TELETRAFFIC AND DATATRAFFIC
in a Period of Change, ITC-13
A. Jensen and V.B. Iversen (Editors)
Elsevier Science Publishers B.V. (North-Holland)
© IAC, 1991

A NEW CALL GAPPING ALGORITHM FOR NETWORK TRAFFIC MANAGEMENT

P M D Turner and P B Key

British Telecom Research Laboratories, Martlesham Heath, Ipswich, IP5 7RE.

This paper examines the performance of Call-Gapping algorithms. Call-gapping is a Network Traffic Management Control that can be applied to individual traffic streams. Such a tool is particularly appropriate for controlling "focussed" overloads, where there is a large amount of traffic directed to an individual number or set of numbers, and which can seriously damage the network's ability to carry traffic. Call-gapping sets an upper limit on the calling-rate of the gapped stream. An ideal algorithm would only block calls when the calling rate exceeded this limit. We show that two existing algorithms do not have this property. We then describe a more-sophisticated algorithm, and examine its performance. It has a performance profile close to the ideal, which means it can be used more widely and safely than existing algorithms.

Main Contribution: This paper describes a new call-gapping algorithm for use in network management.

1. INTRODUCTION

The introduction of digital exchanges and sophisticated routing strategies has led to considerable interest in network traffic management [1,2]. The introduction of new services, and the increasing volatility of demand, only serves to underline the increasing importance of network management.

This paper is concerned with control in a circuit-switched network, although most of the concepts carry through to general networks if control is applied at the "call" level.

There are two main categories of network traffic management control, expansive control and restrictive control. Both can be applied manually, with control via a network management centre, or can be applied automatically, reacting to changes in the network.

In expansive control, spare capacity is used in the network to route around congestion. The congestion may be caused by failure, or by an unusual traffic pattern. Thus this type of control can also be used to maximise the carried traffic in the network under different traffic patterns. In the British Telecom main trunk network one example of this flexibility is planned to be provided by using Dynamic Alternative Routing (DAR) [3,4,5], with multiple parenting of local exchanges onto main switches.

Restrictive controls deliberately reject calls in order to preserve network performance. Their main use is to protect modern digital switches from the effects of overload. This is important, since the throughput of successful calls from a digital switch can deteriorate rapidly under overload. In extreme cases, a switch can spend most of its processing power rejecting call attempts. There is also a related use of such controls, which is to prevent use of the network (including transmission capacity) by calls which have very little chance of being successful, thereby increasing network efficiency.

There are two methods of restrictive control, call gapping and percentage thinning. The latter is a crude control which arbitrarily rejects a certain defined percentage of calls, and which requires constant monitoring and adjustment. Call gapping is a more sophisticated control which limits the rate at which calls are accepted onto the network, and places an upper bound on the acceptance rate [6,7].

In section 2 of the paper we review existing call gapping algorithms, and highlight some of their disadvantages. In section 3 we present a new call-gapping algorithm, [8], based on looking at average behaviour which does not suffer from the shortcomings of existing algorithms, and describe the performance of this new algorithm. Section 4 looks at the network implications of call-gapping and briefly touches on some of the problems of setting gapping intervals.

2. EXISTING CALL-GAPPING ALGORITHMS AND THEIR USE

Call-gapping is a restrictive network traffic management control [6,7]. Its main application is to reduce the adverse effects of focussed overloads. Focussed overloads occur when there is a massive traffic surge directed toward a particular destination or set of destinations. Such overloads often follow the publication of a telephone number or numbers on the media, which might follow a disaster, or which might be for a phone-in, competition or tele-voting exercise. In these circumstances the affected numbers can receive many thousands more calls than they can handle. Thus processor and transmission capacity is used ineffectively, and if the overload is large enough these ineffective calls can prevent the successful completion of calls to other destinations. It is therefore desirable to reject most of the calls offered to the affected numbers. Call-gapping sets a limit on the rate at which calls are accepted — one per gapping interval, where we choose the gapping interval to give the desired performance.

In this section we comment on the characteristics of an ideal call-gapping algorithm, describe two existing algorithms and comment on their performance.

An *ideal* algorithm would only reject calls when the calling rate exceeds one call per interval, in which case the calling rate would be reduced to exactly one. In other words if we define λ to be the offered calling rate, in calls per gapping interval, and $\phi(\lambda)$ to be the accepted calling rate after gapping (the carried call rate), then ideally

$$\phi(\lambda) = \lambda \qquad \lambda \leq 1 \qquad (1)$$
$$= 1 \qquad \lambda > 1.$$

This needs to be made precise, what we really mean is that φ is the *expected* calling rate, and ideally we would like equation (1) to hold for all arrival processes which are not excessively peaky or excessively batched. We would certainly like Eqn. (1) to hold for Poisson processes of rate λ, and we shall take this as our standard.

2.1. Algorithm Descriptions

There are two types of algorithm in existence. Let T be the gapping interval in seconds then :

Algorithm 1: Time is divided into intervals of duration T, and at most one call is allowed in any one interval.

The second algorithm, the Crawford algorithm [9], is used by AT&T.

Algorithm 2: Once a call is accepted, any call arriving during the subsequent T seconds is lost.

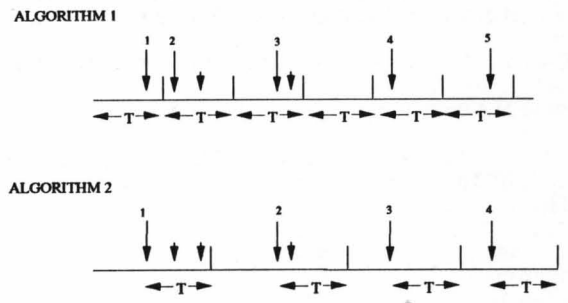

Figure 1: Existing algorithms

The two algorithms are illustrated in Figure 1, where the numbered long arrows refer to accepted calls and the unnumbered short arrows are rejected calls. Note that the boundaries are fixed in algorithm 1 and accepted calls can be quite close together if they are on either side of a time period boundary (for example calls 1 & 2). In Algorithm 2 the interval between calls is always at least the gapping interval. Indeed, we show later that if the same gapping interval is used then Algorithm 2 rejects more calls than Algorithm 1.

2.2. Performance Analysis

We shall assume that the arrivals form a Poisson process of rate λ, in which case we have easily

Algorithm 1.

$$\phi_1(\lambda) = 1 - e^{-\lambda}. \qquad (2)$$

Algorithm 2.

$$\phi_2(\lambda) = \frac{\lambda}{1+\lambda}. \qquad (3)$$

It follows that $\phi_1(\lambda) > \phi_2(\lambda)$ for positive λ, and hence algorithm 2 rejects more calls.

If the same stream is gapped a number of times with the same gapping interval then only the first application would reject any calls. However, when call gapping is applied independently at various different levels in the network the compound effect must be considered. For example if there is a focussed overload on a local exchange then in an extreme case call-gapping could be applied at all intermediate exchanges, which could be four or more times in a hierarchical network. If call-gapping is applied *independently* at s stages in network, with many streams merged at each gapping stage, then the overall performance can be approximated by $\phi^s(\lambda)$, where we define

$$\phi^s(\lambda) = \phi\{\phi^{s-1}(\lambda)\}. \qquad (4)$$

2.3. Results

Expressions (2) and (3) relate the carried calling rate per interval to the offered rate for the two algorithms, and are plotted in Figures 2 & 3. The performance of the new call-gapping algorithm defined below is also given in these figures. Note the poorer performance of the second algorithm.

- Notice how slowly Algorithm 2 converges to a carried rate of one call per interval. However, when call-gapping is used to control a massive focussed overload this does not present a serious problem.

- Further, notice the high rejection rate of 50% at an offered rate of one call per interval. For the first algorithm this is approximately 37%. Therefore it is necessary to deactivate either algorithm when calling rates drop to normal.

For these reasons the present algorithms are not ideal as general tools for the control of calling rates. It is possible to gap on a Route or Destination National Number Group (NNG) basis, however with the present gapping methods this would be of doubtful utility because of the danger of blocking a significant number of potentially successful calls.

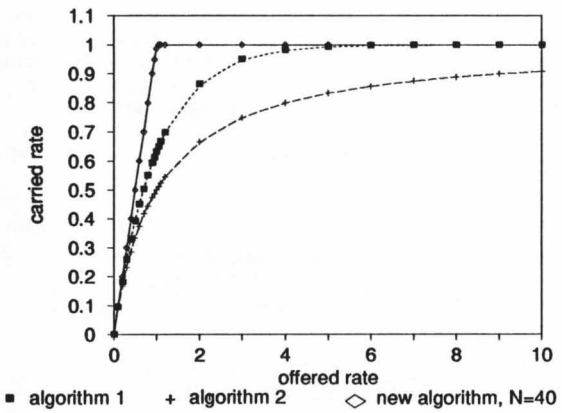

Figure 2: Performance of algorithms

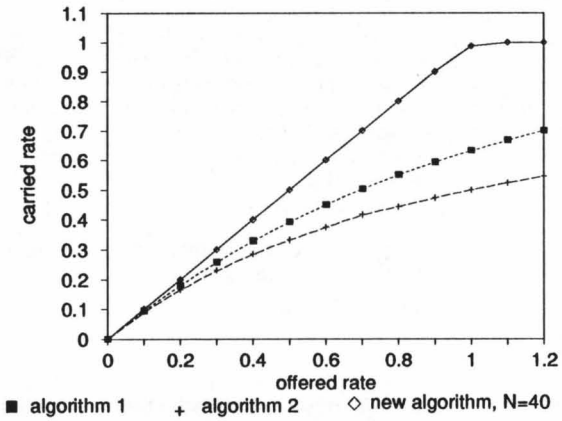

Figure 3: Performance of algorithms, low offered traffic

For repeated applications of algorithm 2 we have

$$\phi_2^s(\lambda) = \frac{\lambda}{s\lambda + 1} \qquad (5)$$

thus if we have the algorithm applied independently s times to an initial calling rate of 1, then carried calling rate could be as low as $1/(1+s)$, although in practice the end rate is more likely to be between 1/2 and 1/3.

3. AN ALTERNATIVE ALGORITHM

In this section we describe and analyse a new algorithm, [8], that has a performance profile close to the ideal.

The two algorithms described in the previous section ensure that the carried calling rate — $\phi(\lambda)$ — never exceeds one. The first algorithm has the property that at most one call is accepted in each T second interval, whilst the second algorithm ensures that all accepted calls are at least T seconds apart. This means that not only is the expected, or long-term calling rate less than one, but also the carried calling rate over any finite number of gapping intervals never exceeds one.

To obtain a better performance, we have to allow extra calls to be carried after short-term downward fluctuations in the offered calling rate. In other words to obtain a performance based on the average or expected number of calls accepted, we have to monitor the average behaviour, and not just individual intervals.

3.1. The new algorithm

We divide time into equal time intervals (of length T). The essential idea is to have an allowance of calls (usually one) for each interval and permit any unused allowance (subject to a given limit) to be carried forward to subsequent intervals.

This is implemented by use of an integer counter that is incremented by the given allowance at the start of each interval. This counter contains the accumulated allowance of calls — subject to a given limit. Each arriving call causes this counter to be decremented by one until it stands at zero. Subsequent calls are rejected until the counter is incremented at the start of the next time interval.

The algorithm works as follows,

(i) At the beginning of each time interval the *COUNTER* is incremented by a constant integer n, subject to a maximum value N for the *COUNTER*.

(ii) When a call arrives the counter is examined. Then

  (a) if *COUNTER* = 0, the call is rejected.

  (b) If *COUNTER* > 0, the call is accepted and the *COUNTER* is decremeneted by 1.

Thus with the notation

t	interval number (integer)
m	size of *COUNTER* increment
N	maximum size of *COUNTER*
$COUNTER_t$	value of counter at the start of time interval t, *after* it has been incremented
X_t	number of call arrivals in $(t-1, t]$

then

$$COUNTER_t = \min\{N, m + \max(COUNTER_{t-1} - X_t, 0)\} \qquad (6)$$

and $\min(COUNTER_{t-1}, X_t)$ is the number of accepted calls in the interval $(t-1, t]$.

At high calling rates (many calls per gapping interval) this behaves like algorithm 1. At low rates the counter quickly rises towards the maximum value *N* ensuring that very few calls are lost at all. The algorithm ensures that over *N* intervals the carried calling rate cannot exceed 1, although in extreme circumstances it is possible for there to be *N* calls in one interval. This can only happen when a surge of calls arrives after a number of quiet periods.

It is possible to generalise the algorithm to prevent this but at the expense of greater complexity. This is done by setting an extra bound *M* on the number of calls allowed in an interval, which requires an additional "Local Counter". If the original counter contains a number larger than *M* at the start of a time interval this extra limitation is active.

3.2. Performance analysis

3.2.1. Steady state behaviour

The time dependent behaviour can be deduced from Eqn. (6), given that we can calculate X_t. We consider first the steady state behaviour. Let us introduce some further notation

C_n probability that the counter contains the number *n* at the start of interval (after incrementing the counter)
P_n probability of *n* arrivals in an interval
Q_n probability of *n* or more arrivals in an interval

therefore $Q_n = \sum_n^{\infty} P_i = 1 - \sum_{i=0}^{n-1} P_i$, $\lambda = \sum i P_i$, and $C_i = 0$ for $i<m$ or $i>N$, where we have assumed that the number of arrivals in an interval is independent of time, and of what happened in the previous intervals.

For Poisson arrival with rate λ

$$P_n = \frac{\lambda^n}{n!} e^{-\lambda}. \tag{7}$$

Under these assumptions the *COUNTER* values form a Markov chain, defined on *m* ... *N*, with transition probabilities p_{ij} given by

$$p_{im} = Q_i \tag{8}$$

$$p_{ij} = P_{i+m-j} \qquad m < j < N$$

$$p_{iN} = \sum_{j=0}^{i+m-N} P_j \qquad N - m < i$$

where some of the p_{ij} are zero, since P_n is only defined for non-negative *n*.

The expected number of calls accepted in state *n*, r_n, is given by

$$r_n = iQ_i + \sum_{j=1}^{n-1} jP_j. \tag{9}$$

and the expected arrival rate is $\sum C_n r_n$. We can regard the accepted call process as a Markov reward process, where we receive unit reward every time we accept a call, or equivalently interpret the call-gapping algorithm as a policy within the context of Markov decision theory. In this case we can write down Howard's equations [10]

$$\phi(\lambda) + v_i = r_i + \sum_j p_{ij} v_j \qquad i = m \ldots n \tag{10}$$

where the *v*'s are the "relative values", and $v_i - v_j$ can be interpreted as the extra number of calls we accept over an infinite horizon if we start with the counter in state *i* rather than *j*. It is clear that $v_i - v_j \to i - j$ as the arrival rate, λ, increases.

3.2.2. Time-dependent behaviour

We now turn to the time-dependent behaviour of the algorithm. For instance, if the counter contains 1 at the beginning of the first period we would like to calculate the expected calling rate during the first *t* time intervals. This problem can be solved by using the dynamic programming or "value iteration" recursions of Markov decision theory [11]: if we define $V_n(t)$ to be the expected total number of carried calls over *t remaining* time intervals, given that the current *COUNTER* value is *n*, then the recursions are

$$V_i(t) = r_i + \sum_j p_{ij} V_j(t-1). \tag{11}$$

In which case $V_n(t)/t$ gives the required calling rate. Note that for $\lambda>1$, this quantity will differ from the steady-state calling rate $\phi(\lambda)$ by less than $\{v_n - v_1\}/t$.

3.3. Results

In what follows we have restricted attention to the simplest case, where *m*=1 — examination of the steady state results shows that there is little to be gained by using other values for *m*. The steady state performance of the algorithm, which is derived from equations (7) - (10) depends upon the counter size, as shown below in Table 1. At values of 30 or over for the counter, the algorithm's deviation from perfect behaviour is insignificant outside the offered calling rate range of between 0.9 and 1.1 calls per interval. For this reason the tables concentrate on this range. The graphs in figures 2 and 3 illustrate this well, where the counter size is 40.

Offered rates		.90	.95	1.0	1.05	1.1
Maxmimum counter size	10	.887	.923	.952	.972	.984
	20	.898	.943	.975	.992	.999
	30	.900	.948	.984	.997	1.000
	40	.900	.949	.988	.999	1.000

Table 1: carried calling rates, $\phi(\lambda)$, for different offered rates λ and counter size.

For a value of 40, the algorithm performs worst at λ = 1 where $\phi(1) = 0.988$., see table 2.1. Recall that for the algorithms 1 and 2, the values of φ at $\lambda = 1$ are 0.63 and 0.5. Outside the range λ = .95 to 1.05 the new algorithm deviates from perfect behaviour by less than one call in a thousand. Thus for practical purposes the algorithm's behaviour is very good.

It can be deduced from Figures 2 and 3 that repeated use of the algorithm presents no problems, since it follows so closely the performance of the ideal algorithm (c.f. Eqn. (1)). For a counter of maximum size 40, calculating the effect of repeated applications directly gives $\phi^4(1) = .972$ and even if we use the algorithm 10 times we have $\phi^{10}(1) = .958$, whereas recall for algorithm 2, $\phi_2^4(1) = 0.2$.

Table 1 suggests a value of at least 30 for the counter maximum. With a value of 32, the relative values found from equation, and are illustrated in Table 2, showing the difference in the number of calls carried if we start in with the counter at *j* rather than 1. Tables 3 and 4 shows the time-dependent behaviour of the algorithm if we have a counter maximum and start with a counter value of 1 or 16.

Offered rates		.90	1.0	1.1	5
relative values	v_2-v_1	0.75	0.96	1.00	1.00
	v_8-v_1	3.09	6.06	6.98	7.00
	v_{16}-v_1	3.85	11.13	14.83	15.00
	v_{32}-v_1	4.01	15.34	26.38	31.00

Table 2: Relative values for a counter maximum of 32

Offered rates	.90	.95	1.0	1.05	1.1
steady state	.900	.948	.985	.998	1.000
First 100 periods	.866	.900	.927	.947	.962
First 500 periods	.892	.934	.966	.983	.991
periods 200-500	.898	.943	.975	.992	.998

Table 3: time dependent behaviour: carried rates for counter initialised to 1, (maximum counter size =32).

Offered rates	.90	.95	1.0	1.05	1.1
steady state	.900	.948	.985	.998	1.000
First 100 periods	.900	.948	.994	1.036	1.071
First 500 periods	.900	.947	.987	1.010	1.020
periods 200-500	.900	.947	.985	1.004	1.007

Table 4: time dependent behaviour: carried rates for counter initialised to 16, (maximum counter size =32).

In order to achieve this behaviour we have allowed more than one call in an interval when justified by the counter. If the counter is large then this can allow a surge of calls in one interval. This is probably not a practical problem but it can be avoided by the use of a Local Counter described at the end of Section 3.1.

Sample results are shown for various Local Counter sizes in table 5, where an incremental allowance of one is assumed (m=1) and the Global Counter is 40. Table entries represent the resulting calling rates for the column heading offered rates. Note that the Local Counter = 1 rows give the same results as for Algorithm 1. The last row has Local Counter = Global Counter and these results are identical to those for the simplified algorithm in which there is no Local Counter restriction.

		Offered rates				
Global counter size	*local counter*	.90	.95	1.0	1.05	1.1
40	1	.593	.613	632	.650	.667
	4	.897	.946	.986	.999	1.000
	40	.900	.949	.988	.999	1.000

Table 5: carried calling rates, $\phi(\lambda)$, for different offered rates λ and counter size.

Note that performance with a Local Counter maximum of 4 is only marginally worse than for the simplified algorithm (compare the starred line with that below it in table 5).

4. NETWORK IMPLICATIONS AND CUSTOMER IMPACT.

It is important to stress that the network implications of call gapping and its impact on the service to a gapped customer have not been discussed in this document. These issues will affect how we chose to implement call-gapping, and in particular how we set the gapping interval *T*, which is the primary parameter that we can vary. Work has already been undertaken in this area. Nevertheless it is worthwhile to list some relevant considerations.

(i) Criteria for acceptable service to the gapped customer must be determined. One possibility is a criterion based on the mean time between a call clear down on a customer line to the next call received on the same line. A value of between 5 and 10 seconds may be deemed acceptable for a manned line.

(ii) Suppose call gapping is activated at all originating DMSU's (Digital Main Switching Unit, i.e. Main Network exchange) to limit the total carried calling rate to an overloaded SP (Service Provider). The application of call gapping sets an upper bound to the carried rate at each DMSU. The gapping interval is set at each DMSU so that the total carried rate from all the DMSU's is limited to the desired

level. There is a danger that uneven DMSU loads will lead to the overall total carried rate being too low.

(iii) For the reasons discussed in (i) & (ii) it is desirable to allow quite high calling rates to the gapped subscriber (perhaps several times the calling rate required to satisfy the criterion in (i) above). Here we are limited by the call handling capacities of the network nodes. After all the whole point of call gapping is to prevent serious switching overload!

(iv) When call gapping is implemented at many exchanges it may be advisable to randomise the start of the time interval boundaries at different DMSU's. Otherwise it is possible to imagine the production of micro surges of traffic. If this is allowed the effects on switch performance should be checked.

The most obvious application of call gapping is to protect the network from focussed overloads directed at individual numbers. However in extreme weather conditions or disasters the overload may be focussed on a particular area. In these circumstances Route or NNG gapping may be useful. However, interactions with existing processor load controls must be considered. For this reason further study is needed.

5. CONCLUSIONS.

We conclude by noting the limitations of the existing algorithms and then listing the advantages and extended applicability of the new algorithms.

The performance of existing call gapping algorithms is such that they can only be used to control focussed overloads, because they allow too little traffic through at low calling rates. Even this usage would have to be monitored very carefully, taking care to remove the control when calling rates return to more normal values. Further monitoring would be required to ensure that multiple applications of call gapping do not take place. These methods cannot be used to fine-tune calling rates which precludes their use for route or NNG (National Number Group) gapping.

The new algorithms behave differently, and their performance profile is close to the ideal, which enables them to be used more widely:

- When applied as with a Global Counter of 40 (and a Local Counter of 4 if required), monitoring and timely cancellation are not critical. However, further studies are required before permanent gapping can be recommended for individual customers.
- The gapping interval is easy to set and understand. If you want a mean carried rate of one call every *n* seconds then the interval can be set to *n* seconds.
- No "multiple application" problem can occur.
- These methods can be used to fine tune calling rates. It is thus worth considering their use on a route or NNG basis as discussed above.
- Since the new algorithms use a simple method based on counters they should not add much to normal call processing requirements. Nevertheless they would require more processing than the existing algorithms.

REFERENCES.

[1] Ebner, G.C. & Haenschke D.G. Network Management. *Bell System Technical Journal*, Vol.62 No.7 September 1983.

[2] Haenschke D.G., Kettler, D.A. & Oberer, E. DNHR: A New SPC/CCIS Network Management Challenge. *10th International Traffic Congress*, Montreal, CANADA 1983.

[3] Stacey, R.R & Songhurst, D.J. Dynamic Alternative Routing in the British Telecom Trunk Network. *International Switching Symposium*, Phoenix, U.S.A. March 1987.

[4] Gibbens, R.J., Kelly, F.P. & Key, P.B. Dynamic Alternative Routing — Modelling and Behaviour. In *Proceedings of the 12th International Teletraffic Congress, Turin, Italy.* Amsterdam: Elsevier, 1988.

[5] Key, P.B. & Whitehead, M.J. Cost-Effective Use of Networks Employing Dynamic Alternative Routing. In *Proceedings of the 12th International Teletraffic Congress, Turin, Italy.* Amsterdam: Elsevier, 1988.

[6] Haenschke D.G., Kettler, D.A. & Oberer, E. Network Management and Congestion in the U.S. Telecommunications Network, *IEEE Trans. Comm.*, Vol VOM-29, No. 4, pp. 376-385, April 1981.

[7] Tokunaga, H., Kawano, H. & Nakagima, S. Traffic Congestion Control System. *Review of the Electrical Communication Laboratories (NTT)*, 36, No, 2, Tokyo, March 1988.

[8] British Telecom plc, *PCT Application No. GB89/00302, European Patent Application No. 893020807.6* and others.

[9] Crawford, K. *U.S.A Patent No. : 4,224,479*, September 23, 1980.

[10] Ross, S.M. *Applied probability models with optimization applications.* San Francisco, CA: Holden Day, 1970.

[11] Howard, R.A. *Dynamic Programming and Markov Processes.* Wiley, New York, 1960.