University of London

Masters Programmes in Telecommunications

Network Planning and Performance

Question Paper

Solution Q2

Solution Q3

Solution Q4

UNIVERSITY OF LONDON

Masters Programmes in Telecommunications

Network Planning and Performance 1998/99

Assignment
Issued 28th April 1999

Notes:

* The paper comprises 5 questions

* The marks allocated to parts of questions are indicated for guidance

- Candidates are required to answer 3 (three) questions

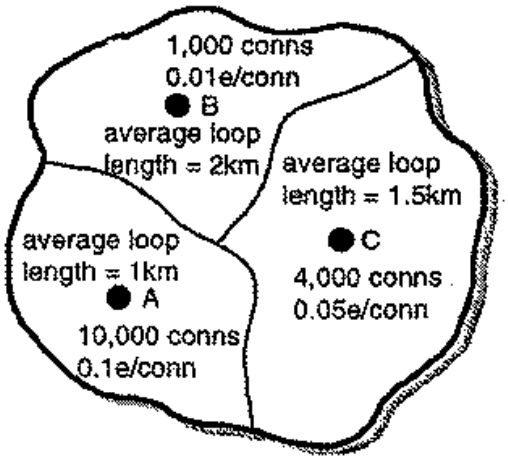
* Assignment to be handed in by 10th June 1999

IGDP Telecommunications for Industry'
MRes in Telecommunications
MSc in Telecommunications

Network Planning and Performance Assignment 1998/99

Question One

Q1[a] The area shown in the figure has a forecast customer distribution grouped around three potential local exchange locations, namely A which is in a predominantly business district, B which is mostly residential and C which is a mix of the two.



Determine the optimum number of exchanges for serving the area at minimum cost, given that:

* Loop cost = £1000/pair km [includes duct] [note - 1 pair/connection]
* Junction cost = £10,000/route + £1000/circuit km [includes cable & transmission]
* Junction duct = £10,000/km
* Building cost = £lm+ £100/connection [includes site]
* Switching cost = £lm + £50/conn + £100/e
* Circuit efficiency = 0.5e/circuit
* Traffic distribution = 20% own exchange, outgoing in proportion to distant end conns

The distance between the potential exchange locations is 3km; hence, extending pairs from one location to another adds 3km to their average length.

marks - 25%

Ql[b] An Internet service has been provided by an ISP [Internet Service Provider] located in area B. It is predicted that 50% of customers will take up the service and 50% of these will make calls to the ISP of one hour duration between 7 pm and 8 pm on Sunday evenings. What is the effect on the optimum network derived in part [a] of the question.

marks - 25%

Ql[c] Discuss the results of part [b] of the question and develop a justified, by reasoned argument, strategy for the network operator to cater with the Internet demand in the most economic manner.

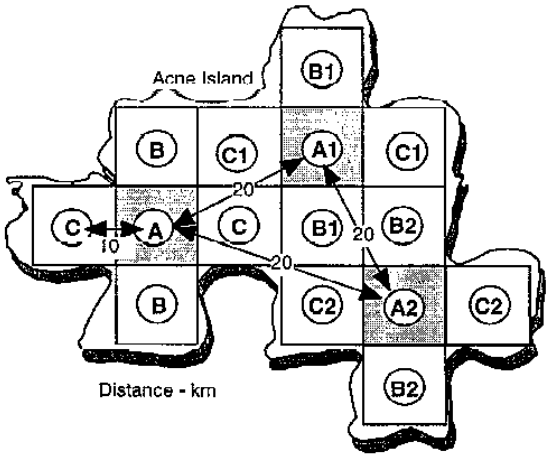
marks - 50%

Question Two

Q2[a] A Telco has decided to compete overseas in the country of Acne. It will initially offer a trunk by pass service for cheap trunk calls, gaining access to the customers via interconnect to the incumbent's, network [shown in the figure]. Each of the incumbent's exchanges serves 10,000 customers but the A exchanges also serve as fully interconnected trunk exchanges, and traffic is evenly distributed between all exchanges. The competing Telco is forecasting a market share of 10%. You are required to design a cost optimum network given that:

* The tree options are: interconnecting at exchange Al, interconnecting at exchanges A, Al and A2, interconnecting at all exchanges.
* The busy hour calling rate for trunk calls is 0.Ole per customer with a call holding time of 3 minutes.
* Exchange capital cost = £O.5m + £100 per switched erlang.
* Exchange accounting life = 10 years.
* Exchange operating cost = £0.03m per exchange
* Circuit efficiency = 0.5e/circuit
* Transmission capital cost [radio] = £0.01m per route + £1,000'per circuit km
* Transmission accounting life = 15 years
* Transmission maintenance cost = £0.02m per route
* Day: busy hour ratio = 10:1 [assume all days in the year are the same]
* Interconnect charges [for ingress and egress] for one point of interconnect = £0.05 per call minute
* Interconnect charges for points of interconnect at three trunk exchanges = £0.02 per call minute
* Interconnect charges for points of interconnect at all exchanges = £0.01 per call minute

marks - 50



Q2[b] The competing Telco is complaining to the regulator that the interconnect charges levied by the incumbent are too high because they are based on "fully allocated" costs. The Telco is proposing that the interconnect charges should he based on "long run incremental costs [LRIC]".

Explain the difference between the two costs and the reasoning behind the requirement for charges bused un LRIC.

marks - 20%

How would you determine LRIC for the incumbents network.

marks - 30%

Question Three

Variable bit-rate (VBR) traffic is to be multiplexed onto a 155.52 Mbit/s link through a buffer with 10 waiting spaces. Each VBR call is described by peak and mean cell rates (20000 cell/s and 2000 cell/s respectively), the performance requirements are a cell loss probability of le-8, and a probability that the cell delay exceeds 5 cell slots of le-5.
Give specific details of the CAC algorithm(s) required to ensure that the performance requirements are met. Explain your approach, giving reasons for any assumptions and simplifications, and assess their likely impact.
Implement your algorithm(s) (e.g. in a spreadsheet, or using a Maths tool, or as a program) and calculate how many connections can be admitted such that the performance requirements are met. With the number of connections you have calculated, find the actual values of cell loss and cell delay performance.

Question Four

Priority control is an important traffic control mechanism, both in ATM networks and, increasingly, in IF networks incorporating some sort of quality of service differentiation. Investigate the behaviour of a simulation model of a time-priority queueing system, which has five Bernoulli sources, and one Poisson source, as input. The Bernoulli sources feed ATM cells into a high priority buffer, and the Poisson source feeds ATM cells into a low priority buffer. If there are any cells in the high priority buffer, the server serves a high priority cell. The server only serves a low priority cell if the high priority buffer is empty.
Build a simulation model for this system (e.g. in a spreadsheet, as a program, or using a maths or simulation package).
Run your model for the following scenarios:

* each Bernoulli source has a mean load of p = 0.15 cells/slot, and the Poisson source has a load of λ = 0.15 cells/slot
* each Bernoulli source has a mean load of p = 0.09 cells/slot, and the Poisson source has a load of λ = 0.45 cells/slot
* each Bernoulli source has a mean load of p = 0.02 cells/slot, and the Poisson source has a load of λ = 0.8 cells/slot

In each case simulate the system for at least 1000 cell slots, but preferably 1 or 2 orders of magnitude more.

(a) Briefly describe how you modelled the sources and the priority queueing system, and how you processed the results.

(b) For each of the three scenarios, plot the queue state probabilities for the high and low priority buffers separately, and for the queueing system as a whole. Show your results both as tables of values, and as graphs.

(c) Comment on the results you obtain for the different scenarios - which results are similar, which are different, and why? Discuss the types of traffic which would benefit from this priority mechanism, and derive general planning rules for partitioning the traffic among priority levels, and for partitioning buffer space between the priority buffers.

Question Five

BT and Ford motors have entered into a joint venture to introduce a Web based electronic car customisation and purchasing service using the Internet. BT will undertake end-to-end service management for the service.

a] Construct enterprise models for the service to illustrate the processing, storage and messaging components necessary for the service.

30 marks

bI Determine the Quality of Service parameters necessary to give good customer satisfaction and speculate on their likely values.

30 Marks

c] Discuss the functionality necessary to ensure that these QOS parameter values are achieved.

40 marks

Andrew Wilkinson

MSc Telecommunications

UCL

University of London

NPP ASSIGNMENT 1999

Questions 2 and 3

Network Planning and Performance, 1999

Question 2

Design of a Cost Optimal Trunk By-Pass Network

and

An Examination of LRIC

1. Introduction

In order to design a cost optimal network it is necessary to determine the expected busy hour traffic demand for each network configuration. An important simplification can be made due to the even distribution of traffic between all exchanges. If we assume traffic terminates in the same ratio as it originates (i.e. at every exchange there is equal originating and terminating traffic) then the traffic carried on routes in opposite directions between the same two exchanges will be equal.

Since traffic is distributed equally between all exchanges, customer demand in terms of originating and terminating traffic density will also be the same for all exchanges. From the given average call holding time, day to busy hour ratio, busy hour calling rate and expected customer penetration. busy hour demand statistics can be calculated.

To aid calculations, Microsoft's Excel was used throughout. For each possible network configuration, separate work sheets were produced from a standard template. In each of the work sheets shown, entries are colour coded as follows:

- Blue entries are for data which doesn't change with different network topologies (e.g. call duration)
- Red entries are for topology dependent data (e.g. interconnection charges)
- Black entries are automatically calculated from blue and red data (e.g. exchange depreciation)

2. Customer Demand Calculations

To determine important customer demand statistics the following formulae were used:

$$\text{Busy hour occupancy} = 60 * \text{Busy hour calling rate}$$

$$\text{Busy hour calls} = \text{Busy hour occupancy} / \text{Call duration}$$

$$\text{Call duration per day} = \text{Busy hour ocupancy} * \text{Day : busy hour ratio}$$

$$\text{Calls per day} = \text{Busy hour calls} * \text{Day : busy hour ratio}$$

The total busy hour switching demand is calculated from:

$$\text{Busy hour switched traffic} = \text{Busy hour calling rate} * \text{Customer base}$$

On a daily and yearly basis, call demand is as follows:

$$\text{Total calls per day} = \text{Customer base} * \text{Calls per day per customer}$$

$$\text{Total call duration per year} = 365 * \text{Total calls per day per customer} * \text{Call duration}$$

$$\text{Yearly interconnection charges} = \text{Total call duration per year} * \text{Interconnection charge}$$

3. The Competitor's Traffic Routes

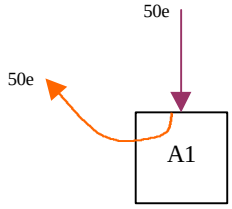
The competitor's traffic routes have been separated into two types

- Junction routes (routes between exchange types A and exchanges types B and C)
- Trunk routes (routes between A exchanges)

This assumption is only significant for a network with interconnection at all the incumbent's exchanges where I have assumed that the competitor's network won't be a fully meshed network as transmission costs would be astronomic. Further details of the individual network topologies are given with each interconnection scenario design.

4.1. Interconnection at Exchange A Only

For interconnection at only the single DMSU exchange, A1, the traffic pattern is shown below:

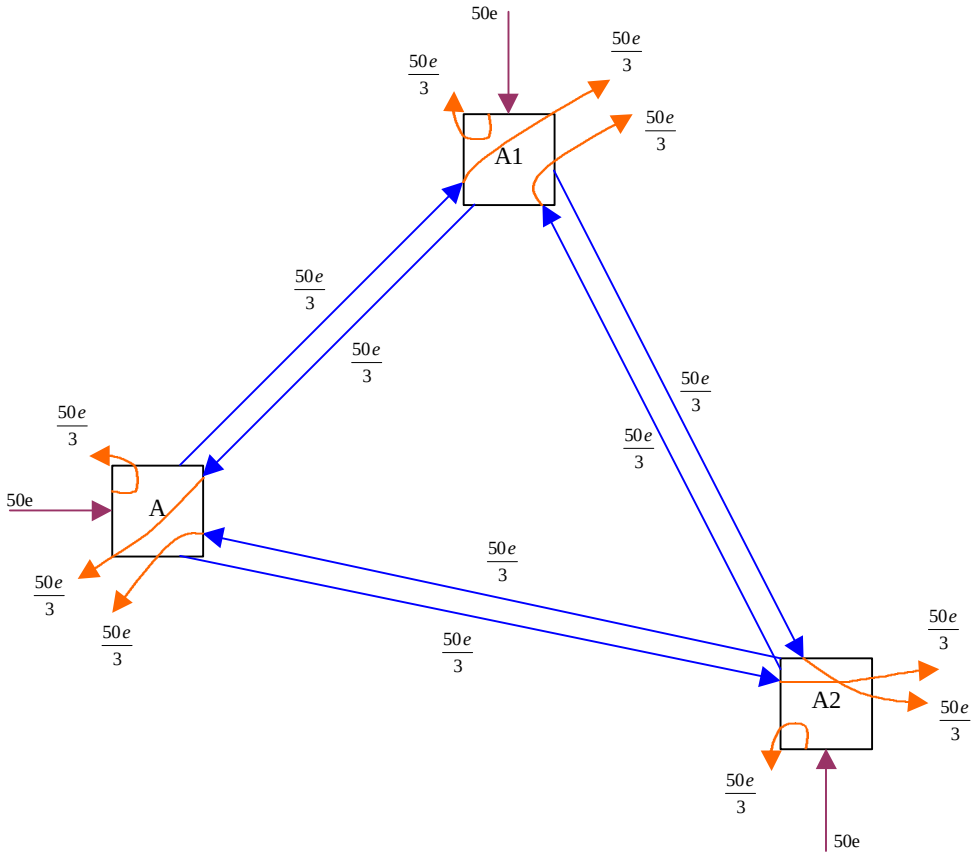


This scenario is the simplest to design for. The entire network is dominated by switching costs as no transmission is carried out. Hence, there are no trunk routes (inter 'A' exchange routes) and no junction routes.

From the data shown on the relevant spreadsheet, it can be seen that the network costs are dominated by interconnection charges that form 95% of the overall annual cost of £1.724m.

4.2. Interconnection at Exchanges A, A1 and A2

In this scenario, the competitor's three interconnecting exchanges can be considered as fully meshed DMSUs. Each DMSU exchange serves five out of the fifteen regions and hence switches one third of the competitor's total switched traffic. Since traffic originates and terminates equally in all fifteen regions, each outgoing trunk from a DMSU will carry one third of the traffic originating from within its own service area to a neighbouring DMSU. Hence, a DMSU's outgoing trunks will each carry one ninth of the competitor's total traffic (150e/9 = 50/3e). Two trunk routes enter each DMSU so a total of two ninths of the operator's total switched traffic enters a DMSU on trunk routes. The network is shown below:



In this scenario, there are six traffic routes interconnecting the three exchanges, each 20 km long. Since the transmission efficiency is 0.5, twice as many circuits as traffic carried in Erlangs are required. All six trunk routes have the same characteristics:

Trunk route length:	20 km
Traffic carried:	50/3 e
Number of circuits per route:	100/3 circuits

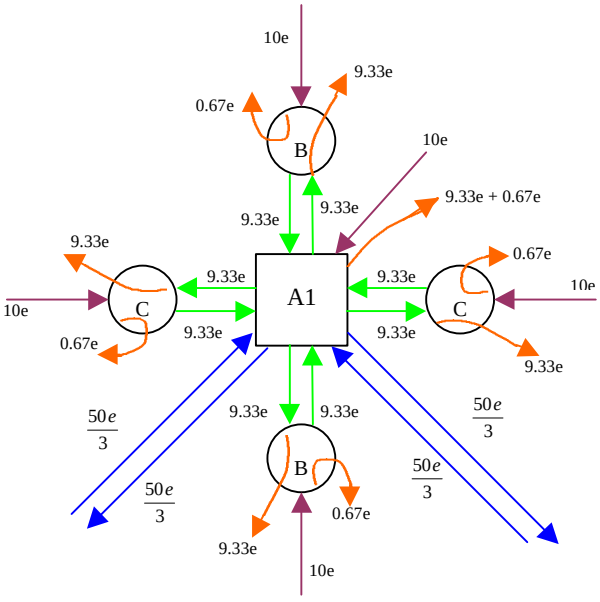
From the data shown on the second spreadsheet, it can be seen that the network costs are evenly distributed between interconnection charges and network costs which form 51% and 49% of the overall annual network cost of £1.289m respectively.

4.3. Interconnection at All Exchanges (A, A1 and A2 DMSUs)

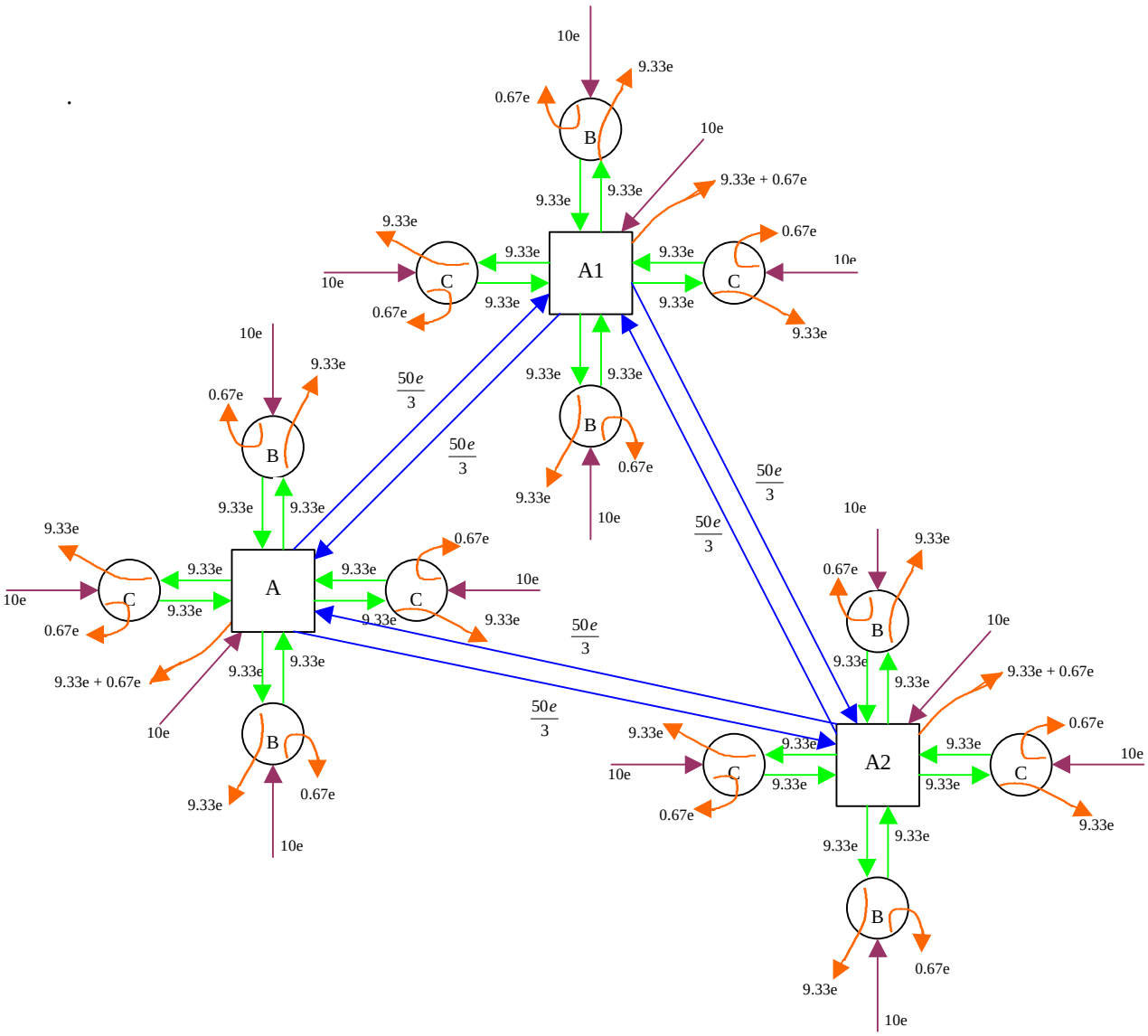
A fully meshed network with 15 exchanges would require 105 traffic routes! Rather than considering a fully meshed network, I have designed a hierarchical topology similar to the incumbents, with A, A1 and A2 being fully meshed DMSUs each with four satellite LEXs (Bs and Cs). Each DMSU also handles originating and terminating traffic within its own catchment area.

With interconnection at all 15 switches, A, A1 and A2 configured as DMSU exchnages, the trunk traffic is the same as for the previous scenario.

Each of the competitor's 12 LEXs serve one fifteenth of their total customer base, with the three DMSUs also serving one fifteenth each, thus LEXs and DMSUs will each originate 10 Erlangs of local traffic. One fifteenth of this traffic terminates locally (10/15e = 0.67e), with the rest conveyed to the competitor's nearest DMSU (9/15e = 9.33e). The traffic pattern for one of the competitor's DMSUs and satellite LEXs is shown below:



All three DMSUs serve the same size population and have the same number of satellite LEX exchanges, therefore their traffic patterns are all the same. The complete network is shown overleaf.



In addition to the six trunk routes, there are 4 junction traffic routes both to and from each of the 3 DMSUs, hence there are a total of 24 junction routes. Each junction route is 10 km long and has the same transmission efficiency as a trunk route. Hence, the 24 junction routes have the following characteristics:

Trunk route length:	10 km
Traffic carried:	50/6 e
Number of circuits per route:	50/3 circuits

With fifteen switches, the competitor has the best of neither worlds: they have both high switching and conveyance costs. The massive increase in both transmission and switching isn't justified by the reduction in interconnection charges and hence this solution would be the least profitable to implement at £2.715m, as shown in the last speadsheet.

5. Conclusions

The lowest cost network would be with three fully meshed exchanges interconnecting at A, A1 and A2. With three exchanges, switching and conveyance costs are almost equal. If the competitor interconnects at all exchanges their switching and conveyance costs rocket. At the other extreme, with only one exchange, the interconnection charges are prohibitive.

Interconnection at A Only

Number of Exchanges	1
Day:busy hour ratio (D:1)	10
Circuit Efficiency	1
Interconnect Charge	£0.05
Exchange Accounting Life	10
Transmission Accounting Life	15
Customer base	15,000

Calls per Day	2
Call Duration (mins)	3
Total Duration (mins)	6
% Busy Hour Calls	10
Busy Hour Occupancy (mins)	0.6
Busy Hour Calls	0.2
Busy Hour Calling rate (e)	0.01
Busy Hour Switched Traffic (e)	150
Calls per Day	30000
Call Duration per Year (mins)	32850000

Capital Costs

Exchanges	
Per Exchange	£500,000
Per switched Erlang	£100
Transmission	
Per Route	£10,000
Per Circuit km	£1,000

Operating Costs

Exchange	
Per Exchange	£30,000

Maintenance Costs

Transmission	
Per Route	£20,000

Traffic Routes

Inter 'A' Exchange Routes	0
Length (km each)	20
Total (km)	0
Inter 'A' Traffic per Route	0
Inter LEX Routes	0
Length (km each)	10
Total (km)	0
Inter LEX Traffic per Route	0

Depreciation

Exchanges (per building)	£50,000
Exchanges (per switched e)	£1,500
Transmission (per route)	£0
Transmission DMSU (per circuit km)	£0
Transmission LEX (per circuit km)	£0

Operating Costs

Exchanges	£30,000

Maintanace

Transmission Routes (DMSU)	£0
Transmission Routes (LEX)	£0

Inter-Connection

Charges for Ingress/Egress	£1,642,500

Total Capital Costs

Exchanges	£500,000
Exchanges (switched e)	£15,000
Transmission (all routes)	£0
Transmission DMSU (circuits)	£0
Transmission LEX (circuits)	£0

OVERALL COST PER YEAR **£1,724,000**

Entry Key:
Red - Scenario Dependent Data
Blue - Constant Data
Black - Fields Calculated From Red and Blue

Interconnection at A, A1 and A2

Number of Exchanges	3
Day:busy hour ratio (D:1)	10
Circuit Efficiency	1
Interconnect Charge	£0.02
Exchange Accounting Life	10
Transmission Accounting Life	15
Customer base	15,000

Calls per Day	2
Call Duration (mins)	3
Total Duration (mins)	6
% Busy Hour Calls	10
Busy Hour Occupancy (mins)	0.6
Busy Hour Calls	0.2
Busy Hour Calling rate (e)	0.01
Busy Hour Switched Traffic (e)	150
Calls per Day	30000
Call Duration per Year (mins)	32850000

Capital Costs

Exchanges	
Per Exchange	£500,000
Per switched Erlang	£100
Transmission	
Per Route	£10,000
Per Circuit km	£1,000

Operating Costs

Exchange	
Per Exchange	£30,000

Maintenance Costs

Transmission	
Per Route	£20,000

Traffic Routes

Inter 'A' Exchange Routes	6
Length (km each)	20
Total (km)	120
Inter 'A' Traffic per Route	16.66666667
Inter LEX Routes	0
Length (km each)	10
Total (km)	0
Inter LEX Traffic per Route	0

Depreciation

Exchanges (per building)	£150,000
Exchanges (per switched e)	£1,500
Transmission (per route)	£4,000
Transmission DMSU (per circuit km)	£266,667
Transmission LEX (per circuit km)	£0

Operating Costs

Exchanges	£90,000

Maintanace

Transmission Routes (DMSU)	£120,000
Transmission Routes (LEX)	£0

Inter-Connection

Charges for Ingress/Egress	£657,000

Total Capital Costs

Exchanges	£1,500,000
Exchanges (switched e)	£15,000
Transmission (all routes)	£60,000
Transmission DMSU (circuits)	£4,000,000
Transmission LEX (circuits)	£0

OVERALL COST PER YEAR **£1,289,167**

Entry Key:
Red - Scenario Dependent Data
Blue - Constant Data
Black - Fields Calculated From Red and Blue

Interconnection at All Exchanges (DMSUs: A, A1 and A2)

Number of Exchanges	15
Day:busy hour ratio (D:1)	10
Circuit Efficiency	1
Interconnect Charge	£0.01
Exchange Accounting Life	10
Transmission Accounting Life	15
Customer base	15,000

Calls per Day	2
Call Duration (mins)	3
Total Duration (mins)	6
% Busy Hour Calls	10
Busy Hour Occupancy (mins)	0.6
Busy Hour Calls	0.2
Busy Hour Calling rate (e)	0.01
Busy Hour Switched Traffic (e)	150
Calls per Day	30000
Call Duration per Year (mins)	32850000

Capital Costs

Exchanges	
Per Exchange	£500,000
Per switched Erlang	£100
Transmission	
Per Route	£10,000
Per Circuit km	£1,000

Operating Costs

Exchange	
Per Exchange	£30,000

Maintenance Costs

Transmission	
Per Route	£20,000

Traffic Routes

Inter 'A' Exchange Routes	6
Length (km each)	20
Total (km)	120
Inter 'A' Traffic per Route	16.66666667
Inter LEX Routes	24
Length (km each)	10
Total (km)	240
Inter LEX Traffic per Route	9.333333333

Depreciation

Exchanges (per building)	£750,000
Exchanges (per switched e)	£1,500
Transmission (per route)	£20,000
Transmission DMSU (per circuit km)	£266,667
Transmission LEX (per circuit km)	£298,667

Operating Costs

Exchanges	£450,000

Maintanace

Transmission Routes (DMSU)	£120,000
Transmission Routes (LEX)	£480,000

Inter-Connection

Charges for Ingress/Egress	£328,500

Total Capital Costs

Exchanges	£7,500,000
Exchanges (switched e)	£15,000
Transmission (all routes)	£300,000
Transmission DMSU (circuits)	£4,000,000
Transmission LEX (circuits)	£4,480,000

OVERALL COST PER YEAR **£2,715,333**

Entry Key:
Red - Scenario Dependent Data
Blue - Constant Data
Black - Fields Calculated From Red and Blue

6. FAC and LRIC

The provision of cheap trunk by-pass services by a new Telco relies on an incumbent to provide access, switching and possibly also conveyance to the Telco's points of interconnection. Traditionally the cost of interconnection to an incumbent's network has been calculated using Fully Allocated Costs (FAC) and the principles of Historic Cost Accounting (HAC). In contrast, Long Run Incremental Costs are determined on the forward-looking principles of Current Cost Accounting (CCA).

FACs are calculated as the entire cost to the incumbent to provide a service as if it were provided alone and without considering the economies of scale that a large incumbent would enjoy when the extra traffic provision is added to existing traffic. Additionally, FACs include costs such as those incurred due to inefficient operation with legacy equipment and large overheads due to items such as generous payouts to executives. If HCA is used to assess the value of a service provided by a given asset, the cost of providing the service when the asset was originally deployed is considered (plus inflation), not the cost of providing the same service with a MEA.

LRICs are different to FACs in two important ways. In its purest form, an interconnection charge based on LRIC would be calculated as the cost to the incumbent of providing the extra service (the increment) requested by the competitor in addition to what they are already providing. The principles of CCA state that the cost of providing a service should be calculated on the cost of providing it with the latest MEA, not through providing it with inefficient legacy equipment.

Modern telecommunication networks rely heavily on extensive micro-processing power. An example of the difference in costs that result when a service is assessed that uses a MEA instead of legacy equipment is that of switching. Consider the cost of switching 'X' Erlangs of traffic which requires 'K' MIPS of processing power at a cost of £ 'C' when the switch was first commissioned by the incumbent in 1993. Since1993, micro-processor power has doubled approximately every two years for the same cost, hence to buy a microprocessor to switch the same traffic in 1999, with a new switch (the MEA), it would cost £ C/8, one eighth of that in 1993. HCA would assess the switching interconnection charge on the 1993 asset cost, whereas CCA would assess it on the far cheaper equivalent asset cost of 1999. Other examples include the use of fibre instead of copper and SDH instead of PDH in the trunk network, especially if the competitor's points of interconnection require some trunk conveyance by the incumbent. It is reasonable to assume that a new entrant wouldn't install legacy equipment and hence should only be charged the cost they would incur themselves to provide the increment in traffic in the most efficient manner.

In the telecommunications market however, things are complicated by the fact that the competitor is effectively purchasing bearer services from the incumbent, which is not what they sell onto their subscribers. Many telecommunication services offered by Telcos include a large value added component, for example the 0898 private services offered by the company Telecomm's Utopia Group, 'TUG!'. These services are obtained via the incumbent's access network for which they receive a relatively small fee compared to that received by 'TUG!' from the sad users of their service. The incumbent can argue that in providing the basic service to 'TUG!' that they've lost revenue by not being able to also sell the value added components. This gives rise to LRIC 'plus', where some compensation is included in the interconnection charge to account for this.

Judging by the looks of ACNE's Posts and Telecommunication minister, Sir Edwin Nerd, who looks to be a prime user of the services provided by Telcos such as 'TUG!', ACNE's government will agree with the policy of LRIC 'plus'. Sir Nerd of course always has his voters' 'satisfaction' on his mind !

7. Determination of LRIC

There are basically two models used to calculate LRIC, namely the top-down and the bottom-up models. Top-down models are generally favoured by incumbents and considerably more complex than bottom-up models which tend to be preferred by regulators. As an example, in the UK the incumbent, BT, favours calculating LRIC interconnection charges using a top-down model, whereas Oftel, the regulator, considers the bottom-up model more transparent and easier to use.

7.1. The Top-Down Model

To calculate the LRIC of providing a service using the top-down model, the full management accounts of the incumbent are considered and whittled away by removing any costs which are non-incremental. Generally the management accounts are split into a large number of categories such as direct conveyance costs and indirect personnel costs. A big incumbent may identify many hundreds of separate categories. The total incremental cost is found by summing the cost incurred over all categories. Finally, since the incumbent's existing accounts will have been prepared using HCA, the total incremental cost is multiplied by an adjustment factor between 0 and 1 to convert the cost to that which would have been found using CCA.

An asset's depreciation in the top-down model is based on traditional linear depreciation from installation cost over an assets accounting lifetime.

7.2. The Bottom –Up Model

The bottom-up model comprises three parts:

1. The incremental costs of conveyance and switching, determined from the incumbent's Network model.
2. The annual replacement costs of plant such as switches and transmission links.
3. The incremental costs of providing local access.

Depreciation in the bottom-up model follows a non-linear relationship between an asset's age and its rate of depreciation. Typically, an exponential depreciation profile is appropriate, in which the value of an asset drops faster the older it gets.

7.3. Determination of LRIC for ACNE's Incumbent Based on the Bottom-Up Approach

Interconnection charges in ACNE are likely to be calculated using the bottom-up model in sympathy to the preferences of most telecommunication regulators.

The incumbent's assets used in providing the interconnect service vary depending on the competitor's network topology. For ACNE's incumbent it will include all or some of the following:

Asset Group	**Asset**	**Relevant With**
Exchanges (switching)	LEX Switches (B and C switches)	All scenarios
	DMSU Switches (A, A1 and A2)	First and Second scenario only
Transmission (conveyance)	Subscriber to LEX (Access)	All scenarios
	LEX to DMSU (Junction)	First and Second scenario only
	Inter DMSU (Trunk)	First scenario only

7.3.1. Incremental Costs of Conveyance, Switching and Access

The incremental costs for assets used in the provision of a service are calculated on the basis of what would be saved in the long run by considering each area of provision in isolation to all the other areas that are required to provide the entire service.

The long run incremental cost of conveyance is the cost that would be saved in the long run if no traffic were provided over the network, but access were to continue to be provided. The incremental cost of access is the cost that would be saved in the long run if no final links to customers were provided (if conveyance was still provided in theory). Common costs of the network, are the costs of the stand-

alone network that are incurred when conveyance and access are provided simultaneously, but are incremental to neither conveyance nor access on their own.

It is important to note that in considering each area, all factors adding to the cost of that area should be included. For example, when considering transmission, duct investment and maintenance will also contribute to the overall incremental cost for that area, not just the capacity taken-up on the physical transmission media itself.

7.3.2. Replacement Costs

All assets involved in proving a service depreciate over time. As previously mentioned, LRIC use CCA principles with MEAs depreciating non-linearly with time.

7.3.3. Overall Interconnection Charge

From data supplied by the incumbent and other industry benchmark costs, ACNE's regulator will produce a set of recommendations for interconnect charges in the form of floors and ceilings for each area of the incumbent's network used in providing the service. Although exact values of floors and ceilings can't be calculated without extensive network data, the set of charges would be presented as follows:

Item	Floor (**exampl**e values – pence/min)	Ceiling (**example** values – pence/min)
Switching:		
Exchanges B and C	0.15	0.21
DMSU exchanges A, A1 and A2	0.02	0.07
Transmission		
Access	0.32	0.37
Junction	0.03	0.08
Trunk	0.34	0.54

From these figures, the incumbent can calculate the total interconnection charge that they will levy on the competitor on a pence per minute basis.

Network Planning and Performance, 1999

Question 3

An ATM Connection Admission Control Algorithm

Implemented Using Matlab

1. Introduction

Connection Admission Control (CAC) and Usage Parameter Control (UPC) mechanisms are the two main tools used by a network operator to control the entry of new traffic into a network and to police a source's offered traffic once accepted. As such, they must be both accurate and operate in real time. Although operators would like to run their networks at the highest possible loading for maximised profit, there is a trade-off between loading and network performance. Cell loss probability (CLP), cell delay (CD) and cell delay variation (CDV) are intricately linked not just to available system resources, such as transmission line service capacity and buffer length, but also to the level of traffic loading and the source characteristics of each connection. Hence the requirement of a heavily loaded network, offering a high Quality of Service (QOS) isn't easily achieved.

CAC algorithms must be able to accurately and reliable predict the effect that a new traffic source will have on the system's performance. In order to arrive at a connection decision, i.e. to accept or reject a new connection, the CAC algorithm requires information from both the new source describing its cell generation characteristics and about the current state of the network.

Current network conditions fall into two broad categories of dynamic loading dependant data such as established mean and peak loads and the number of sources currently held, and static system attributes such as buffer length and service capacity.

If a CAC algorithm decides to accept a new connection it must not cause established traffic contracts to be violated. In order to do this, there must be a maintained, dynamic database containing the agreed performance parameters of all established sources, together with characteristics such as their mean and peak cell rates. The CAC algorithm must verify that the most stringent traffic contract of all the sources will still be met after the acceptance of a new source.

2. Source Modelling

Sources can be classified into two main types

- Deterministic or Constant Bit Rate sources (DBR / CBR)
- Variable or Statistical Bit Rate sources (VBR / SBR)

DBR sources are considerably easier to design CAC algorithms for because they have just one important characteristic, the cell generation rate. A DBR source will inject one cell then be silent for a number of cell slots, then inject a second cell and so on. The mean rate of cell production is simply the number of cells generated each second. A switch serving DBR sources can accept sources with a combined cell rate upto its service capacity. Due to the multiplexing of multiple asynchronous individual sources however, large groups of cells can arrive in just one time slot. The switch can only service one cell per time slot, so to avoid cell loss, buffers must be provided to temporarily store cells. Since the overall cell generation rate is less than the service rate, there is no long-term build up of cells. Buffer queues providing cell storage in such situations are called cell scale buffers. A switch with a buffer space equal in length to the number of accepted connections will have zero CLP. In practice more connections are accepted than the number of buffer spaces available, resulting in a non-zero CLP. Since cells will generally be subject to cell scale queuing, they will experience a variation in delay between arriving at a switch and being served during the lifetime of a connection. This is called Cell Delay Variation (CDV). The CAC algorithm must access the likely CLP and the CD probability distribution, i.e. the CDV, for the proposed number of connections to the available buffer space and compare this with the requested performance parameters of all established connections each time a new connection request is made.

VBR sources are characterised by a variable rate of cell generation over time. Typically these sources are characterised by their long-term mean cell rate and their maximum, peak cell rate. Commonly these sources are referred to as 'bursty' sources as they produce groups of cells in close succession, followed by a silent period. Multiple bursty sources, may for a short period, generate more cells than the service capacity of the switch they are multiplexed into. Large buffer spaces must be provided to prevent cell loss in these situations. Such buffering is called burst scale queuing. The job of the CAC algorithm is now considerably more complex as both cell and burst scale queuing is present. The CAC algorithm must decide on the number of acceptable VBR sources between the two extreme limits of allocation: on

peak rate (the 'lowest' maximum number of acceptable VBR sources) or allocation on mean rate (the 'highest' maximum number of acceptable VBR sources). Again, a decision to accept a new connection, is made by predicting the likely CLP and CDV and comparing with the most stringent requested QoS.

3. Assignment Specific Details

The buffer in question is small with only 10 spaces. From this, an important simplifying assumption can be made in that the system will exhibit cell scale queuing only. To provide burst scale queuing, many hundreds of spaces are needed before significant reductions in CLP are experienced. Immediately it can be qualitatively deduced that the overall system loading is likely to be very low. The reason is simple: the sources are fairly bursty with peak rates ten times their mean rate (20 000 cells/sec compared with 2 000 cells/sec) and have fairly low peak and mean rates compared with the service rate. This indicates a potentially large maximum number of acceptable connections which can be calculated by considering the source's mean rate:

$$N_{\max} = \frac{C}{m} = \frac{353208}{2000} = 176.7$$

However, the required CLP is very low, $1x10^{-8}$, so without significant burst scale provision in the buffer, the CAC algorithm will have no choice but to significantly limit the number of sources to reduce to probability of a large number of sources being active simultaneously.

The lower limit on the maximum number of connections is given by considering the source's peak rate:

$$N_0 = \frac{C}{h} = \frac{353208}{20000} = 17.6$$

The limits on the maximum possible number of connections allow a second assumption to be made concerning the arrivals process to the buffer. Since a reasonably large number of sources will be multiplexed to the buffer's input (between 17 and 176), the arrivals process can be approximated as Poisson. As such, the Poisson probability generating function can be used to give the arrival distribution per time slot:

$$P(a = k) = \frac{e^{-\lambda} \lambda^{k}}{k!}$$

where the mean rate will be the sum of all connection mean rates:

$$\lambda = \frac{\sum_{i=1}^{n} m_i}{C}$$

4. The CAC algorithm

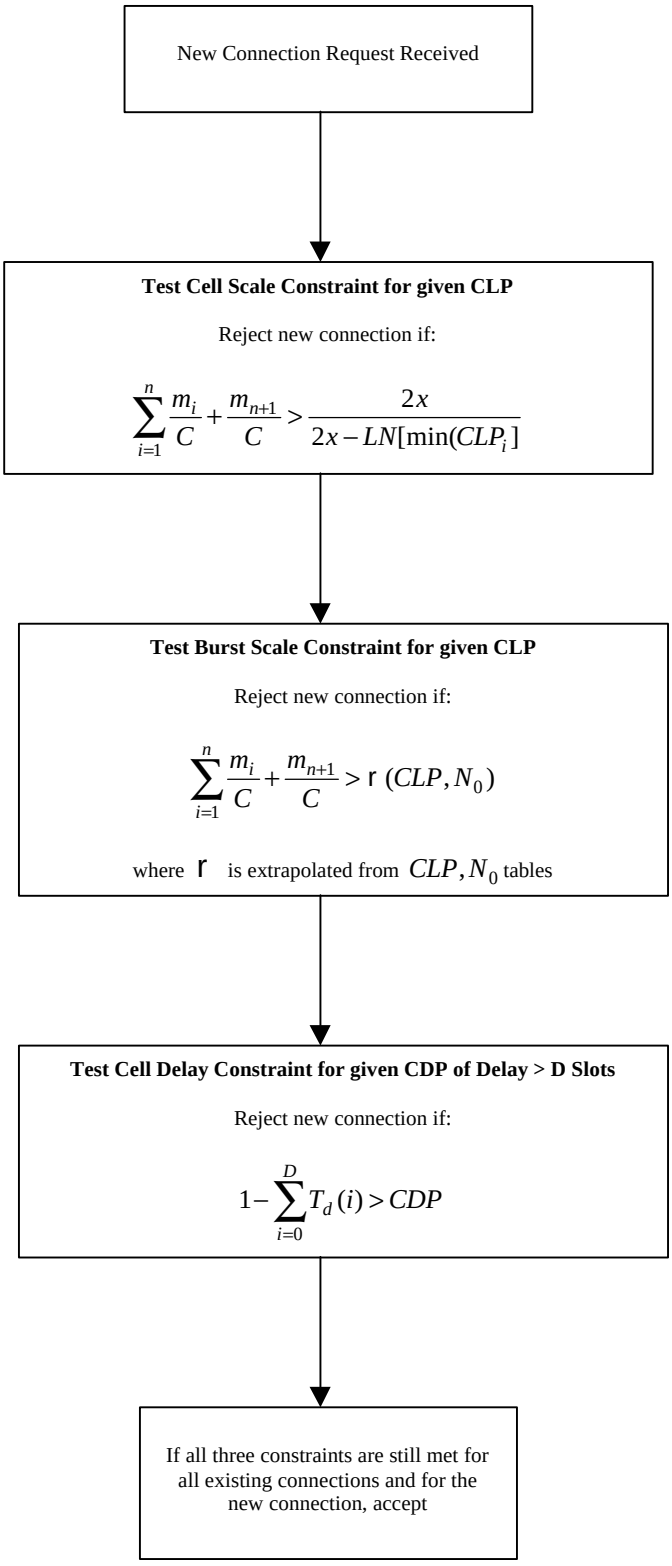
The CAC functions on a layered set of three tests when a new connection request is made:

1. Check the cell scale constraint on CLP
2. If the cell scale constraint is satisfied, check the burst scale constraint on CLP
3. If both the cell and burst scale constraints are satisfied, check the cell delay constraint

The CAC must verify that all three constraints are met for the most stringent QoS requirements of all the existing connections and the new connection before accepting. In realistic situations with non-homogenous sources, it must search is connection table to find the minimum required CLP and CDP for given CD. For the assignment, all connections are the same and have the same QoS requirements, but to make the CAC algorithm more realistic, Matlab code has been written which enables sources with differing peak, mean, required CLP and CDV to be considered. This is achieved by creating an

established connection table and searching for the most stringent QoS requirement each time a new connection is requested. Further details of the CAC implementation are given later in the report.

The CAC algorithm decision's flow is implemented in the Matlab function `received_request`, and is shown below.



4.1. Testing the Cell Scale Constraint

As previously noted, the utilisation is expected to be very low. As such the heavy traffic approximation to the M/D/1 queue will give very optimistic values for acceptable load:

$$\rho = \frac{2x}{2x - Ln[\min CLP]}$$

However, it may still be used as a fast initial CAC test for the cell scale CLP constraint because a far more conservative second test for the burst scale CLP constraint will always follow. It is interesting to note just how inaccurate this approximation is for light loads: exact analysis[1] gives a maximum utilisation of just 29.9% for a buffer with 10 cells at a CLP of $1x10^{-8}$, whereas the M/D/1 approximation would allow a maximum utilisation of 52% !

4.2. Testing the Burst Scale Constraint

Since the arrivals process can be approximated by a Poisson model and the buffer is effectively zero in length at the burst scale, the CLP can be calculated using a bufferless approach. For a given utilisation, a buffer fed by multiple bursty On/Off sources has an approximate CLP given by

$$CLP \approx \frac{1}{(1-\rho)^2 N_0} \cdot \frac{(\rho N_0)^{\lfloor N_0 \rfloor}}{\lfloor N_0 \rfloor !} . e^{-\rho N_0}$$

This approximation can be rearranged to give

$$\frac{(1-\rho)^2}{e^{-\rho N_0}} \rho^{\lfloor N_0 \rfloor} = \frac{1}{CLP} \frac{N_0^{\lfloor N_0 \rfloor}}{N_0 \lfloor N_0 \rfloor !}$$

Since the function is has no turning points over $0 < \rho < 1$, it can be reliably solved using Newton-Raphson differential iteration, with a start value of $\rho = 0.5$. An alternative approach is to use a look-up table of pre-computed values of maximum permissible utilisation, ρ, as a function of maximum CLP and the number of connections if allocation was on peak rate only, N_0 [2]. Since only key values are tabulated, some form of interpolation is required for continuous valued look-up. The CAC algorithm implemented in Matlab uses the '`csaps`' function, which is a cubic spline interpolation function with maximum smoothing. As written, the CAC algorithm only computes for a CLP of $1x10^{-8}$ as only a 2D interpolation is used. To allow the CAC to work for various CLP requirements, a 3D interpolation routine for table values would be needed. This would allow the maximum permissible utilisation to be determined for any value of CLP and N_0.

4.3. Testing the Cell Delay Constraint

The cell delay constraint of the traffic contract forms the third and final test for a new connection. When a cell enters a FIFO buffer it will enter at the end of the queue and wait until all the cells previously queued have been served, and in the worst case, until all the cells arriving in the same batch have also been served. The service of cells already waiting is termed the unfinished work, U_d and the service of cells arriving in the same batch, the batch work, B_d. The overall delay, T_d, for a cell entering a queue is given by:

$$T_d = U_d + B_d$$

[1] Pitts and Schormans, ATM Design and Performance, p 97
[2] Pitts and Schormans, ATM Design and Performance, p 105

The delay distribution due to the unfinished work, $p(U_d = k)$ is simply given by the queue state probabilities.

$$p(U_d = 1) = s(0) + s(1)$$
$$p(U_d = k) = s(k), \quad k > 1$$

Queue state occupancy probabilities, s(k), can be calculated using an infinite buffer model, or more accurately using an actual finite buffer model. The CAC algorithm written in Matlab determines the queue state probabilities using the finite queue model described by the equations below:

$$s(0) = \frac{1}{\sum_{i=1}^{x} u(i)}$$

$$s(k) = s(0)u(k), \quad k > 0$$

$$u(0) = 1$$

$$u(1) = \frac{1 - a(0)}{a(0)}$$

$$u(k) = \frac{u(k-1) - a(k-1) - \sum_{i=1}^{k-1} u(i)a(k-i)}{a(0)} \quad k > 1$$

As stated previously, the cell arrival process is modelled by a Poisson distribution.

The batch work delay probability distribution, $p(B_d = k)$, can be determined from the equation:

$$p(B_d = k) = \frac{1 - \sum_{i=0}^{k} a(i)}{E[a]}$$

where E[a] is the total offered load.

The total delay probability distribution is the convolution of the unfinished and batch delay probabilities:

$$p(T_d = k) = p(U_d = k) \otimes p(B_d = k)$$
$$= \sum_{j=1}^{k} p(U_d = j)p(B_d = k - j)$$

The probability that the delay exceeds the specified number of cell slots is calculated by the Matlab function `calculate_total_delay.`

5. Simulation

With the specified traffic source and required QoS, the CAC algorithm rejects the 35th connection request. The request is rejected due to the QoS CLP requirement of $1x10^{-8}$, rather than the specified cell delay constraint that a cell should only experience a delay of greater than 5 time slots with a probability of less than $1x10^{-5}$.

With 34 connections, the system performance is predicted as:

Cell Loss Probability = 8.835×10^{-9}
Probability of a cell waiting more than 5 time slots = 2.807×10^{-6}

6. Matlab Implementation

In order to create a more realistic CAC algorithm, considerable time has been spent writing Matlab code to implement a CAC algorithm which allows sources with different mean and peak rates, requested CLP and cell delay requirements to be connected to the buffer. The CAC algorithm creates a connection table (`conn_table`) which holds the source parameters and QoS requirements of every source that has been accepted onto the network. Connections to real switches are obviously being continually made and established ones released. To reflect this, the function `test_new_conn` allows new connection requests to be made with specifiable parameters and if accepted, adds an entry to the connection table. The function `conn_release` allows established connections to be released by removing the relevant entry from the connection table. When new connections are accepted or old ones released, system data in the structure `estab,` which includes the number of connections currently held, their combined mean and peak rates and other information are recomputed and updated

The decision processes of the CAC are implemented in the function `received-request`. The final cell delay constraint is tested by calculating the cell delay distribution using the function `calculate_total_delay.`

All Matlab functions' m files have been included to enable a quick assessment of the scope of the written CAC algorithm to be made.

Parameter Initiation in Matlab (MD1PeakMeanCAC.m)

```
%switch_params: General parameters relating to the ATM switch running the CAC algorithm

k = 1000;   %Maximum possible number of connections
x = 10;     %Buffer length
C = 353208; %Cell service rate in cells per sec

switches = struct('C', C, 'x', x, 'k', k);



%-------------------------------------------------------------------------------------------------------
%conn_table: Current connection table
%Row1 contains the connections' requested CLP
%Row2 contains the connections' stated peak load
%Row3 contains the connections' stated mean load
%Row4 contains the connections' requested maximum CD only to be exceeded with probablity less than the CDP
%Row5 contains the connections' requested CDP for a cell to queue longer than the stated max CD

conn_table(1,:) = ones(1,k);  %make all CLPs one
conn_table(2,:) = zeros(1,k); %make all peak loads zero
conn_table(3,:) = zeros(1,k); %make all mean loads zero
conn_table(4,:) = x;          %make all max CDs 'x' in length
conn_table(5,:) = ones(1,k);  %make all CDPs 'x' in length



%-------------------------------------------------------------------------------------------------------
%established_params: List of important parameters for all connections so far accepted

min_CLP = 1;
min_CD = x;
min_CDP = 1;
sum_m = 0;
sum_h = 0;
no_conns = 0;
N0 = 0;
N0Full = 0;

estab = struct('no_conns',no_conns,'sum_m’,sum_m,'sum_h',sum_h,'min_CLP',min_CLP,'min_CDP',min_CDP,'min_CD',min_CD,'N0',N0,'N0Full',N0Full);



%-------------------------------------------------------------------------------------------------------
%new_params: List of parameters of the new requested connection
```

```
%default: Assignment specific connection parameters

CLP = 0.00000001;
CD = 5;
CDP = 0.00001;
h = 20000;
m = 2000;

default = struct('m', m, 'h', h, 'CLP', CLP, 'CD', CD, 'CDP', CDP);
new = struct('m', m, 'h', h, 'CLP', CLP, 'CD', CD, 'CDP', CDP);



%-----------------------------------------------------------------------------------------------------
%Actual performance
actual = struct('CLP', 0, 'CDP', 0);


%-----------------------------------------------------------------------------------------------------

accept_reject = 0;

format long g

clear C x k;
clear CLP CD CDP h m
clear min_CLP min_CD min_CDP sum_m sum_h no_conns N0 N0Full;
```

New Connection Request Initiation (test-new-conn.m)

```
% Ask to use default parameters or prompt user for new ones
% In a real situation, these would be supplied by the requesting % connection at setup

n = input('Use default connection parameters (1 for yes)?');
if n == 1
   new = default;
else
   new.m = input('New connections mean cell rate: ');
   new.p = input('New connections peak cell rate: ');
   new.CLP = input('New connections requested CLP: ');
   new.CD = input('New connections minimum buffer length for CDP: ');
   new.CDP = input('New connections CDP for buffer length > CD: ');
end

% Call CAC algorithm when a new connection is requested with parameters in data structure 'new'
[accept_reject, actual] = received_request(switches, estab, new, actual);

% CAC agorithm makes a decision on whether to accept or reject the new connection
% If accepted, 'accept_reject' is set
```

Decision Making Process in CAC (received-request.m)

```
function [accept_reject, actual] = received_request(switches, estab, new, actual)

%function [accept_reject, actual] = received_request(switches, estab, new, actual)

%This is the decision part of the CAC algorithm which determines the most stringent QoS
%parameters of all existing connections and the new requested connection


%First test the Cell scale constraint considering all source mean loads
test_mean_load = (new.m/switches.C) + (estab.sum_m/switches.C); %New mean load with new connection added

Q = min([estab.min_CLP, new.CLP]); %Find minimum CLP of all connections, established and new

test_mean_value = (2*switches.x)/(2*switches.x - (log(Q)));

if test_mean_load < test_mean_value %Tests the new connection's effect on CLP calculated using total mean loading

   %If Cell scale constraint OK, test for the Burst scale consraint
   %A full implementation would require a lookup table for all possible values of N0, CLP and maximum permissible load,
   %or an extrapolation algorithm to find the maximum permissible load from a finite table contain key values of N0, CLP
   %and the maximum permissible load.
   %Alternatively and iterative solution to the approximate formulae for the burst scale loss factor could be employed

   %A full solution will require interpolation of the Burst scale - utilisation/CLP/N0 table for any value of CLP and calculated N0
   %This solution assumes CLP = 1x10-8 and any value of N0 between 1 and 100

   N0_table = 10:10:100; %Values of N0
   max_utilisation08 = 0.01 .* [9.7 23.6 32.9 39.4 44.3 48.2 51.3 54.0 56.2 58.1]; %Values of max load for CLP = 1x10-8
   %max_utilisation07 = 0.01 .* [12.5 27.4 36.8 43.3 48.1 51.8 54.8 57.3 59.5 61.3];%Values of max load for CLP = 1x10-7
   %max_utilisation06 = 0.01 .* [16.2 32.0 41.4 47.7 52.4 55.9 58.8 61.2 63.2 64.9];%Values of max load for CLP = 1x10-6


   mean_h = (estab.sum_h + new.h)/(estab.no_conns + 1);
   N0 = switches.C/mean_h;
   test_N0 = floor(N0);

   %This solution uses an extrapolation function, the matlab function 'csaps' to find the load for N0=17, CLP =1x10-8
   %'csaps' is a cubic spline interpolation function with a specified degree of smoothing, here maximum smoothing is used
   burst_utilisation = csaps(N0_table, max_utilisation08, 1, test_N0);

   mean_m = (estab.sum_m + new.m)/(estab.no_conns + 1);
```

```
   max_accept_burst_scale = (burst_utilisation * switches.C)/(mean_m);

   if estab.no_conns + 1 < max_accept_burst_scale

      %If Burst scale constraint OK, test for new connections effect on the CD

      [accept_reject, actual] = calculate_total_delay(switches, estab, new, actual); %Accepted or Rejected on CD constraint
      if accept_reject == 0
         str = 'Connection Rejected on Cell Delay Constraint'
      end
   else
      accept_reject = 0; %Rejected on Burst Scale Constraint
      str = 'Connection Rejected on Burst Scale Constraint'
   end
else
   accept_reject = 0; %Rejected on Cell Scale Constaint
   str = 'Connection Rejected on Cell Scale Constraint'
end

if accept_reject
   str = 'Connection Accepted'
end
```

Cell Delay Probability Calculation (calculate_total_delay.m)

```
%function [accept_reject, actual] = calculate_total_delay(switches, estab, new, actual)

%This function calculates the Cell Delay Probability for a cell to wait longer than the minimum
%Cell Delay ‘CD_min’

%Calculate the mean offered load
lambda = (estab.sum_m + new.m)/ switches.C ;

%Assume Poisson arrivals and generate the cell arrival distribution
k = exp(-lambda);
for i=1:1:switches.x+1
   a(i) = (lambda^(i-1) * k)/factorial(i-1);
end

%Calculate the intermediate variable 'u' to calulate queue state occupancies 's' for a finite buffer
u(1) = 1;
u(2) = (1 - a(1))/a(1);
for k=3:1:switches.x+1
   temp = 0;
   for i = 1:1:k-2
      temp = temp + u(i + 1)*a(k - i);
   end
   u(k) = (u(k - 1) - a(k - 1) - temp)/a(1);
end

s(1) = 1/sum(u(1:switches.x+1));
for i=2:1:switches.x+1
   s(i)=s(1)*u(i);
end

%Calaculate the unfinished work in front of newly arriving cells, 'Ud'
Ud(1) = 0;
Ud(2) = s(1) + s(2); %matlab vector entry Ud(2) = actual Ud(1)
Ud(3:min([estab.min_CD new.CD])+1) = s(3:min([estab.min_CD new.CD])+1);

%Calculate the work of cells arriving in the same batch, ‘Bd'
for i=1:1:min([estab.min_CD new.CD])
   Bd(i) = (1 - sum(a(1:i))) / lambda;
end

%Calculate the overall CDP 'Td' for queue length CD_min by convolving the
%probabilities for unfinished work and newly arrived preceeding work
Td = conv(Ud, Bd);
```

```
sum_Td = sum(Td(1:min([estab.min_CD new.CD])+1));

%Test for most stringent CDP for minimum CD
if 1 - sum_Td <= min([estab.min_CDP new.CDP])
   accept_reject = 1;
   actual.CDP = 1 - sum_Td;
else
   accept_reject = 0;
end
```

System Update If a New Connection is Accepted (update-connection-table-accept.m)

```
function [conn_table, switches, estab, new, actual] = update_connection_table_accept(conn_table, switches, estab, new, actual)

%function [conn_table, switches, estab, new, actual] = update_connection_table_accept(conn_table, switches, estab, new, actual)

%This function takes care of updating the connection table, established system parameters
%and actual performance data structures

%Find first free space in connection table
first_spare_place = switches.k;
i = 0;
not_found = 1;

spare_entries = conn_table(2,:) == 0; %Looks for first entry with zero peak traffic
                                      %(Could search for any of the default zeroed paramters
while not_found == 1 & i < switches.k
   i = i + 1;
   if spare_entries(i) == 1
      first_spare_place = i;
      not_found = 0;
   end
end

%Update established connection parameters to include new connection
estab.min_CLP = min([estab.min_CLP new.CLP]);
estab.min_CD = min([estab.min_CD new.CD]);
estab.min_CDP = min([estab.min_CDP new.CDP]);
estab.sum_m = estab.sum_m + new.m;
estab.sum_h = estab.sum_h + new.h;
estab.no_conns = estab.no_conns + 1;
mean_h = estab.sum_h/estab.no_conns;
estab.N0Full = switches.C/mean_h;
estab.N0 = floor(switches.C/mean_h);

%Update connection table to include new connection
conn_table(1,first_spare_place) = new.CLP;
conn_table(2,first_spare_place) = new.h;
conn_table(3,first_spare_place) = new.m;
conn_table(4,first_spare_place) = new.CD;
conn_table(5,first_spare_place) = new.CDP;


%Calculate the actual CLP and update actual parameters
temp1 = 1/(estab.N0Full* (1 - estab.sum_m/switches.C)^2);
```

```
temp2 = ((estab.sum_m/switches.C*estab.N0Full)^estab.N0)/(factorial(estab.N0));
temp3 = exp(-estab.sum_m/switches.C*estab.N0Full);
actual.CLP = temp1 * temp2 * temp3;
```

Script File to Release a Connection (conn-release.m)

```
%Script file to ask which connection to release
%Actual release in a real system would be triggered by connection release signalling from the connection host

n = input('Release which connection ?');

[conn_table, switches, estab, new] = update_connection_table_release(n, conn_table, switches, estab, new);
```

System Update If a Connection Release Occurs (update_connection_table_release.m)

```
function [conn_table, switches, estab, new, actual] = update_connection_table_release(n, conn_table, switches, estab, new, actual)

%function [conn_table, switches, estab, new, actual] = update_connection_table_release(n, conn_table, switches, estab, new, actual)

%This function removes a connection from the connection table and updates all necessary data structures
if conn_table(2,n) ~= 0
   %Remove connection's effect on established connection paramaters
   estab.no_conns = estab.no_conns - 1;
   estab.sum_h = estab.sum_h - conn_table(2,n);
   estab.sum_m = estab.sum_m - conn_table(3,n);
   estab.min_CLP = min(conn_table(1,:));
   estab.min_CD = min(conn_table(4,:));
   estab.min_CDP = min(conn_table(5,:));

   %Remove connection's entry in connection table
   conn_table(1,n) = 1;
   conn_table(2,n) = 0;
   conn_table(3,n) = 0;
   conn_table(4,n) = switches.x;
   conn_table(5,n) = 1;
end
```

Factorial Calculation (factorial.m)

```
function [x_factorial] = factorial(x)

%function [x_factorial] = factorial(x)

%A function to calculate factorials, surprisingly not included in matlab
if x ~= 0
   x_factorial = 1;
   for i=1:1:x
      x_factorial = i * x_factorial;
   end
else
   x_factorial = 1;
end
```

Network Planning and Performance, 1999

Question 4

The Analysis of Priority Queuing Control in ATM

Buffers Using Matlab

1. Source Modelling and Results Processing

The basic method used to generate both the Bernoulli and Poisson arrivals was to compare randomly generated numbers between 0 and 1 with the Bernoulli success probability or the relevant Poisson cumulative distribution function.

The matlab function `rand(1, no_cells)` generates a row vector with 'no_cells' random numbers. In matlab version 5 the `rand` function theoretically repeats itself once in 2^{1429} numbers. To test the randomness of this function a RUNS test was performed on the vectors of 100 000 random numbers using the statistical package GENSTAT. Not surprisingly, the data produced by the matlab `rand` function showed no significant deviation from randomness at the 5% level. With this assurance, the `rand` function could be used to generate random cell streams.

The function `my_bernoulli_arrivals(p, n, no_cells)` forms a vector of Bernoulli inputs for 'n' sources, each with a probability of success, 'p'. Initially the matlab function `binoinv(z, n, p)`, with 'z' being a row vector of 100 000 random numbers was used, but due to an excessive processing time of nearly a minute to generate 100 000 arrivals, a customised function was written. '`if`' statements were used to compare the random numbers with the probability of success, but again this approach was too slow. Matlab implements logical statements on vectors very rapidly, hence the function was re-written using logical statements to compare the random numbers with success probability, for example:

```
arrivals = p_vector > z;
```

Simply, if `p_vector > z` then `arrivals` is true (ie `arrivals = 1`), otherwise `arrivals` is false (ie `arrivals = 0`). This is repeated for each Bernoulli input and summed to give an arrivals vector for the five Bernoulli sources. For 100 000 cell slots, computation time is approximately 500 ms.

A similar approach was used to implement a function to generate the Poisson arrivals, although this was forced by a bug in the Matlab `poissinv` m file function, preventing it from handling vectored inputs. With a logical operator approach to testing the random number's value compared with values of the Poisson cumulative probability distribution, arrivals from the source over 100 000 cell slots are generated in approximately 1 s by the function `my_poisson_arrivals(lambda, no_cells)`.

To verify that the arrival vectors were representative of Bernoulli and Poisson sources, both vectors were tested for their:

- Mean ($E_{Binomial}[x] = n\,p$; $E_{Poisson}[x] = lambda$)
- Variance ($var_{Binomial}[x] = n\,p\,(1-p)$; $var_{Poisson}[x] = lambda$)

Finally, a function was written to generate the high and low priority buffer queues, state probabilities and the service channel occupancy. Once again, logical statements were used to determine which queue should be served. The lines from the function which determines which priority queue will be served in each time slot are shown below:

```
hp_channel(t_slot) = hp_q_length(t_slot) > 0;

lp_channel(t_slot) = (hp_q_length(t_slot) == 0) &
                                          (lp_q_length(t_slot) > 0);

service_channel(t_slot) = sum(hp_channel(t_slot) + lp_channel(t_slot));
```

With this approach, the service channel is effectively represented by the sum of the two 'virtual' service channels: one for high and one for low priority cells. This method enables a graphical display of the queue evolution to be visualised and the accuracy of the queue service algorithm verified. The upper graph in figures 1, 3 and 5, show the first twenty cell arrivals for each traffic scenario simulation of 100 000 cells. In

tables 1 and 2, the queue state probabilities are first grouped by queue type, then by traffic profile for ease of reference.

	HPQ State Probabilities			LPQ State Probabilities			Overall State Probabilities		
	p = 0.15 λ = 0.15	p = 0.09 λ = 0.45	p = 0.02 λ = 0.80	p = 0.15 λ = 0.15	p = 0.09 λ = 0.45	p = 0.02 λ = 0.80	p = 0.15 λ = 0.15	p = 0.09 λ = 0.45	p = 0.02 λ = 0.80
State									
0	5.67E-01	8.82E-01	9.96E-01	3.62E-01	2.71E-01	2.47E-01	2.63E-01	2.55E-01	2.47E-01
1	2.12E-01	9.42E-02	4.29E-03	1.81E-01	1.50E-01	1.43E-01	1.56E-01	1.45E-01	1.43E-01
2	1.07E-01	1.98E-02	9.00E-05	1.20E-01	1.19E-01	1.20E-01	1.26E-01	1.20E-01	1.20E-01
3	5.51E-02	3.77E-03	0	8.54E-02	9.22E-02	9.48E-02	9.91E-02	9.49E-02	9.50E-02
4	2.80E-02	6.00E-04		5.90E-02	7.05E-02	7.84E-02	7.70E-02	7.48E-02	7.85E-02
5	1.53E-02	1.00E-04		4.74E-02	5.81E-02	6.33E-02	5.84E-02	5.97E-02	6.34E-02
6	8.07E-03	0		3.37E-02	4.75E-02	5.05E-02	4.69E-02	4.94E-02	5.05E-02
7	4.11E-03			2.51E-02	3.81E-02	3.82E-02	3.83E-02	4.01E-02	3.83E-02
8	1.90E-03			1.94E-02	2.93E-02	3.23E-02	3.02E-02	3.09E-02	3.23E-02
9	9.90E-04			1.49E-02	2.40E-02	2.55E-02	2.28E-02	2.49E-02	2.55E-02
10	4.50E-04			1.01E-02	1.93E-02	2.07E-02	1.72E-02	1.98E-02	2.07E-02
11	2.00E-04			8.44E-03	1.46E-02	1.76E-02	1.32E-02	1.56E-02	1.77E-02
12	1.60E-04			6.06E-03	1.18E-02	1.41E-02	9.78E-03	1.25E-02	1.41E-02
13	1.10E-04			5.17E-03	9.61E-03	1.06E-02	8.33E-03	1.01E-02	1.07E-02
14	1.10E-04			4.64E-03	7.71E-03	8.31E-03	6.33E-03	8.03E-03	8.33E-03
15	6.00E-05			3.99E-03	6.19E-03	7.24E-03	5.42E-03	6.50E-03	7.24E-03
16	2.00E-05			3.54E-03	5.25E-03	5.55E-03	4.83E-03	5.24E-03	5.56E-03
17	0			2.73E-03	3.62E-03	4.30E-03	4.39E-03	4.18E-03	4.31E-03
18				2.10E-03	3.68E-03	3.12E-03	3.23E-03	3.83E-03	3.12E-03
19				1.28E-03	3.44E-03	2.77E-03	1.95E-03	3.51E-03	2.76E-03
20				7.20E-04	3.03E-03	2.19E-03	1.29E-03	3.16E-03	2.21E-03
21				8.10E-04	2.04E-03	1.83E-03	1.17E-03	2.12E-03	1.82E-03
22				3.00E-04	1.89E-03	1.84E-03	1.06E-03	1.88E-03	1.84E-03
23				2.10E-04	1.43E-03	1.39E-03	6.70E-04	1.65E-03	1.38E-03
24				8.60E-04	1.35E-03	1.13E-03	5.60E-04	1.32E-03	1.15E-03
25				3.50E-04	1.13E-03	9.30E-04	7.20E-04	1.14E-03	9.30E-04
26				1.30E-04	8.90E-04	6.30E-04	3.60E-04	9.70E-04	6.30E-04
27				2.70E-04	8.00E-04	3.90E-04	1.70E-04	7.80E-04	3.90E-04
28				1.30E-04	6.80E-04	2.10E-04	1.90E-04	8.10E-04	2.10E-04
29				2.50E-04	7.60E-04	1.20E-04	1.50E-04	7.40E-04	1.20E-04
30				6.00E-05	5.70E-04	5.00E-05	1.60E-04	5.70E-04	5.00E-05
31				8.00E-05	2.70E-04	1.00E-04	1.20E-04	3.90E-04	1.00E-04
32				1.00E-05	6.00E-05	1.90E-04	1.50E-04	1.30E-04	1.90E-04
33				0	6.00E-05	7.00E-05	7.00E-05	6.00E-05	7.00E-05
34					7.00E-05	1.50E-04	8.00E-05	6.00E-05	1.50E-04
35					4.00E-05	2.40E-04	8.00E-05	6.00E-05	2.40E-04
36					6.00E-05	2.10E-04	5.00E-05	6.00E-05	2.10E-04
37					2.00E-05	2.20E-04	5.00E-05	2.00E-05	2.20E-04
38					0	1.40E-04	4.00E-05	0	1.40E-04
39						3.00E-05	0		3.00E-05
40						9.00E-05			9.00E-05
41						1.20E-04			1.20E-04
42						1.00E-04			1.00E-04
43						2.00E-05			2.00E-05
44						0			0
								.	
Mean Length	0.906	0.148	0.004	2.506	3.698	3.863	3.412	3.846	3.868

Table 1 Queue State Probabilities Listed by Queue Type

	Simulation Scenario 1			Simulation Scenario 2			Simulation Scenario 3		
	p = 0.15, λ = 0.15			p = 0.09, λ = 0.45			p = 0.02, λ = 0.80		
	HPQ	LPQ	Overall	HPQ	LPQ	Overall	HPQ	LPQ	Overall
State									
0	5.67E-01	3.62E-01	2.63E-01	8.82E-01	2.71E-01	2.55E-01	9.96E-01	2.47E-01	2.47E-01
1	2.12E-01	1.81E-01	1.56E-01	9.42E-02	1.50E-01	1.45E-01	4.29E-03	1.43E-01	1.43E-01
2	1.07E-01	1.20E-01	1.26E-01	1.98E-02	1.19E-01	1.20E-01	9.00E-05	1.20E-01	1.20E-01
3	5.51E-02	8.54E-02	9.91E-02	3.77E-03	9.22E-02	9.49E-02	0	9.48E-02	9.50E-02
4	2.80E-02	5.90E-02	7.70E-02	6.00E-04	7.05E-02	7.48E-02		7.84E-02	7.85E-02
5	1.53E-02	4.74E-02	5.84E-02	1.00E-04	5.81E-02	5.97E-02		6.33E-02	6.34E-02
6	8.07E-03	3.37E-02	4.69E-02	0	4.75E-02	4.94E-02		5.05E-02	5.05E-02
7	4.11E-03	2.51E-02	3.83E-02		3.81E-02	4.01E-02		3.82E-02	3.83E-02
8	1.90E-03	1.94E-02	3.02E-02		2.93E-02	3.09E-02		3.23E-02	3.23E-02
9	9.90E-04	1.49E-02	2.28E-02		2.40E-02	2.49E-02		2.55E-02	2.55E-02
10	4.50E-04	1.01E-02	1.72E-02		1.93E-02	1.98E-02		2.07E-02	2.07E-02
11	2.00E-04	8.44E-03	1.32E-02		1.46E-02	1.56E-02		1.76E-02	1.77E-02
12	1.60E-04	6.06E-03	9.78E-03		1.18E-02	1.25E-02		1.41E-02	1.41E-02
13	1.10E-04	5.17E-03	8.33E-03		9.61E-03	1.01E-02		1.06E-02	1.07E-02
14	1.10E-04	4.64E-03	6.33E-03		7.71E-03	8.03E-03		8.31E-03	8.33E-03
15	6.00E-05	3.99E-03	5.42E-03		6.19E-03	6.50E-03		7.24E-03	7.24E-03
16	2.00E-05	3.54E-03	4.83E-03		5.25E-03	5.24E-03		5.55E-03	5.56E-03
17	0	2.73E-03	4.39E-03		3.62E-03	4.18E-03		4.30E-03	4.31E-03
18		2.10E-03	3.23E-03		3.68E-03	3.83E-03		3.12E-03	3.12E-03
19		1.28E-03	1.95E-03		3.44E-03	3.51E-03		2.77E-03	2.76E-03
20		7.20E-04	1.29E-03		3.03E-03	3.16E-03		2.19E-03	2.21E-03
21		8.10E-04	1.17E-03		2.04E-03	2.12E-03		1.83E-03	1.82E-03
22		3.00E-04	1.06E-03		1.89E-03	1.88E-03		1.84E-03	1.84E-03
23		2.10E-04	6.70E-04		1.43E-03	1.65E-03		1.39E-03	1.38E-03
24		8.60E-04	5.60E-04		1.35E-03	1.32E-03		1.13E-03	1.15E-03
25		3.50E-04	7.20E-04		1.13E-03	1.14E-03		9.30E-04	9.30E-04
26		1.30E-04	3.60E-04		8.90E-04	9.70E-04		6.30E-04	6.30E-04
27		2.70E-04	1.70E-04		8.00E-04	7.80E-04		3.90E-04	3.90E-04
28		1.30E-04	1.90E-04		6.80E-04	8.10E-04		2.10E-04	2.10E-04
29		2.50E-04	1.50E-04		7.60E-04	7.40E-04		1.20E-04	1.20E-04
30		6.00E-05	1.60E-04		5.70E-04	5.70E-04		5.00E-05	5.00E-05
31		8.00E-05	1.20E-04		2.70E-04	3.90E-04		1.00E-04	1.00E-04
32		1.00E-05	1.50E-04		6.00E-05	1.30E-04		1.90E-04	1.90E-04
33		0	7.00E-05		6.00E-05	6.00E-05		7.00E-05	7.00E-05
34			8.00E-05		7.00E-05	6.00E-05		1.50E-04	1.50E-04
35			8.00E-05		4.00E-05	6.00E-05		2.40E-04	2.40E-04
36			5.00E-05		6.00E-05	6.00E-05		2.10E-04	2.10E-04
37			5.00E-05		2.00E-05	2.00E-05		2.20E-04	2.20E-04
38			4.00E-05		0	0		1.40E-04	1.40E-04
39			0					3.00E-05	3.00E-05
40								9.00E-05	9.00E-05
41								1.20E-04	1.20E-04
42								1.00E-04	1.00E-04
43								2.00E-05	2.00E-05
44								0	0
Mean Length	0.906	2.506	3.412	0.148	3.698	3.846	0.004	3.846	3.868

Table 2 Queue State Probabilities Listed by Traffic Scenario

2. Results

Initially, the system was simulated for just 1000 time slots, but the results were far from satisfactory for the higher queue occupancies due to the limited incidence of these states. This was the main reason for code optimisation which would allow very long simulation runs. Although all simulations were of 100 000 time slots in length, the plots of state probabilities on log scales still show considerable fluctuations for large buffer lengths (centre plots, figures 2, 4 and 5). The reason is simple: these buffer states are occurring only once or twice during the simulation. A longer simulation run would smooth these results, but as a result, new peakiness would be seen for even longer buffer states.

In order to draw conclusions from the results obtained concerning the 'benefits' of buffer priority scheduling, it is useful to consider the service mechanism's effect on high priority cells compared with when only one queue is used for both inputs. Figure 7 shows the queue state probabilities when cells from both high and low priority cell sources are buffered in just one queue, compared with the state probabilities for the high priority queue for the same simulation run. (The queue state probabilities for a single queue are actually the same as the state probabilities for the two priority queuing system as a whole.)

Figure 7 shows that the queue state probabilities for the system as a whole are similar for all three traffic patterns. This result is expected as the total loading in each case is 90%, the only difference being in the proportion of traffic from each type of source.

The average HP queue length increases as the overall proportion of HP traffic increases, but is always considerably less than the average LP queue length, shown in Figures 2, 4 and 6.

3. Conclusions From Simulations

From figure 7, for all three traffic scenarios it can be seen that the queue lengths as a whole (equivalent to having a single non-prioritorised queue), exhibit considerable variation in length for total loads of 90% compared to the variation in length of the high priority queue. In this context, 'considerable variation' refers to a queue state distribution with a large span of queue lengths, each with a significant probability of occurrence. This holds even when the proportion of traffic entering the HP buffer is as high as 80%. Clearly, the priority system is minimising the delays experienced by HP traffic at the expense of longer delays for LP traffic.

With reference to the traffic scenario of 45% loading with HP cells and 45% with LP cells (centre plot, figure 7), a cell from an HP source can expect delays of greater then three time slots with a probability of only 0.001. However, with a single queue, any cell would expect to be delayed this amount with a probability of 0.1. Infact, with this 90% overall load, 1 cell per 1000 time slots entering a non-prioritorised buffer would on average be delayed by upto 25 time slots. If this cell happened to contain frame synchronisation data from a real time video source, that frame would probably be lost to the viewer.

4. Time Priority Applications and Practical Considerations

One advantage of a priority queuing system, is to allow an operator to offer a grade of service in terms of an upper bound on cell delay and therefore CDV for high priority traffic, but still load their network up with extra lower priority traffic without degrading the contract for the HP users. To illustrate this, the queuing system was additionally simulated for the following scenarios:

- 5 HP sources with p = 0.15 **only** (Network loading 75%)
- 6 HP sources with p = 0.15 (Network loading 90%)
- 5 HP sources with p = 0.15 and a LP Poisson source with mean 0.15 (Network loading 90%)

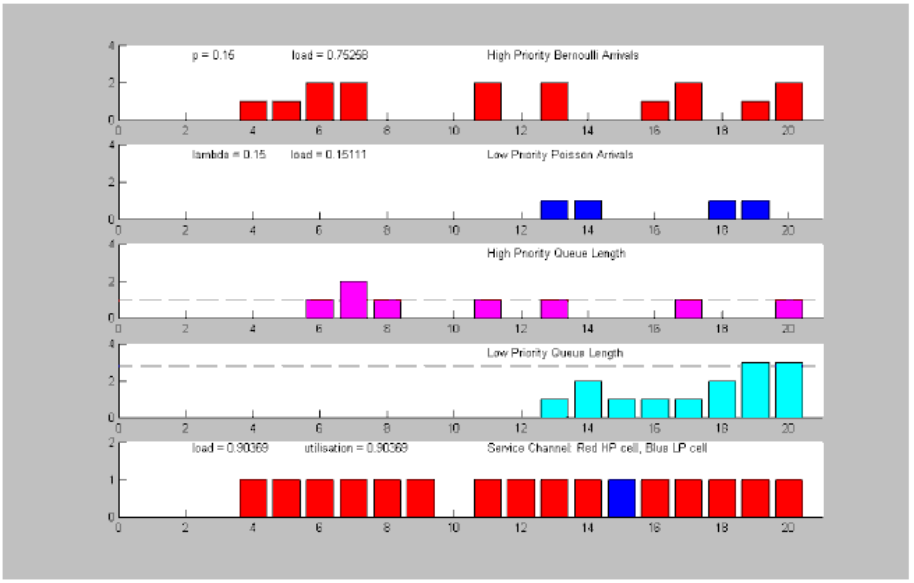
The high priority queue state occupancies are tabulated in table 3. The rationale for the operator would be to guarantee a maximum cell delay and CDV to HP users based on the network's loading of 5 HP sources only (75%). If the operator increased the HP loading to 90% with more HP traffic, the performance will be degraded. However, if the loading is increased to 90% by the inclusion of 15% LP traffic, the performance is not degraded for the existing HP users. Although the tariff charged for LP cells will be less than that for HP cells, the operator will have increased their revenue compared with only loading the network to 75% for fear of violating an agreed traffic contract.

	75% Load p=0.15, n=5, λ=0	**90% Load p=0.15, n=6, λ=0**	**90% Load p=0.15, n=5, λ=0.15**
State			
0	5.66E-01	2.60E-01	5.67E-01
1	2.08E-01	1.56E-01	2.12E-01
2	1.09E-01	1.26E-01	1.07E-01
3	5.58E-02	9.59E-02	5.51E-02
4	3.01E-02	7.39E-02	2.80E-02
5	1.46E-02	5.75E-02	1.53E-02
6	7.33E-03	4.68E-02	8.07E-03
7	4.33E-03	3.83E-02	4.11E-03
8	2.05E-03	2.97E-02	1.90E-03
> 9	2.10E-03	1.16E-01	1.52E-03
Mean HPQ length	0.88	3.11	0.91

Table 3 High Priority Queue State Probabilities in Priority Queuing System

Practically, partitioning traffic by priority may be accomplished using the VC/VP identifiers in a cell's header to direct it to the relevant queue. More than two levels of priority may be used, the highest priority maybe assigned 10% of the channel capacity and as previously mentioned, reserved for extremely delay sensitive traffic such as video framing cells. Any real-time application will benefit from a time priority mechanism, other examples being voice and interactive video transmission. Non real-time traffic such as data from e-mail and file transfers can safely be queued in the lowest priority buffers.

Time priority queues are simplest to implement as separate queues, but this has the effect of increasing CLP for a given memory resource since the total memory space is separated into parts and not shared. An alternative approach to achieve service differentiation which retains a single, shared resource buffer, is through 'logical' partitioning, but is at the expense of added complexity and cost.



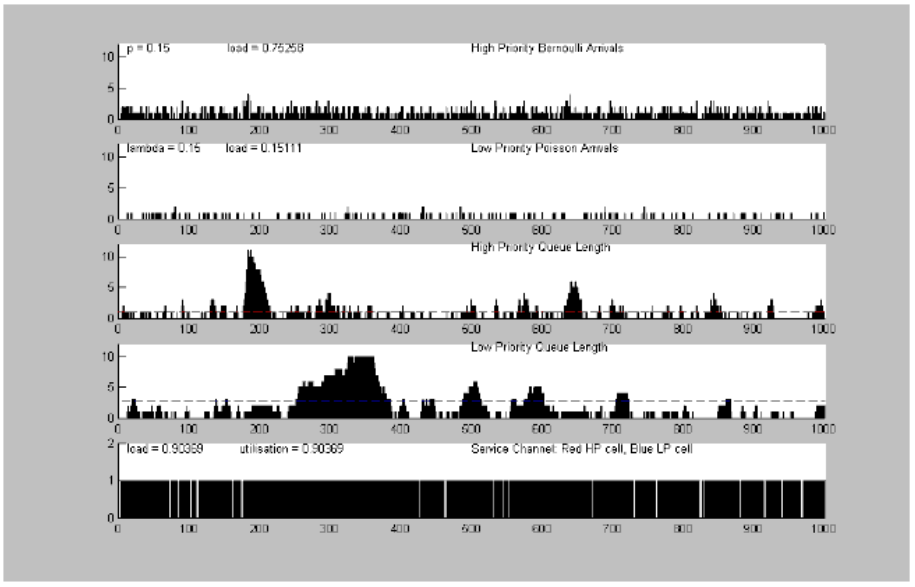
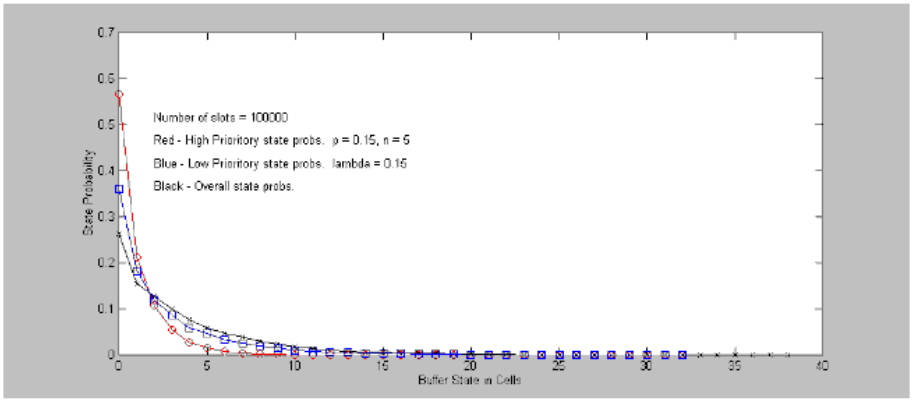
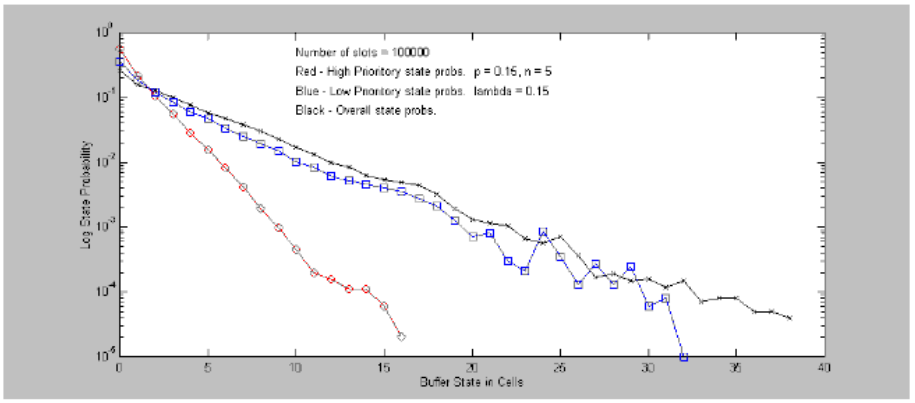


Figure 1 Arrivals, Queue States and Service Channel ($p = 0.15$, $\lambda = 0.15$)





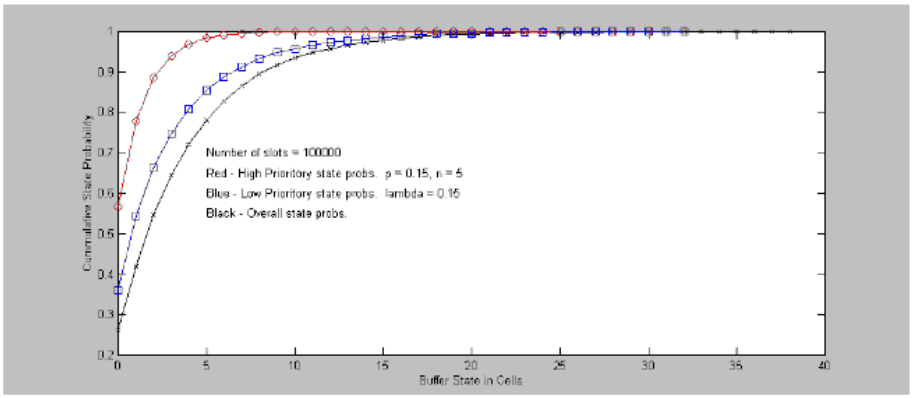
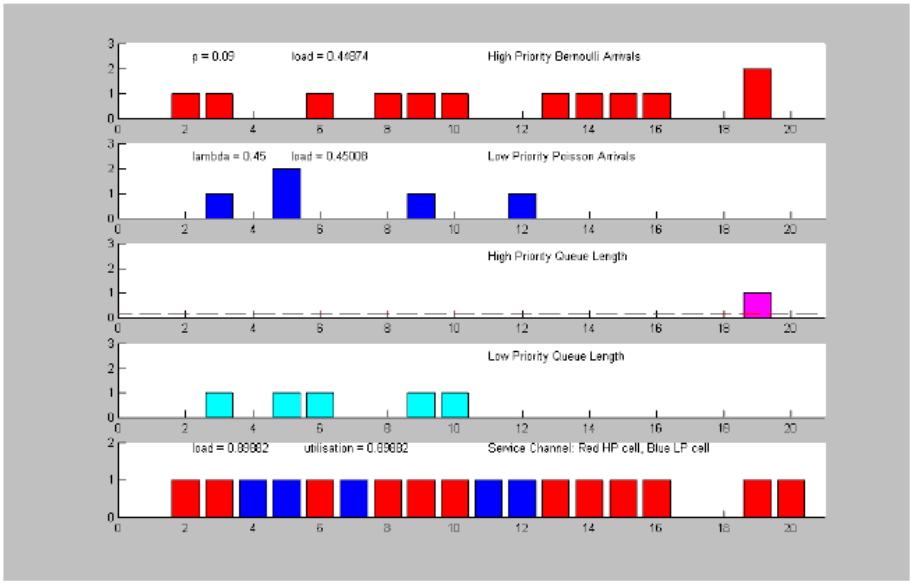


Figure 2 Queue State Probabilities (p = 0.15, λ = 0.15)



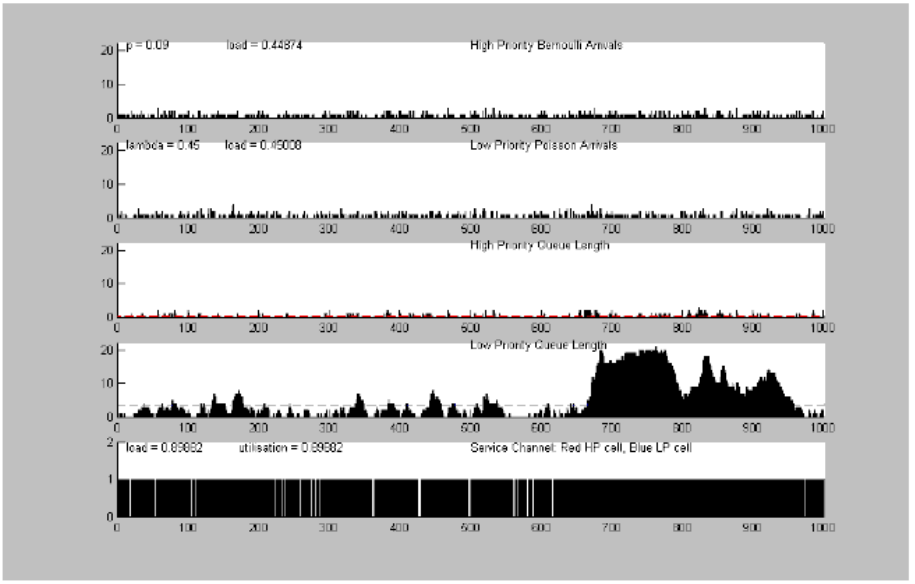
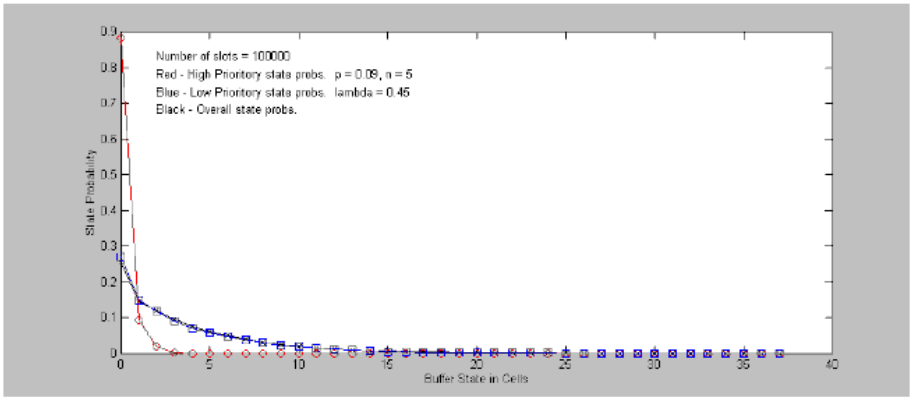
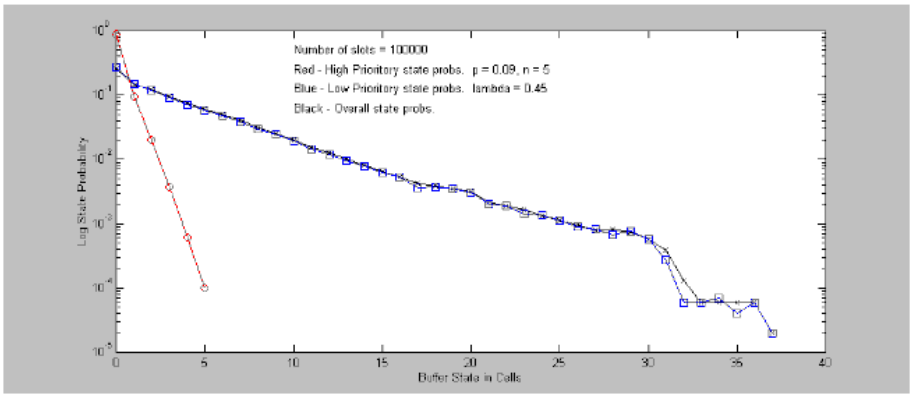


Figure 3 Arrivals, Queue States and Service Channel (p = 0.09, λ = 0.45)





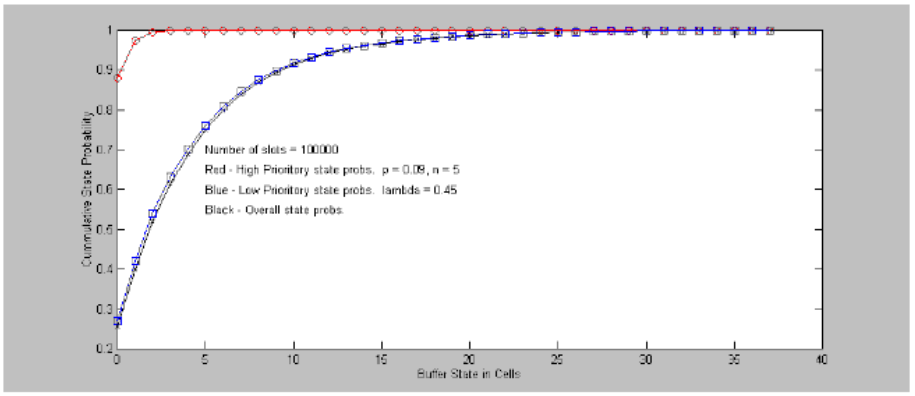
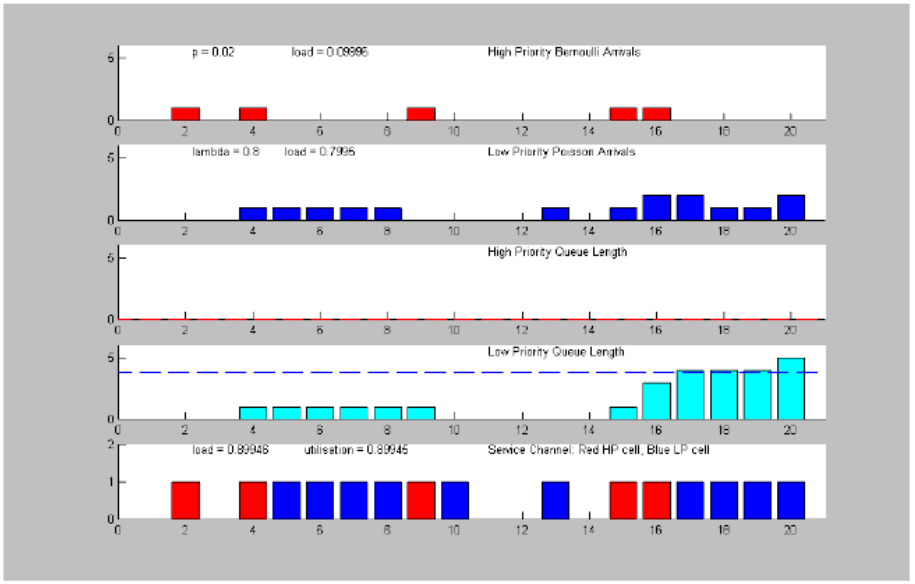


Figure 4 Queue State Probabilities ($p = 0.09$, $\lambda = 0.45$)



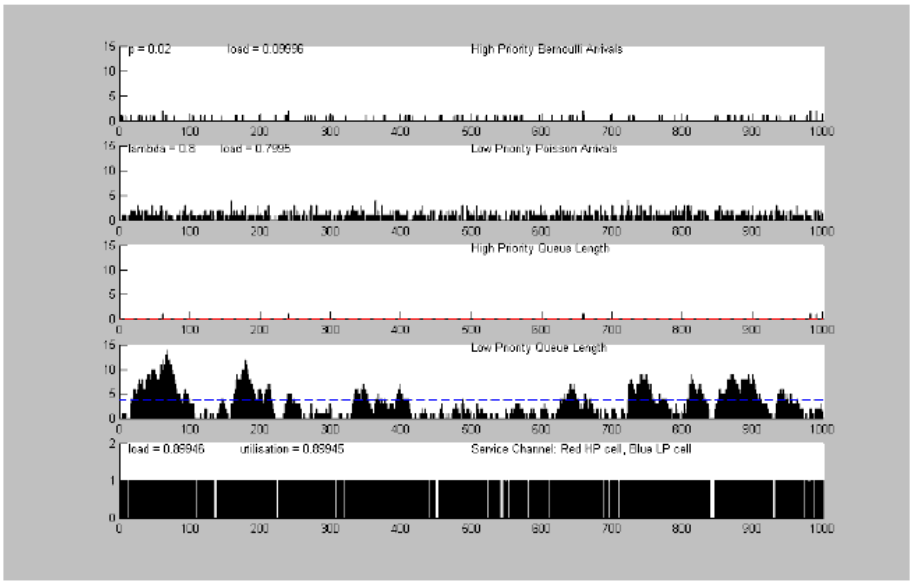


Figure 5 Arrivals, Queue States and Service Channel (p = 0.09, λ = 0.80)

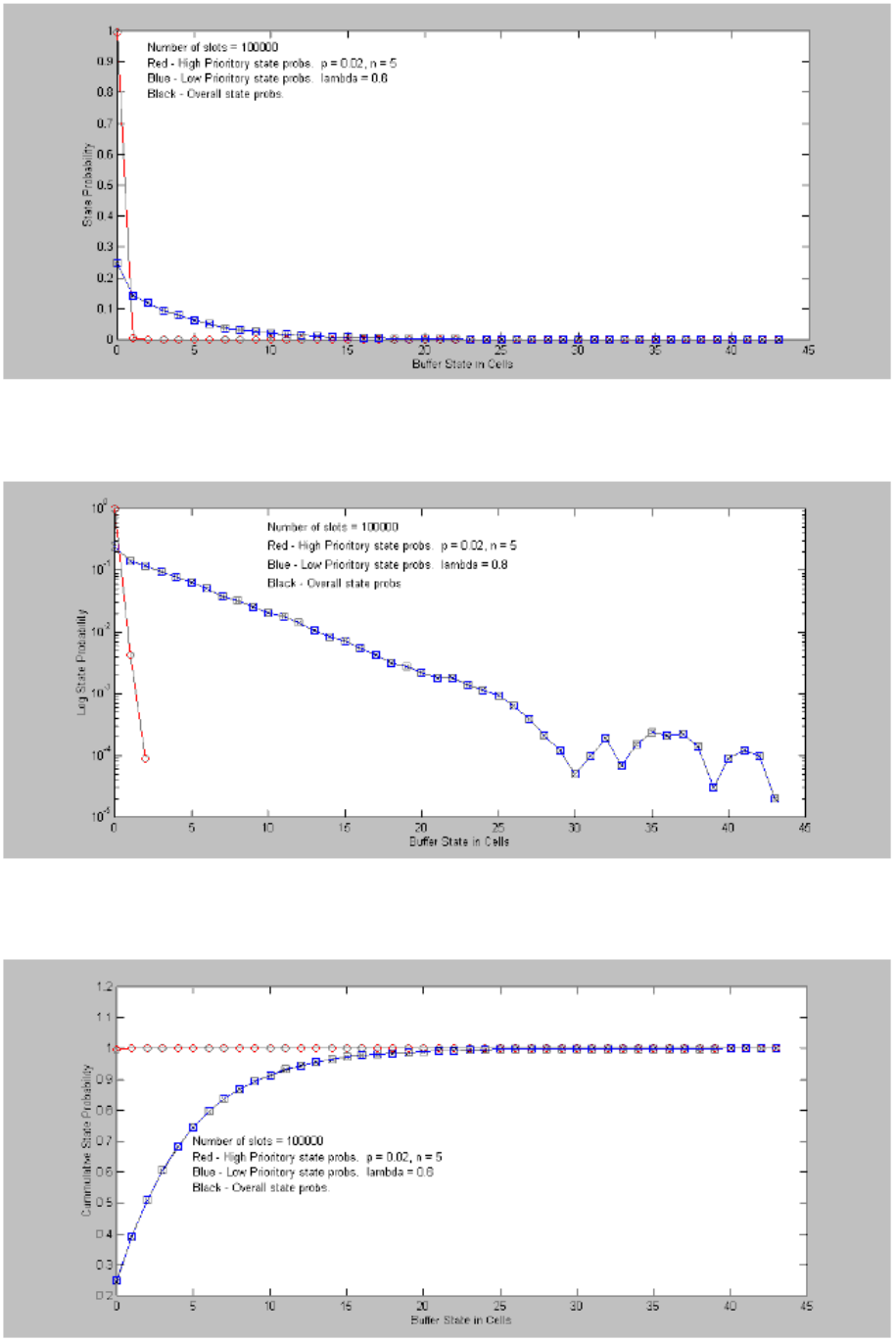
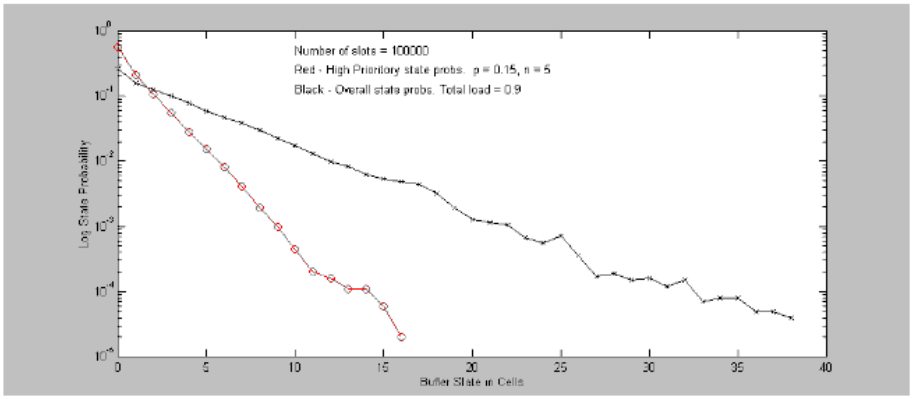
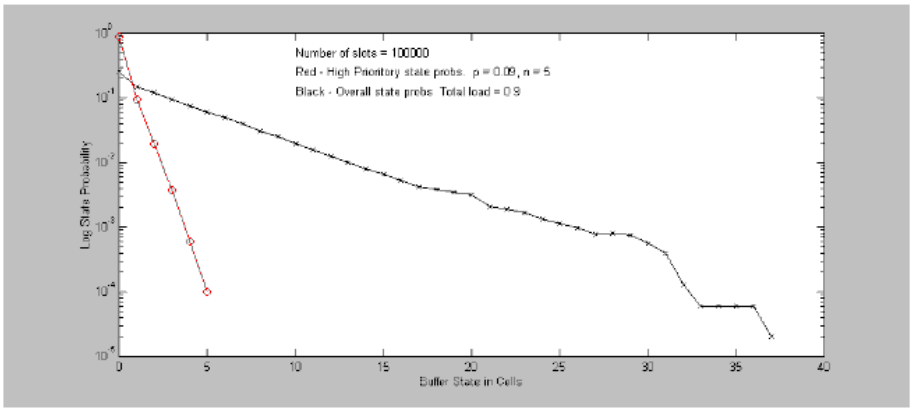


Figure 6 Queue State Probabilities (p = 0.02, λ = 0.80)





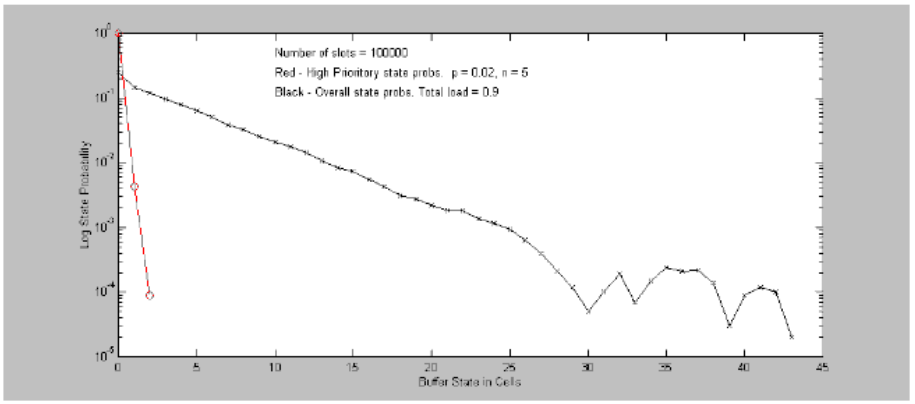


Figure 7 High Priority Queue and Overall State Probabilities

Queue Service Function (q-serve5.m)

```
function [hpq_length,lpq_length,overall_q_length,loading,util,hp_channel,lp_channel] = q_serve5(my_bernoulli_arrivals,my_poisson_arrivals)

%function [hpq_length,lpq_length,overall_q_length,loading,util,hp_channel,lp_channel] = q_serve5(my_bernoulli_arrivals,my_poisson_arrivals)

no_slots=length(my_poisson_arrivals);

hp_channel = zeros(1,no_slots);
lp_channel = zeros(1,no_slots);
hpq_length = zeros(1, no_slots);
lpq_length = zeros(1, no_slots);

previous_hp_q_len = 0;
previous_lp_q_len = 0;


%The actual queue service priority mechanism
for t_slot=1:1:no_slots

   hpq_length(t_slot) = previous_hp_q_len + my_bernoulli_arrivals(t_slot);
   lpq_length(t_slot) = previous_lp_q_len + my_poisson_arrivals(t_slot);

   hp_channel(t_slot) = hpq_length(t_slot) > 0;
   lp_channel(t_slot) = (hpq_length(t_slot) == 0) & (lpq_length(t_slot) > 0);

   lpq_length(t_slot) = lpq_length(t_slot) - ((hpq_length(t_slot) == 0) & (lpq_length(t_slot) > 0));
   hpq_length(t_slot) = hpq_length(t_slot) - (hpq_length(t_slot) > 0);

   previous_hp_q_len = hpq_length(t_slot);
   previous_lp_q_len = lpq_length(t_slot);

end

overall_q_length = hpq_length + lpq_length;

total_arrivals = sum(my_bernoulli_arrivals) + sum(my_poisson_arrivals);

%Sum the two virtual service channels to give the overall service channel occupancy
occupied_slots = sum(hp_channel) + sum(lp_channel);

loading = total_arrivals/no_slots;
util = occupied_slots/no_slots;
```

ATM Prioritorised Queue Simulation Function (ATM-simulate.m)

```
function ATM_simulate(p, no_bernoulli_sources, lambda, no_slots, file_path, file_name, line_type)

%function ATM_simulate(p, no_bernoulli_sources, lambda, no_slots, file_path, file_name, line_type)

sim_run_day_month_year = date;
sim_run_start = clock;

s_time = clock;
bernoulli_arrivals = my_bernoulli_arrivals(p, no_bernoulli_sources, no_slots);
show = ['Bernoulli arrivals generated in ', num2str(etime(clock,s_time)), ' secs']


s_time = clock;
poisson_arrivals = my_poisson_arrivals(lambda, no_slots);
show = ['Poisson arrivals generated in ', num2str(etime(clock, s_time)), ' secs']


s_time = clock;
[hpq, lpq, overallq, loading, utilisation, service_channel_hp, service_channel_lp] = q_serve5(bernoulli_arrivals, poisson_arrivals);

[hpq_state_probs, lpq_state_probs, overall_state_probs, hpq_ave, lpq_ave, overall_ave] = determine_state_probs2(hpq, lpq, overallq);

show = ['Queue state occupancies generated in ', num2str(etime(clock, s_time)), ' secs']


sim_run_end = clock;
sim_process_time = etime(clock, sim_run_start);
bernoulli_prob = p;

clear show s_time blocksize no_blocks

if nargin == 4
   file_path = path;
   file_name = 'no_name';
end

save([file_path, file_name])

%Display the arrival, queue lengths and service channel
graph_state_probs_internal(hpq_state_probs, lpq_state_probs, overall_state_probs, no_slots, bernoulli_prob, no_bernoulli_sources, lambda,
line_type)
```

Poisson Arrivals Generating Function (my-poisson-arrivals.m)

```
function [poisson_arrivals] = my_poisson_arrivals(lambda,no_slots)

%function [poisson_arrivals] = my_poisson_arrivals(lambda,no_slots)
%
%This function generates the cell arrivals from the Poisson source
%over no_slots cell slots, with a mean probability of cell
%arrival lambda in each time slot.
%The function returns a vector of arrivals.

poisson_arrivals=zeros(1,no_slots);

p = rand(1,no_slots);

max_prob = max(p);
max_arrivals = poissinv(max_prob, lambda);

p_boundaries=zeros(max_arrivals+1,no_slots);

for j=1:1:max_arrivals+1
   p_boundaries(j,:)=poisscdf(j-1, lambda);
   f(j,:) = j* (p > p_boundaries(j,:));
end

poisson_arrivals=max(f);
```

Bernoulli Arrivals Generating function (my-bernoulli-arrivals.m)

```
function [bernoulli_arrivals] = my_bernoulli_arrivals(p, no_ber, no_slots)

%function [bernoulli_arrivals] = my_bernoulli_arrivals(p, no_ber, no_slots)
%
%This function generates the cell arrivals from no_ber Bernoulli
%sources over no_slots cell slots, with a probability of cell
%arrival p in each time slot from each source.
%The function returns a vector of arrivals.

z=rand(no_ber, no_slots);
p_vector(1:no_ber,1:no_slots)=p;

arrivals = p_vector > z;

bernoulli_arrivals = sum(arrivals);
```