Performance Modelling — Lecture 4
More Complex Markov Processes

Jane Hillston
School of Informatics
The University of Edinburgh
Scotland

26th January 2017

Upgrading a PC LAN

Suppose 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.

Upgrading a PC LAN

Suppose 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.

Token ring communication

A node has control of the medium, i.e. it can transmit, only whilst it holds the token.

Token ring communication

A node has control of the medium, i.e. it can transmit, only whilst it holds the token.

In a PC LAN every PC corresponds to a node on the network.

Token ring communication

A node has control of the medium, i.e. it can transmit, only whilst it holds the token.

In a PC LAN every PC corresponds to a node on the network.

Other nodes on the network might be peripheral devices such as printers or faxes but for the purposes of this study we make no distinction and assume that all nodes are PCs.

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.

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, so at each visit of the token there is either one packet waiting or no packet waiting. The average rate at which each PC generates data packets for transmission is known to be λ.

Modelling Assumptions

Each PC can only store one data packet waiting for transmission at a time, so at each visit of the token there is either one packet waiting or no packet waiting. The average rate at which each PC generates data packets for transmission is known to be λ.

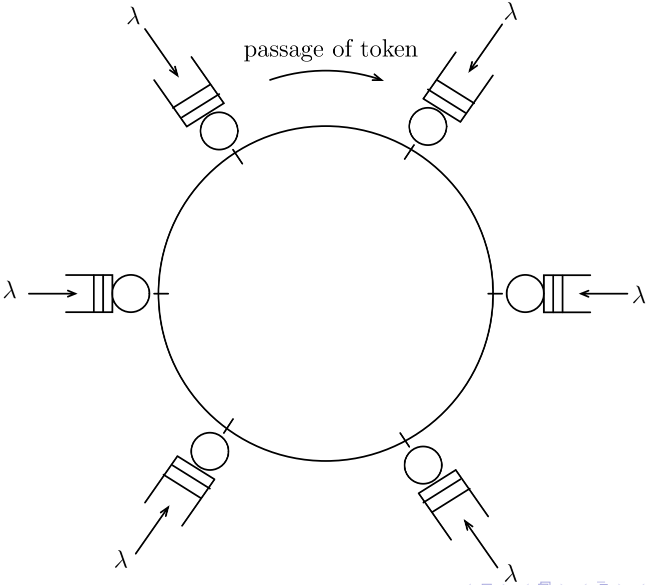
We also know the mean duration, d, of a data packet transmission, and the mean time, m, taken for the token to pass from one PC to the next.

Modelling Assumptions

Each PC can only store one data packet waiting for transmission at a time, so at each visit of the token there is either one packet waiting or no packet waiting. The average rate at which each PC generates data packets for transmission is known to be λ.

We also know the mean duration, d, of a data packet transmission, and the mean time, m, taken for the token to pass from one PC to the next.

It is assumed that if another data packet is generated, whilst the PC is transmitting, this second data packet must wait for the next visit of the token before it can be transmitted. In other words, each PC can transmit at most one data packet per visit of the token.


λ
λ
passage of token
λ
λ
λ
λ

Modelling the system: capturing state

The first stage in developing a model of the system is to determine the random variables which will characterise the state of the system and the events which will move the model between states.

Modelling the system: capturing state

The first stage in developing a model of the system is to determine the random variables which will characterise the state of the system and the events which will move the model between states.

It seems clear that one aspect of state corresponds with the PCs. We are told that each PC can only hold one data packet at a time which limits the possible states of a PC.

Modelling the system: capturing state

The first stage in developing a model of the system is to determine the random variables which will characterise the state of the system and the events which will move the model between states.

It seems clear that one aspect of state corresponds with the PCs. We are told that each PC can only hold one data packet at a time which limits the possible states of a PC.

Since we are not told that they differ in any way we may assume that the four/six PCs have essentially the same behaviour.

Modelling the system: capturing state

The first stage in developing a model of the system is to determine the random variables which will characterise the state of the system and the events which will move the model between states.

It seems clear that one aspect of state corresponds with the PCs. We are told that each PC can only hold one data packet at a time which limits the possible states of a PC.

Since we are not told that they differ in any way we may assume that the four/six PCs have essentially the same behaviour.

However since the order in which the token visits nodes is important we do need to distinguish the PCs.

Modelling the system: capturing state

We also need to represent the medium in some way.

Modelling the system: capturing state

We also need to represent the medium in some way.

In fact for the token ring it is only necessary to represent the location of the token as this is sufficient to determine whether a PC can transmit or not.

Modelling the system: capturing state

$$\text{Each PC} \longrightarrow \{0, 1\}$$

$$\text{Token} \longrightarrow \{1, 2, \ldots, N\}$$

Modelling the system: capturing state

$$\text{Each PC} \longrightarrow \{0,1\}$$

$$\text{Token} \longrightarrow \{1,2,\ldots,N\}$$

$$(\underbrace{PC_1}_{\{0,1\}}, \underbrace{PC_2}_{\{0,1\}}, \ldots, \underbrace{PC_N}_{\{0,1\}}, \underbrace{Token}_{\{1,2,\ldots,N\}})$$

Modelling the system: capturing events

The description of the PC behaviour is very simple in this case. It only has two events which can occur:

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

Modelling the system: capturing events

The description of the PC behaviour is very simple in this case. It only has two events which can occur:

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

Since a PC can only hold one data packet at a time, these events must be undertaken sequentially.

Modelling the system: capturing events

The description of the PC behaviour is very simple in this case. It only has two events which can occur:

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

Since a PC can only hold one data packet at a time, these events must be undertaken sequentially.

However we will see that we need to be careful and this view will need some refinement.

Modelling the system: capturing events

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

Modelling the system: capturing events

For the token we want its current state to be 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 events for the token are

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

Modelling the system: capturing events

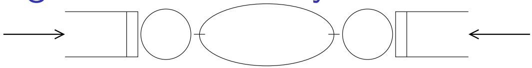
For the token we want its current state to be characterised by its current position. Thus, if there are N PCs in the network the states of the token correspond to the values $\{1, 2, \ldots N\}$.

When it is at the ith PC then events for the token are

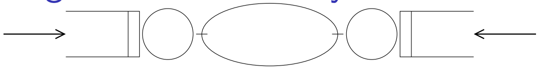
- transmit a packet if there is one to transmit and then walk on; or
- walk on at once if there is no token waiting;

These are two distinct types of event.

Validate against a minimal system

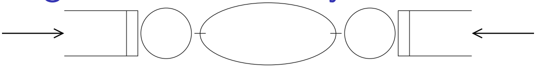


Validate against a minimal system



(1,1,1) both PC1 and PC2 ready to transmit; token at PC1

Validate against a minimal system

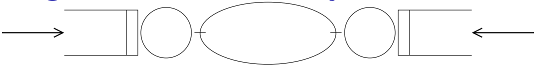


(1,1,1) both PC1 and PC2 ready to transmit; token at PC1

↓ PC1 transmits

(0,1,1) PC2 ready to transmit; token at PC1

Validate against a minimal system



(1,1,1) both PC1 and PC2 ready to transmit; token at PC1

↓ PC1 transmits

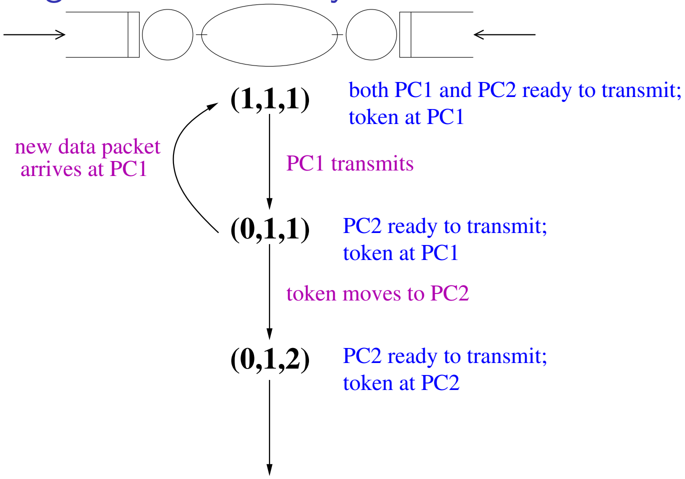
(0,1,1) PC2 ready to transmit; token at PC1

↓ token moves to PC2

(0,1,2) PC2 ready to transmit; token at PC2

↓

Validate against a minimal system



Refining the components

Using the current definition the token will decide between transmitting a packet and walking on probabilistically (i.e. through the race condition).

Refining the components

Using the current definition the token will decide between transmitting a packet and walking on probabilistically (i.e. through the race condition).

In order to ensure that the choice is made dependent on the state of the current PC, we add another state to the token for each PC recording the difference between the token being there before a packet has been transmitted and being there afterwards.

Refining the components

Using the current definition the token will decide between transmitting a packet and walking on probabilistically (i.e. through the race condition).

In order to ensure that the choice is made dependent on the state of the current PC, we add another state to the token for each PC recording the difference between the token being there before a packet has been transmitted and being there afterwards.

This will enforce the gated service, i.e. that at most one packet is transmitted per visit of the token.

A refined notion of state

$$\text{Each PC} \longrightarrow \{0, 1\}$$

$$\text{Token} \longrightarrow \{1, 2, \ldots, N, 1^+, 2^+, \ldots, N^+\}$$

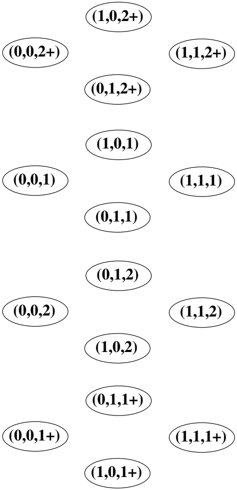
A refined notion of state

$$\text{Each PC} \quad \longrightarrow \quad \{0, 1\}$$

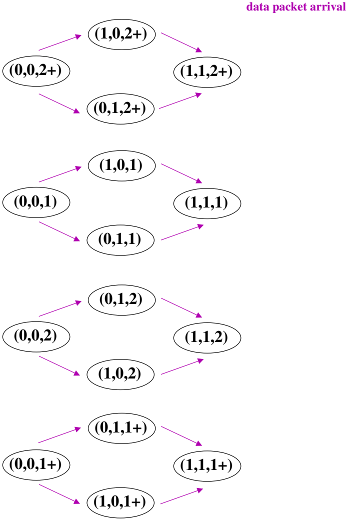
$$\text{Token} \quad \longrightarrow \quad \{1, 2, \ldots, N, 1^+, 2^+, \ldots, N^+\}$$

$$(\underbrace{PC_1}_{\{0,1\}}, \underbrace{PC_2}_{\{0,1\}}, \ldots, \underbrace{PC_N}_{\{0,1\}}, \underbrace{Token}_{\{1,\ldots,N,1^+,\ldots,N^+\}})$$

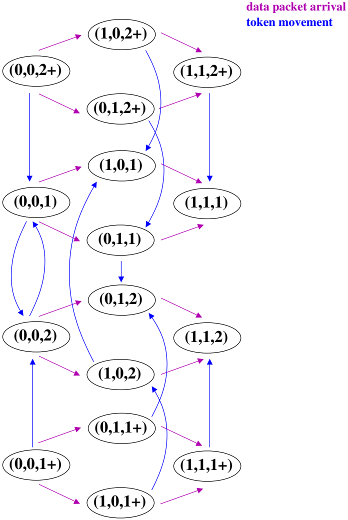
State space for minimal model



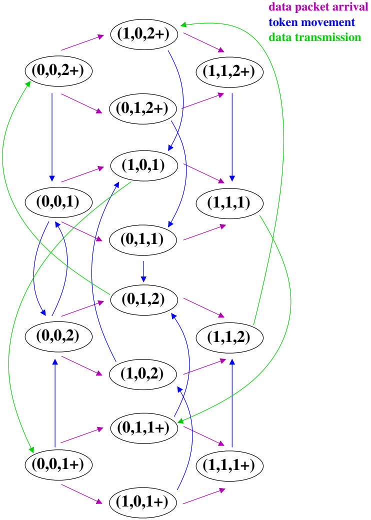
State space for minimal model



State space for minimal model



State space for minimal model



State space size

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

State space size

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

N	10	20	30
$\|S\|$	2048	4.194304×10^7	6.442450×10^{10}

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),

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

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

As discussed in the previous lecture 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.

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.

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.

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.

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

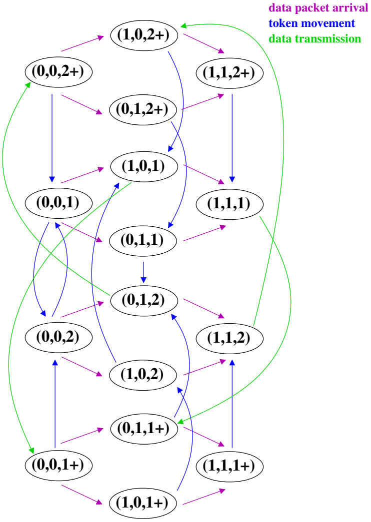
Waiting time, which is the residence time of a data packet at the interface minus the transmission time cannot be derived directly from the steady state distribution since it is neither a state-based nor a rate-based performance measure.

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

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


data packet arrival
token movement
data transmission
(1,0,2+)
(0,0,2+)
(1,1,2+)
(0,1,2+)
(1,0,1)
(0,0,1)
(1,1,1)
(0,1,1)
(0,1,2)
(0,0,2)
(1,1,2)
(1,0,2)
(0,1,1+)
(0,0,1+)
(1,1,1+)
(1,0,1+)

Calculating waiting time

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

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

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.

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 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

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

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

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.

Difficulties of working with Markov processes

It soon becomes time-consuming and error-prone to work directly at the level of state transitions diagrams or infinitesimal generator matrices, if not entirely infeasible.

Difficulties of working with Markov processes

It soon becomes time-consuming and error-prone to work directly at the level of state transitions diagrams or infinitesimal generator matrices, if not entirely infeasible.

The solution is to use a high-level modelling language to describe the behaviour we are interested in more closely to the actual observed behaviour of the system, rather than at the level of states and transitions.

High-level modelling formalisms

Several such high-level modelling languages for performance modelling based on Markov processes have been developed over the last decades.

High-level modelling formalisms

Several such high-level modelling languages for performance modelling based on Markov processes have been developed over the last decades.

We will consider three:

1. Queues and queueing networks
2. Stochastic Petri nets
3. The stochastic process algebra, PEPA.

High-level modelling formalisms

Several such high-level modelling languages for performance modelling based on Markov processes have been developed over the last decades.

We will consider three:

1. Queues and queueing networks
2. Stochastic Petri nets
3. The stochastic process algebra, PEPA.

Each can be used to automatically generate a Markov process and derive performance measures from a high-level description, as we will see.