METHODS FOR TRUNK DIMENSIONING IN MULTISERVICE NETWORKS

By L.T.M. Berry, R.J. Harris, and L.K. Puah.

Abstract

This paper proposes two methods for determining the trunk capacities required on a multiservice network to satisfy the grades of service (call blocking probabilities) for the different offered reference traffic streams used to dimension the network in the planning process. Calls in progress are modelled as On-Off Markovian sources.

Method 1 involves inverting the generalised-Erlang formula on each link to determine the trunk capacity. Method 2 (Loss Equivalent Single-slot method) involves transforming multislot traffic streams to equivalent rough non-random traffic streams. A simulation study demonstrates that the approaches have acceptable accuracy for practical purposes.

1. Introduction

Asynchronous Transfer Mode (ATM) networks are expected to carry a mixture of broadband and narrowband data. Due to the non-homogeneity of the traffic and the difficulty of its characterization, dimensioning a network for its trunk capacities remains a challenge for ATM network planners. Over-provisioning is a conservative but costly solution, while empirical methods tend to be restricted to inflexible conditions. In our approach, we attempt to find a balance between over-provisioning and cost effectiveness.

We present two dimensioning methods to determine the trunk capacities to be allocated to satisfy a user specified call grade of service (GOS) standard. Both methods use the equivalent bandwidth\(^1\) approach that transforms the problem to a multi-slot dimensioning problem. Method one involves inverting the Kaufman-Roberts' Generalised Erlang Formula\(^2,3\) to find the required capacity of each trunk. Method two (Loss Equivalent Single-slot method\(^4\)) involves transforming the multi-slot traffic into equivalent non-random traffic streams. The Chain Flow Method\(^5\) for single slot dimensioning can then be used to dimension the whole network for end-to-end grades of service.

2. Traffic streams

Each traffic stream is defined as a single class of traffic between an origin and destination (OD) pair. We model the stream arrival processes at the call level as Poisson processes. Each stream \(i\) is defined by its call arrival rate \((\lambda_i)\), call service rate \((\mu_i)\), peak rate \((R_{\text{peak}})\), mean rate \((m_i)\) and average burst length \((b_i)\). The last three parameters defining the On-Off bursty cell arrival process within a call.

In this paper, we consider that each stream is assigned to a fixed route; a route is simply a collection of links traversed from the origin to the destination. The call will only be accepted if there are capacities left to establish a virtual channel (VC) between the origin and destination nodes, i.e. all the links on the route have at least the minimum capacity to accommodate the call.

Each call, once accepted, will generate cells at a peak rate \((R_{\text{peak}})\) during an active (On) period and no cells during an inactive (Off) period. The times in the On and Off states are exponentially distributed.

By modelling the traffic as an On-Off process, the method of Guerin et al\(^1\) is then used to determine the equivalent bandwidth of each traffic stream. In our study, the equivalent bandwidth is expressed as a multiple of 64 Kbps ATM slots. (The granularity of 64Kbps can be altered if desired.)

\[
\begin{align*}
\lambda_1, \mu_1, R_{\text{peak}1}, m_1, b_1 \\
\lambda_2, \mu_2, R_{\text{peak}2}, m_2, b_2
\end{align*}
\]

Figure 1: 2 streams of traffic with origin N1 and destination N3

3. Network model

The network is simply a collection of links connecting all the nodes together. A sample network is shown in figure 2. This sample network will be used to demonstrate the dimensioning process later in this paper.

Virtual channels (routes) are defined by a chain of links. For example, origin destination pair (N9, N8) would have links 8, 4, 5 and 7 as its chain while OD pair (N9, N3) would have links 8 and 2 as its chain.

Virtual path (VP) dimensioning requires a simple process of representing the virtual paths as links on a multilink network. For example, a connection between 2 nodes consisting of N virtual paths can be represented by N links. Both dimensioning methods are applicable to such multilink networks.
4. Dimensioning Method

4.1 Method 1 – Inverse Kaufman-Roberts formula

4.1.1 Converting On-Off parameters to equivalent stream slot sizes

Taking into consideration the buffer size, x, and the On-Off parameters, Rpeak, m and b, an equivalent bandwidth is estimated to guarantee maximum cell loss probability of \( \varepsilon \). The streams can then be treated as a constant bit-rate (CBR) streams with rate equal to their equivalent bandwidths.

The equivalent bandwidth formula is described by Guerin et al.\(^1\).

\[
\text{Equivalent bandwidth} = \min(C_s, C_f)
\]

Where

\[
C_f = m + \alpha' \sigma \\
C_s = m + \alpha' \sigma \\
\sigma^2 = m(R_{\text{peak}} - m) \\
\alpha' = \sqrt{-2\ln(\varepsilon) - \ln(2\pi)} \\
\sigma^2 = \sum_{i=1}^{N} \sigma_i^2 \\
\varepsilon = \text{cell loss probability (dimensionless)} \\
x = \text{buffer size (bits)} \\
b = \text{burst length (sec)} \\
R_{\text{peak}} = \text{peak bit rate (Kbps)} \\
m = \text{mean bit rate (Kbps)} \\
\rho = \frac{m}{R_{\text{peak}}} = \text{utilisation}
\]

In our approach, the equivalent bandwidth is further rounded up to give an integer multiple of 64 Kbps. For stream \( i \) this value is denoted by \( c_i \).

As an example, a video source can be modelled using the following parameters:

- \( R_{\text{peak}} = 1.5 \text{Mbps} \)
- Mean = 1.0 Mbps
- Burst = 2 sec
- Cell Loss Probability = 10\(^{-9}\)
- Buffer size = 10\(^6\) bits

Equivalent bandwidth = 1.48 Mbps = 24 x 64 kbps

Therefore the slot capacity, \( c = 24 \) slots

\[
\lambda_1, \mu_1, R_{\text{peak}1}, m_1, b_1 \\
\lambda_2, \mu_2, R_{\text{peak}2}, m_2, b_2
\]

\[
\lambda_1, \mu_1, c_1 \\
\lambda_2, \mu_2, c_2
\]

Figure 3: Multirate to multislot transformation

4.1.2 Invert generalised Erlang recursion to determine the link capacities

The streams are now described by three parameters – call arrival rate (\( \lambda \)), call service rate (\( \mu \)) and slot size (\( c \)). The network dimensioning thus becomes a multislot problem.

Link by link, the inverse generalised Erlang formula is used to determine the minimum bandwidth required for a particular link GOS.

The algorithm can be summarised as follows:

For all links,

i. Find all traffic streams that utilise the link.

ii. Estimate the maximum allowed grade of service of the link (alternatively, an average traffic-weighted GOS could be specified).

iii. Use the inverse Generalised Erlang formula to find the link capacity.

**Estimating the link GOS**

In step ii of the algorithm, the link GOS has to be estimated. The end-to-end GOS for a stream will depend on the blocking probability on each of the links or VPs along its chain.

Two approaches were considered:

1. **LGOS1** - We chose the simple method of applying the minimum end-to-end call GOS for
all streams using the link. This method will be shown to be appropriate in averaging out the link blocking and producing reasonable network congestions.

2. LGOS2 - An alternative method is to reduce the link GOS to a conservatively safe level. Instead of just taking the minimum of all call grades of service, the call grades of service are first divided by the hop count of the stream. The hop count is simply the number of links in the OD connection. Then the minimum of these values for all streams using the VP is chosen as the link GOS. This has the effect of guaranteeing an end-to-end GOS for all the streams.

Inverse Generalised Erlang Formula

The generalised Erlang recursion formula gives the blocking probability of each class given the traffic stream parameters - call arrival rate ($\lambda_i$), call service rate ($\mu_i$) and slot size ($c_i$) – and the total trunk capacity, $C$, to be shared.

A modification of the generalised Erlang recursion is made to find the trunk capacity given a certain GOS criterion. The inverse generalised Erlang program recursively increases the trunk capacity until the GOS criteria is met.

One of 2 kinds of criteria can be chosen as the GOS. One is to aim for a maximum GOS of the stream with the maximum slot size. The stream with the maximum slot size always experiences the largest blocking probability. The second is to aim for an average GOS for all streams. Either of these criteria may be used with the inverse generalised Erlang method.

The capacity at which the recursion stops is the minimum capacity needed to satisfy the criteria.

4.2 Method 2 – Loss Equivalent Single-slot Transformation

4.2.1 Converting On-Off parameters to equivalent slot sizes

This process is the same as 4.1.1.

4.2.2 Converting multislot traffic to single slot traffic

We have a multislot problem with stream $i$ parameters: call arrival rate ($\lambda_i$), call service rate ($\mu_i$) and slot size ($c_i$). For each OD pair, we convert its stream parameters to an equivalent mean ($M$) and variance ($V$). The Loss Equivalent Single-slot method assumes that for multislot traffic streams on a VC, there is an equivalent non-random single slot traffic (with mean $M$ and variance $V$) which behaves similarly with regard to changes in losses as the trunk capacity is altered.

![Figure 4: Equivalent Single-slot Traffic](image)

Figure 4 shows that for the same trunk capacity $n$ and mean lost traffic, $m$, the traffic streams ($\lambda_i$, $\mu_i$, $c_i$) can be transformed to an equivalent single-slot traffic stream with mean $M$ and variance $V$. Theoretically, $n$ can be any number greater than max {$c_i$}. However, the assumption that this equivalence holds for all $n$ is an approximation. It has been shown that the losses obtained, as $n$ varies for the two characterisations of the traffic streams are very similar; we exploit this insensitivity to varying $n$. The inverse generalised Erlang formula is used to determine the $n$ to be used for the traffic parameter transformation.

Computing $M$ and $V$

Firstly, we need to find $n$ as mentioned in the previous paragraph. The mean lost traffic, $m$ can therefore be found by the generalised Erlang recursion.

We know that there is an equivalent $M$ and $V$, that when offered to the same number of circuits will result in the same lost traffic, $m$. $M$ is simply the addition of all $\lambda_i c_i / \mu_i$ for the streams belonging to an OD pair.

i.e. $M = \sum (\lambda_i c_i / \mu_i)$. To find $V$, we solve the Equivalent Random Equations below.

\[
M = A \cdot E(N(A)) \quad (1)
\]

\[
N = \frac{A}{(M + V / M)} - M - 1 \quad (2)
\]

\[
m = A \cdot E(V + \frac{1}{A}) \quad (3)
\]
where
\[ E(A) = \frac{A^n}{N!} / \left(1 + A^1 / 1! + A^2 / 2! + \ldots + A^n / n!\right) \]

\( A \) - equivalent random hypothetical load
\( N \) - equivalent random hypothetical trunks

First, \( A \) and \( N \) are found using an iterative method from equations (1) and (3). Then from equation 2, \( V \) can be obtained.

4.2.3 Dimensioning using the Chain Flow Model

With the transformation of multislot traffic to single slot traffic, we can then use the Chain Flow Model to dimension the multislot network.

5. Dimensioning example

Consider the 9-node network shown in figure 2 with 3 classes of service to be carried. Without trunk reservation, all services have equal access to resources (no priority, complete sharing).

The service data is as follows:

<table>
<thead>
<tr>
<th>Service</th>
<th>Voice</th>
<th>Video</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak rate, ( R_{peak} )</td>
<td>64 Kbps</td>
<td>1.5 Mbps</td>
<td>288 Kbps</td>
</tr>
<tr>
<td>Mean, ( m )</td>
<td>64 Kbps</td>
<td>1.0 Mbps</td>
<td>5 Kbps</td>
</tr>
<tr>
<td>Burst period, ( b )</td>
<td>8 sec</td>
<td>2 sec</td>
<td>2 sec</td>
</tr>
<tr>
<td>Service rate, ( \mu )</td>
<td>0.333 min(^{-1})</td>
<td>0.01667 min(^{-1})</td>
<td>50.0 min(^{-1})</td>
</tr>
</tbody>
</table>

Table 1: Parameters for the 3 classes of traffic

<table>
<thead>
<tr>
<th>Origin</th>
<th>Dest</th>
<th>Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>N1</td>
<td>N2</td>
<td>1</td>
</tr>
<tr>
<td>N1</td>
<td>N3</td>
<td>1</td>
</tr>
<tr>
<td>N1</td>
<td>N4</td>
<td>3</td>
</tr>
<tr>
<td>N1</td>
<td>N5</td>
<td>1</td>
</tr>
<tr>
<td>N1</td>
<td>N6</td>
<td>1</td>
</tr>
<tr>
<td>N1</td>
<td>N7</td>
<td>3</td>
</tr>
<tr>
<td>N1</td>
<td>N8</td>
<td>1</td>
</tr>
<tr>
<td>N1</td>
<td>N9</td>
<td>8</td>
</tr>
<tr>
<td>N2</td>
<td>N3</td>
<td>2</td>
</tr>
<tr>
<td>N2</td>
<td>N4</td>
<td>1</td>
</tr>
<tr>
<td>N2</td>
<td>N5</td>
<td>4</td>
</tr>
<tr>
<td>N2</td>
<td>N6</td>
<td>4</td>
</tr>
<tr>
<td>N2</td>
<td>N7</td>
<td>9</td>
</tr>
<tr>
<td>N2</td>
<td>N8</td>
<td>4</td>
</tr>
<tr>
<td>N2</td>
<td>N9</td>
<td>8</td>
</tr>
<tr>
<td>N3</td>
<td>N4</td>
<td>2</td>
</tr>
<tr>
<td>N3</td>
<td>N5</td>
<td>2</td>
</tr>
<tr>
<td>N3</td>
<td>N6</td>
<td>2</td>
</tr>
<tr>
<td>N3</td>
<td>N7</td>
<td>2</td>
</tr>
<tr>
<td>N3</td>
<td>N8</td>
<td>2</td>
</tr>
<tr>
<td>N3</td>
<td>N9</td>
<td>2</td>
</tr>
<tr>
<td>N4</td>
<td>N5</td>
<td>3</td>
</tr>
<tr>
<td>N4</td>
<td>N6</td>
<td>3</td>
</tr>
<tr>
<td>N4</td>
<td>N7</td>
<td>6</td>
</tr>
<tr>
<td>N4</td>
<td>N8</td>
<td>3</td>
</tr>
<tr>
<td>N4</td>
<td>N9</td>
<td>3</td>
</tr>
<tr>
<td>N5</td>
<td>N6</td>
<td>5</td>
</tr>
<tr>
<td>N5</td>
<td>N7</td>
<td>4</td>
</tr>
<tr>
<td>N5</td>
<td>N8</td>
<td>5</td>
</tr>
<tr>
<td>N5</td>
<td>N9</td>
<td>4</td>
</tr>
<tr>
<td>N6</td>
<td>N7</td>
<td>5</td>
</tr>
<tr>
<td>N6</td>
<td>N8</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 2: Arrival rates (calls/min) for each OD pair.
Results of dimensioning

We applied the dimensioning algorithm for the network above given the conditions:

Buffer size = 1 M bit, for all links.
Cell loss probability = $10^{-9}$ for all streams.
Call GOS needed = 0.01 for all streams.

<table>
<thead>
<tr>
<th>Inverse Generalised Erlang</th>
<th>Loss Equivalent Single Slot</th>
</tr>
</thead>
<tbody>
<tr>
<td>LGOS1</td>
<td>LGOS2</td>
</tr>
<tr>
<td>Link 1 (1,2) : 154006</td>
<td>156723</td>
</tr>
<tr>
<td>Link 2 (2,3) : 115161</td>
<td>117133</td>
</tr>
<tr>
<td>Link 3 (1,4) : 86393</td>
<td>88194</td>
</tr>
<tr>
<td>Link 4 (2,5) : 227519</td>
<td>231130</td>
</tr>
<tr>
<td>Link 5 (5,6) : 158111</td>
<td>160882</td>
</tr>
<tr>
<td>Link 6 (4,7) : 53032</td>
<td>53981</td>
</tr>
<tr>
<td>Link 7 (6,8) : 103246</td>
<td>105293</td>
</tr>
<tr>
<td>Link 8 (2,9) : 77570</td>
<td>79061</td>
</tr>
<tr>
<td>Link 9 (2,7) : 112198</td>
<td>114134</td>
</tr>
<tr>
<td>total slots : 1087236</td>
<td>1106531</td>
</tr>
</tbody>
</table>

Table 4: Results of dimensioning (Sizes given in multiples of 64 kbps slots)

For the Inverse Generalised Erlang method, results from both approaches to estimate the link GOS are tabulated for comparison. The approaches are referred to as the LGOS1 and LGOS2, as explained previously.

The LGOS1 and the Loss Equivalent Single-slot method show excellent consistency with each other. There is only a 0.14% difference in the total number of slots.

Method LGOS2 aims at guaranteeing all end-to-end GOS for all of the traffic streams. The link grades of service used for dimensioning are extremely conservative values. As expected, method LGOS2 requires larger trunk capacity than the other 2 methods. Compared to LGOS1, it requires extra 19295 slots or a 1.8% increase in total capacity.

6. Simulation of the network

A multislot simulator was used to verify the validity of the dimensioning results.

For all these cases, there were 5 runs of 300,000 calls after warmup periods of 5000 calls each. The average link grades of service are shown below:

<table>
<thead>
<tr>
<th>Inverse Generalised Erlang</th>
<th>Loss Equivalent Single Slot</th>
</tr>
</thead>
<tbody>
<tr>
<td>LGOS1</td>
<td>LGOS2</td>
</tr>
<tr>
<td>Link 1 (1,2) : 0.0061</td>
<td>0.0013</td>
</tr>
<tr>
<td>Link 2 (2,3) : 0.0079</td>
<td>0.0021</td>
</tr>
<tr>
<td>Link 3 (1,4) : 0.0079</td>
<td>0.0013</td>
</tr>
<tr>
<td>Link 4 (2,5) : 0.0060</td>
<td>0.0018</td>
</tr>
<tr>
<td>Link 5 (5,6) : 0.0055</td>
<td>0.0014</td>
</tr>
<tr>
<td>Link 6 (4,7) : 0.0033</td>
<td>0.0006</td>
</tr>
<tr>
<td>Link 7 (6,8) : 0.0043</td>
<td>0.0007</td>
</tr>
<tr>
<td>Link 8 (2,9) : 0.0071</td>
<td>0.0026</td>
</tr>
<tr>
<td>Link 9 (2,7) : 0.0052</td>
<td>0.0015</td>
</tr>
</tbody>
</table>

Table 4: Mean link GOS determined by the simulator

6.1 Link probabilities

Table 4 shows reasonable results. We specified a 0.01 maximum end-to-end GOS, thus we would expect each link to have a better GOS than 0.01. However, the effect on the link GOS on the end-to-end grades of service is the important issue and will be examined in the next section.

6.2 End-to-end Stream blocking for the inverse generalised Erlang scheme

6.2.1 LGOS1

The load-weighted average network congestion is $(0.0071 \pm 0.0009)$. This is a reasonable result as 0.01 is expected to be the average congestion.

Network Load-weighted GOS = \[
\frac{\sum_k M_k \cdot B_k}{\sum_k M_k}
\]

where $M_k$ is the load offered and $B_k$ is the GOS, for OD pair $k$.

Due to the complete sharing scheme on VP’s and absence of trunk reservation, we expect the blocking of each class to be approximately proportional to its slot size. In our example, the slot size used for voice, data and video are 1, 4 and 24 slots respectively.

As expected, the simulator showed that none of the voice and data streams exceeded the GOS of 0.01.

There are a total of 36 video streams and the simulator showed that seven streams exceeded the
GOS of 0.01, while the rest were within the specified limit.

The individual call grades of service for these seven streams were: (0.0123±0.0019), (0.0150±0.0028), (0.0178±0.0023), (0.0157±0.0038), (0.0164±0.0058), (0.0130±0.0028) and (0.0183±0.0061). The hop counts for these streams are 4 hops, 4 hops, 5 hops, 3 hops, 3 hops, 4 hops and 4 hops respectively.

The results clearly show the direct impact of hop count on the stream GOS. The worst performing streams are those with the largest number of hops.

Even though the load-weighted average network congestion and link congestions are within limits, the individual end-to-end stream blocking probabilities are not always acceptable. The links were dimensioned for the streams with the largest slot size to have a GOS (max GOS) less than 0.01. The inherent weakness of choosing the call GOS as the link GOS is apparent in this case.

6.2.2 LGOS2

The load-weighted average network congestion for this case is (0.0017±0.0006). The low network congestion results from the low values of the link GOS in this scheme.

As expect, none of the stream end-to-end grades of service exceed the specified value of 0.01. Simulation shows that the worst end-to-end grade of service is (0.0045±0.0026).

The extremely good performance despite the 1.8% increment in trunk capacities over LGOS1 makes this scheme attractive to network planners.

6.3 Stream blocking for Loss Equivalent Single-slot scheme

The network congestion for this case was (0.0071±0.0017), which is close to the result presented for LGOS1.

Similarly, no voice or data streams exceeded the prescribed GOS. While out of the 36 video streams, only two exceeded the GOS of 0.01 by relatively small amounts. The individual call grades of service for these two streams were (0.0170±0.0057) and (0.0183±0.0077). Their hop counts are 3 hops and 5 hops respectively.

As in the case of the LGOS1 scheme, hop count is a major factor in causing the higher blocking values. However, it appears that this scheme does perform somewhat better than the LGOS1 scheme.

7. Conclusion

Both the LGOS1 method and Loss Equivalent Single-slot method are fairly consistent with each other. In terms of stream blocking, the Loss Equivalent Single-slot approach seems to perform slightly better than the LGOS1 method. The LGOS2 approach is extremely effective in guaranteeing end-to-end grades of service, at a cost of increasing the total trunk capacity. All methods show the inherent weakness of adopting complete sharing without trunk reservation on multiservice networks. The unequalised blocking means that some streams perform worse than others. Our studies lead us to conclude that the methods have merit as a basis for practical dimensioning.

8. Acknowledgement

The authors wish to express their appreciation to the Research and Development Division of Telekom Malaysia Sdn Bhd and Australian Research Council (ARC) for supporting this work.

9. References: