = 1/\mu$), and the mean time for the token to pass from one PC to the next is m ($m = 1/\omega$).
- Transmission is gated: each PC can transmit at most one data packet per visit of the token.


passage of token
λ
λ
λ
λ
λ
λ

Modelling the system: choosing components

The first stage in developing a model of the system in PEPA is to determine the components of the system and the actions which they can undertake.

It seems clear that one type of component should be used to represent the PCs. The components representing the four/six PCs with have essentially the same behaviour. But since token visits the nodes in order we will need to distinguish the components.

We will need another component to represent the medium. As remarked previously, the medium can be represented solely by the token.

Modelling the system: choosing components

The first stage in developing a model of the system in PEPA is to determine the components of the system and the actions which they can undertake.

It seems clear that one type of component should be used to represent the PCs. The components representing the four/six PCs with have essentially the same behaviour. But since token visits the nodes in order we will need to distinguish the components.

We will need another component to represent the medium. As remarked previously, the medium can be represented solely by the token.

Modelling the system: choosing components

The first stage in developing a model of the system in PEPA is to determine the components of the system and the actions which they can undertake.

It seems clear that one type of component should be used to represent the PCs. The components representing the four/six PCs with have essentially the same behaviour. But since token visits the nodes in order we will need to distinguish the components.

We will need another component to represent the medium. As remarked previously, the medium can be represented solely by the token.

Modelling the system: choosing actvities

The description of the PC is very simple in this case. It only has two activities which it can undertake:

- generate a data packet;
- transmit a data packet.

Moreover we are told that it can only hold one data packet at a time and so these activities must be undertaken sequentially.

This suggests the following PEPA component for the ith PC:

$$\begin{aligned} PC_{i0} &\stackrel{def}{=} (arrive, \lambda).PC_{i1} \\ PC_{i1} &\stackrel{def}{=} (transmit_i, \mu).PC_{i0} \end{aligned}$$

This will need some refinement when we consider interaction with the token.

Modelling the system: choosing actvities

The description of the PC is very simple in this case. It only has two activities which it can undertake:

- generate a data packet;
- transmit a data packet.

Moreover we are told that it can only hold one data packet at a time and so these activities must be undertaken sequentially.

This suggests the following PEPA component for the ith PC:

$$\begin{aligned} PC_{i0} &\stackrel{\mathit{def}}{=} (arrive, \lambda).PC_{i1} \\ PC_{i1} &\stackrel{\mathit{def}}{=} (transmit_i, \mu).PC_{i0} \end{aligned}$$

This will need some refinement when we consider interaction with the token.

Modelling the system: choosing actvities

The description of the PC is very simple in this case. It only has two activities which it can undertake:

- generate a data packet;
- transmit a data packet.

Moreover we are told that it can only hold one data packet at a time and so these activities must be undertaken sequentially.

This suggests the following PEPA component for the ith PC:

$$\begin{aligned} PC_{i0} &\stackrel{\mathit{def}}{=} (\mathit{arrive}, \lambda).PC_{i1} \\ PC_{i1} &\stackrel{\mathit{def}}{=} (\mathit{transmit}_i, \mu).PC_{i0} \end{aligned}$$

This will need some refinement when we consider interaction with the token.

Modelling the system: choosing activities

For the token we can think of its current state being characterised by its current position. Thus, if there are N PCs in the network the states of the token correspond to the values $\{1, 2, \ldots N\}$.

When it is at the ith PC then the token may

- transmit a data packet if there is one to transmit and then walk on; or
- walk on at once if there is no data packet waiting.

$$Token_i \stackrel{def}{=} (walkon_{i+1}, \omega).Token_{i+1} + (transmit_i, \mu).(walk_{i+1}, \omega).Token_{i+1}$$

Modelling the system: choosing activities

For the token we can think of its current state being characterised by its current position. Thus, if there are N PCs in the network the states of the token correspond to the values $\{1, 2, \ldots N\}$.

When it is at the ith PC then the token may

- transmit a data packet if there is one to transmit and then walk on; or
- walk on at once if there is no data packet waiting.

$$\begin{aligned} Token_i \overset{def}{=} & (walkon_{i+1}, \omega).Token_{i+1} + \\ & (transmit_i, \mu).(walk_{i+1}, \omega).Token_{i+1} \end{aligned}$$

Modelling the system: choosing activities

For the token we can think of its current state being characterised by its current position. Thus, if there are N PCs in the network the states of the token correspond to the values $\{1, 2, \ldots N\}$.

When it is at the ith PC then the token may

- transmit a data packet if there is one to transmit and then walk on; or
- walk on at once if there is no data packet waiting.

$$\begin{aligned} Token_i \stackrel{\mathit{def}}{=} & (walkon_{i+1}, \omega).Token_{i+1} + \\ & \qquad (transmit_i, \mu).(walk_{i+1}, \omega).Token_{i+1} \end{aligned}$$

Refining the components

In order to ensure that the token's choice is made dependent on the state of PC being visited, we add a walkon action to the PC when it is empty, and impose a cooperation between the PC and the Token for both walkon and transmit.

$$
\begin{aligned}
PC_{i0} &\overset{\mathit{def}}{=} (arrive, \lambda).PC_{i1} + (walkon_2, \omega).PC_{i0} \\
PC_{i1} &\overset{\mathit{def}}{=} (transmit_i, \mu).PC_{i0}
\end{aligned}
$$

Refining the components

In order to ensure that the token's choice is made dependent on the state of PC being visited, we add a walkon action to the PC when it is empty, and impose a cooperation between the PC and the Token for both walkon and transmit.

$$\begin{aligned} PC_{i0} &\overset{def}{=} (arrive, \lambda).PC_{i1} + (walkon_2, \omega).PC_{i0} \\ PC_{i1} &\overset{def}{=} (transmit_i, \mu).PC_{i0} \end{aligned}$$

Complete model: four PC case

$$
\begin{aligned}
PC_{10} &\overset{def}{=} (arrive, \lambda).PC_{11} + (walkon_2, \omega).PC_{10} \\
PC_{11} &\overset{def}{=} (transmit_1, \mu).PC_{10} \\
\\
PC_{20} &\overset{def}{=} (arrive, \lambda).PC_{21} + (walkon_3, \omega).PC_{20} \\
PC_{21} &\overset{def}{=} (transmit_2, \mu).PC_{20} \\
\\
PC_{30} &\overset{def}{=} (arrive, \lambda).PC_{31} + (walkon_4, \omega).PC_{30} \\
PC_{31} &\overset{def}{=} (transmit_3, \mu).PC_{30} \\
\\
PC_{40} &\overset{def}{=} (arrive, \lambda).PC_{41} + (walkon_1, \omega).PC_{40} \\
PC_{41} &\overset{def}{=} (transmit_4, \mu).PC_{40}
\end{aligned}
$$

$$\begin{aligned}
Token_1 &\stackrel{def}{=} (walkon_2, \omega).Token_2 + (transmit_1, \mu).(walk_2, \omega).Token_2 \\
Token_2 &\stackrel{def}{=} (walkon_3, \omega).Token_3 + (transmit_2, \mu).(walk_3, \omega).Token_3 \\
Token_3 &\stackrel{def}{=} (walkon_4, \omega).Token_4 + (transmit_3, \mu).(walk_4, \omega).Token_4 \\
Token_4 &\stackrel{def}{=} (walkon_1, \omega).Token_1 + (transmit_4, \mu).(walk_1, \omega).Token_1
\end{aligned}$$

$$\begin{aligned}
LAN &\stackrel{def}{=} (PC_{10} \parallel PC_{20} \parallel PC_{30} \parallel PC_{40}) \underset{L}{\bowtie} Token_1 \\
&\quad \text{where } L = \{walkon_1, walkon_2, walkon_3, walkon_4, \\
&\qquad\qquad transmit_1, transmit_2, transmit_3, transmit_4\}.
\end{aligned}$$

Here we have arbitrarily chosen a starting state in which all the PCs are empty and the Token is at PC1.

State space size

N	2	3	4	5	6	8
$\|S\|$	16	48	128	320	768	4096

N	20	30
$\|S\|$	4.194304×10^7	6.442450×10^{10}

Current tools can analyse models up the size of approximately 10^7 states.

State space size

N	2	3	4	5	6	8
$\|S\|$	16	48	128	320	768	4096

N	20	30
$\|S\|$	4.194304×10^7	6.442450×10^{10}

Current tools can analyse models up the size of approximately 10^7 states.

State space size

N	2	3	4	5	6	8
$\|S\|$	16	48	128	320	768	4096

N	20	30
$\|S\|$	4.194304×10^7	6.442450×10^{10}

Current tools can analyse models up the size of approximately 10^7 states.

Performance results

In order to calculate performance measures from the model we must first assign values to the parameters.

The following values were assigned to parameters of the model in both cases (the unit of time is assumed to be milliseconds),

$$\lambda = 0.01 \qquad d = 10 \qquad \mu = 0.1 \qquad m = 1.0 \qquad \omega = 1.0$$

We wish to calculate the average waiting time of data packets at any PC in the network. Since all the PCs are statistically identical, we can arbitrary choose one as representative—we select PC_1.

Performance results

In order to calculate performance measures from the model we must first assign values to the parameters.

The following values were assigned to parameters of the model in both cases (the unit of time is assumed to be milliseconds),

$$\lambda = 0.01 \qquad d = 10 \qquad \mu = 0.1 \qquad m = 1.0 \qquad \omega = 1.0$$

We wish to calculate the average waiting time of data packets at any PC in the network. Since all the PCs are statistically identical, we can arbitrary choose one as representative—we select PC_1.

Derivation of performance measures

There are three different ways in which performance measures can be derived from the steady state distribution of the Markov process, corresponding to different types of measure:

- state-based measures, e.g. utilisation;
- rate-based measures, e.g. throughput;
- other measures which fall outside the above categories, e.g. response time.

State-based measures

State-based measures are those which clearly correspond to the probability that the model is in a state, or a subset of states, which satisfy some condition.

For example, utilisation will correspond to those states where a resource is in use.

Thus in order to calculate utilisation we sum the steady state probabilities of being in any of the states where the resource is in use.

State-based measures

State-based measures are those which clearly correspond to the probability that the model is in a state, or a subset of states, which satisfy some condition.

For example, utilisation will correspond to those states where a resource is in use.

Thus in order to calculate utilisation we sum the steady state probabilities of being in any of the states where the resource is in use.

State-based measures

State-based measures are those which clearly correspond to the probability that the model is in a state, or a subset of states, which satisfy some condition.

For example, utilisation will correspond to those states where a resource is in use.

Thus in order to calculate utilisation we sum the steady state probabilities of being in any of the states where the resource is in use.

Rate-based measures

Rate-based measures are those which correspond to the predicted rate at which some event occurs.

This will be the product of the rate of the event, and the probability that the event is enabled, i.e. the probability of being in one of the states from which the event can occur.

Thus to calculate the throughput of the transmission we consider the probability of being a state where tranmission can occur and multiply it by μ the transmission rate.

Rate-based measures

Rate-based measures are those which correspond to the predicted rate at which some event occurs.

This will be the product of the rate of the event, and the probability that the event is enabled, i.e. the probability of being in one of the states from which the event can occur.

Thus to calculate the throughput of the transmission we consider the probability of being a state where tranmission can occur and multiply it by μ the transmission rate.

Rate-based measures

Rate-based measures are those which correspond to the predicted rate at which some event occurs.

This will be the product of the rate of the event, and the probability that the event is enabled, i.e. the probability of being in one of the states from which the event can occur.

Thus to calculate the throughput of the transmission we consider the probability of being a state where tranmission can occur and multiply it by μ the transmission rate.

Calculating waiting time

Waiting time, which is the residence time of a data packet at the interface minus the transmission time cannot be derived directly from the steady state distribution since it is neither a state-based nor a rate-based performance measure.

However, if we know the average number of data packets at the interface, and the average throughput, we can apply Little's law to calculate residence time

$$\mathcal{R} = \frac{\mathcal{N}}{X}$$

Calculating waiting time

Waiting time, which is the residence time of a data packet at the interface minus the transmission time cannot be derived directly from the steady state distribution since it is neither a state-based nor a rate-based performance measure.

However, if we know the average number of data packets at the interface, and the average throughput, we can apply Little's law to calculate residence time

$$\mathcal{R} = \frac{\mathcal{N}}{X}$$

Calculating waiting time

Once we have the residence time the waiting time can be calculated as:

$$\mathcal{W} = \frac{\mathcal{N}}{X} - 1/\mu$$

There will be a data packet waiting in the interface whenever the first PC is occupied. So the average number of data packets, $\mathcal{N}$, is the total probability of being in one of these states.

Similarly, a data transmission can occur whenever there is a data packet waiting for transmission and the token is at PC1. So the average throughput X will be the rate at which transmission occurs, μ, multiplied by the total probability of being in one of these states.

Calculating waiting time

Once we have the residence time the waiting time can be calculated as:

$$\mathcal{W} = \frac{\mathcal{N}}{X} - 1/\mu$$

There will be a data packet waiting in the interface whenever the first PC is occupied. So the average number of data packets, $\mathcal{N}$, is the total probability of being in one of these states.

Similarly, a data transmission can occur whenever there is a data packet waiting for transmission and the token is at PC1. So the average throughput X will be the rate at which transmission occurs, μ, multiplied by the total probability of being in one of these states.

Calculating waiting time

Once we have the residence time the waiting time can be calculated as:

$$\mathcal{W} = \frac{\mathcal{N}}{X} - 1/\mu$$

There will be a data packet waiting in the interface whenever the first PC is occupied. So the average number of data packets, $\mathcal{N}$, is the total probability of being in one of these states.

Similarly, a data transmission can occur whenever there is a data packet waiting for transmission and the token is at PC1. So the average throughput X will be the rate at which transmission occurs, μ, multiplied by the total probability of being in one of these states.

Predicted waiting time

Based on these expressions and the calculated steady state distributions we find the following values:
for 4 PCs

$$\text{average waiting time, } \mathcal{W} = \frac{0.1333}{0.008666} - \frac{1}{0.1} = 15.3862 - 10 = 5.3862$$

for 6 PCs

$$\text{average waiting time, } \mathcal{W} = \frac{0.1719}{0.00828} - \frac{1}{0.1} = 20.7678 - 10 = 10.7678$$

Thus the average waiting time for data packets will almost double when two more PCs are added to the network.

Predicted waiting time

Based on these expressions and the calculated steady state distributions we find the following values:
for 4 PCs

average waiting time, $\mathcal{W} = \frac{0.1333}{0.008666} - \frac{1}{0.1} = 15.3862 - 10 = 5.3862$

for 6 PCs

average waiting time, $\mathcal{W} = \frac{0.1719}{0.00828} - \frac{1}{0.1} = 20.7678 - 10 = 10.7678$

Thus the average waiting time for data packets will almost double when two more PCs are added to the network.

Predicted waiting time

Based on these expressions and the calculated steady state distributions we find the following values:
for 4 PCs

$$\text{average waiting time, } \mathcal{W} = \frac{0.1333}{0.008666} - \frac{1}{0.1} = 15.3862 - 10 = 5.3862$$

for 6 PCs

$$\text{average waiting time, } \mathcal{W} = \frac{0.1719}{0.00828} - \frac{1}{0.1} = 20.7678 - 10 = 10.7678$$

Thus the average waiting time for data packets will almost double when two more PCs are added to the network.

References

- J.P. Buzen, *Fundamental operational laws of computer system performance*, Acta Inf. 7, 2 (1976), 167-182.
- A.Aldini, M.Bernardo and F.Corradini, *A Process Algebraic Approach to Software Architecture Design*, Springer, 2010.
- J.Hillston, *A Compositional Approach to Performance Modelling*, Cambridge University Press, 1996.