

A Unified Global Congestion Control Scheme for Broadband Signalling Networks

K. Kant

Abstract

This paper proposes a unified user-part (UP) congestion control mechanism for reacting to signalling link, level 3 processor, level 4 processor, and trunk (or bandwidth) congestion in a signalling network supporting point to point connection setups. Although this paper concentrates on BISUP (Broadband ISDN User Part) as the application layer, the results apply equally well to ISUP as well. Unlike current congestion control mechanisms, the scheme attempts to provide traffic control as close to the traffic source as possible by propagating the congestion indication backwards along the call path. Such a scheme minimizes processing and transmission resources wasted on unsuccessful call attempts and can improve overload performance dramatically. However, there are number of implementation difficulties in the scheme, which the paper attempts to address. The paper also examines how to detect “bandwidth congestion” and how to choose severity levels of such a congestion.

1 Introduction

Effective network level congestion control is essential for ensuring network integrity and providing acceptable service during periods of overload. As the telecommunication networks evolve in many directions, it is necessary to examine if the existing congestion controls are adequate, and if not, to enhance them appropriately. This study is an attempt in this direction and considers enhancements to congestion control capabilities in the context of ISDN User Part (ISUP) and Broadband ISUP (BISUP) signalling protocols. These enhancements are motivated by a variety of factors that can impact overload performance of the telecommunication network. Some of these factors are: introduction of advanced intelligent network (AIN) services, reduced control over the kind of traffic entering the network due to mediated services and multiplicity of service providers, a variety of traffic types (e.g., voice, compressed video and data), a wide range of connection bandwidths (ranging from a few Kbits/sec to tens of Mbits/sec), a variety of connection types (point-to-point, point-to-multipoint, and perhaps multiconnections), and a variety of signalling protocols that a switch may have to handle (e.g., ISUP, BISUP, PNNI, TCAP, MMAP, IS-41, and perhaps others). In general, these factors

make the network traffic much less predictable and more bursty thereby requiring better congestion controls. Unfortunately, the transition from a single large network to an interconnection of multiple networks makes congestion control even more difficult for reasons of inaccessibility of information and ability to control traffic between networks.

There are a number other factors too that call for improved congestion controls. At least in North America, the current narrowband signalling networks use quasi-associated signalling (i.e., all switches and SCPs communicate via STPs) whereas the broadband networks are expected to use associated mode signalling (i.e., direct signalling between switches and SCPs) initially and perhaps quasi-associated signalling in the long run. Thus, any congestion control scheme must work well for both signalling modes and in the mixed environment. Associated signalling and the changed telecommunication environment also seem to imply that broadband networks will not be as well connected as narrowband networks thereby requiring more intermediate switches for long-distance connections than in the current public network. This implies the need for more global congestion controls (in the sense of going across multiple user part nodes such as switches and SCPs). Current congestion controls in current networks tend to be very local, in that they either do not go across a user-part node (link and SMH congestion controls) or go only to a neighboring user part node (ACC and ACG controls).

This paper proposes a unified user-part (UP) congestion control mechanism for reacting to all four types of congestions on an ISUP or BISUP call path: (1) Signalling link congestion, (2) SMH (i.e., level 3 processor) congestion, (3) Call processor (i.e., level 4 processor) congestion, and (4) Bandwidth (or trunk) congestion. Unlike current congestion control mechanisms, the scheme attempts to provide traffic control as close to the traffic source as possible. This is done by propagating congestion indication backwards along the call path through the user part at each intermediate switch. The mechanism can be viewed as an extension of the current automatic congestion control (ACC) mechanism for propagating call processor congestion. Unlike the current ACC mechanism, the new mechanism handles all four types of congestions mentioned above and is not limited to propagation the indication to the neighbor only. We henceforth call the new mechanism as global congestion control (GCC) scheme.

Propagation of congestion indication and hence the throttling of traffic as close to the source as possible minimizes processing and transmission resources wasted on unsuccessful call attempts and can improve overload performance dramatically. Unfortunately, implementing such a mechanism presents a number of practical difficulties. This paper highlights those problems and suggests solutions. The paper also examines how to detect “bandwidth congestion” and how to choose severity levels of such a congestion. Finally, the paper compares the performance of the new controls against present mode of operation (PMO) with and without ACC controls (although ACC controls have been specified in the standards, they are rarely implemented in current switches).

The outline of the paper is as follows. Section 2 reviews the existing SS7 congestion control schemes. Section 3 describes the congestion indicators used by the proposed GCC mechanism at the application level for all four types of congestions mentioned

above. Section 4 discusses the difficulties in propagating GCC globally and the solutions adopted. It also discusses how the traffic is throttled both locally and globally. Section 5 discusses the results of the study, which compares the performance of GCC against the existing controls. It is shown that GCC can achieve much better performance than existing controls. Finally, section 6 summarizes the work and delineates issues for future study.

2 Existing Congestion Controls

The signalling link congestion detection is done independently for each link in the link-set by monitoring the number of messages that have not yet been successfully transmitted (i.e., not yet acknowledged by the other end). The detection mechanisms assume that the traffic is almost equally divided across all links in the link-set, and hence the congestion on any link in the link-set indicates congestion on the entire link-set. In particular, when multiple congestion severity levels are recognized, the maximum severity level over all links in the link-set represents the current congestion level of the link-set. For each severity level i , three thresholds are specified on the number of unacknowledged messages: abatement (A_i), onset (O_i), and discard (D_i), where $A_i < O_i < D_i$.

Consider a quasi-associated signalling network having user-part (UP) nodes A and B that “home” on a common STP S. Let us examine the impact of congestion along the path AB on node A. For simplicity, let us assume a single severity level. Let L_x denote the number of unacknowledged messages on link-set x . Then, if $L_{AS} > O_{AS}$, i.e., the number of unacknowledged messages on link-set AS exceeds the onset threshold for this link-set, node A starts dropping messages going along the path AB until $L_{AS} \leq A_{AS}$. Similarly, if $L_{SB} > D_{SB}$, the STP S starts dropping messages until $L_{SB} \leq O_{SB}$. However, if $O_{SB} < L_{SB} < D_{SB}$, the STP does not drop any messages. Instead, whenever $L_{SB} > O_{SB}$, the STP sends a TFC (Transfer Controlled) message backwards on the call path to the nearest node running a MTP-user application (e.g., ISUP, BISUP, SCCP, TCAP, etc. to inform it of the congestion. (Actually, a TFC is sent out on every n th message, where by default $n = 8$.) Thus, node A would know about the congestion on segment SB and can act in the same way as if the congestion was on the AS segment.

Three different options are provided in terms of precise details of congestion detection and its use in traffic throttling. One aspect in these options is the “congestion priority” of the messages. Each message carries a “priority”, which is a number in the range 0-3, and indicates its relative importance. The priorities are used only for message discarding (and not for scheduling purposes). The three options are as follows[3]

1. International Option, i.e., no congestion priorities, only one congestion level, and no traffic discarding at MTP3 level. The congestion indication is passed to the APPL level which assigns congestion severity using timers T29 and T30, as discussed below. Each severity level is associated with a throttling percentage, with higher severity resulting in higher throttling. This throttling applies only to new

calls (i.e., IAMs). Messages associated with existing calls are not throttled.

2. National Option without Congestion Priorities, which is same as the international option except that multiple congestion severity levels are recognized. These are again passed on to the APPL level by the MTP3 layer.
3. National Option with Congestion Priorities, where the messages are given congestion priorities in the range 0-3 and 3 congestion severity levels are recognized. At congestion severity level of i , MTP3 discards messages with congestion priority $\leq i$. This option uses $n = 1$ for link congestion. Since this option is used in the US [1], we shall primarily concentrate on it in this study.

In the international option, the mechanism used for assigning congestion severity level is as follows: Suppose that the APPL level receives the first indication of MTP congestion. At this time timers T29 and T30 are started and the APPL level congestion indicator is set to 1. Further congestion indications while T29 is running are ignored. Any congestion indications after T29 expires (but before T30 expires) results in further increase in the congestion indicator and restarting of T29 and T30. Finally, if T30 expires, the congestion indicator is decreased by 1 (and T30 is restarted if the congestion level is still nonzero). Setting of appropriate throttling percentages is left to implementation and will not be addressed here.

Although the above mechanism does not place any upper bound on the congestion severity level, one could enforce an upper bound, say L_{\max} , explicitly. That is, if the severity level is already at L_{\max} and a congestion indication is received, the severity level is not incremented further, but T29 is started and T30 is re-started as usual. This change will have little impact on performance provided that the throttling percentage at L_{\max} is very high.

The reception of a TFC can only raise the congestion severity level. In the first two options, the decrementing of severity level is considered to be a UP functionality and is not addressed at MTP3 level. Note that the T29/T30 mechanism described above decrements the severity level by itself; therefore, the international option does not need anything beyond the TFC for handling remote MTP3 congestion. However, Option 2 becomes a bit awkward. If it uses the T29/T30 mechanism, the multiple congestion levels at MTP3 level become useless¹; if not, new mechanism is needed to decrement remote MTP3 congestion. We shall henceforth ignore option 2 from our discussions.

In Option 3, the congestion level is decremented using timers T15 and T16 at MTP3 level along with the sending of SRSCT messages by the receiver. It has been observed by many people (including our own experiments) that standardized values of T15/T16 timers are too large, and their constant duration may cause oscillations, however, we shall not dwell on such issues in this paper. Also, the TFC procedure indicates congestion towards a destination (e.g., node B), rather than over a particular route-set. In spite

¹Actually, option 2 specifies how to adjust locally detected congestion levels, but not the one reported by TFC [3]

of losing some information, such an indication is desirable for network level control because it only requires maintaining node-level information. We shall continue to use this approach in dealing with bandwidth congestion on the trunks in the context of GCC. In the case of associated signalling, there is no TFC procedure, and each node only keeps track of congestion on all its outgoing links.

The above mechanisms have also been adopted for dealing with signal message handling (SMH) congestion (i.e., the congestion of level 3 processor). (Often, the onset and discard thresholds are chosen to coincide in this application.)

Automatic congestion control (ACC) was defined to handle trunk and application processor congestion in ISUP (but also adopted for BISUP). The basic procedure is as follows: an exchange can define up to 3 (but usually 2) levels of congestion for the application subsystem. The exact mechanism for measuring congestion (e.g., queue occupancy, processor utilization, etc.) is implementation dependent and is not specified. If a call attempt (i.e., an IAM message) encounters processor congestion at a non-originating UP node, it is dropped. This requires sending a REL message to the preceding exchange, which carries the ACC indication. The ACC receiver responds to the ACC indication by cutting down traffic headed towards the ACC source.

Each exchange defines a number of *response categories* for dealing with received ACC indication. Each incoming trunk at an exchange is assigned a response category. The exchange also defines the percentage of traffic to block for each response category at each ACC level. The amount of time for which the blocking remains in effect towards the ACC source is governed by a timer, whose value could be a function of the response category. Further ACC indications from this node simply extend (i.e., restart) the timer. If the timer expires, the traffic is not blocked any longer. In the BISUP context, a response category could be associated with a virtual path.

Automatic code gapping (ACG) controls have been defined specifically for application level congestion control of the SCPs. An ACG message from SCP to the query source specifies two parameters: (a) a gap code (or level), and (b) gapping interval T . The gap code is a number which is looked up in a table to determine the gap duration t . Then for time T , the query source is limited to sending at most one query to the SCP every t seconds. Note that while the ACC results in a percentage thinning of the traffic, the ACG control enforces an upper bound on the traffic rate.

A study of ACC controls in [4] has shown that two ACC levels are inadequate to smoothly control the traffic, and significantly better performance is achieved by using 10 levels instead. It is also shown that ACC shows poorer behavior than the ACG procedure in the same environment, since unlike ACC, the congestion indication does go to the traffic source. This has two implications: (1) sending ACC to the source (as attempted in the GCC mechanism explored here) would be beneficial, and (2) in the emerging AIN/PCS service scenarios where SCPs communicate directly or the querying SSP is not the originating SSP, ACG controls would also benefit from being carried to their respective sources. It may also be worth investigating whether one of these two mechanisms can be used for both SSPs and SCPs, although we shall not do so here.

As stated earlier, at least in the near future, broadband networks are expected to be rather sparsely connected, thereby requiring most calls to go through one or more tandems or intermediate switches. Thus, in a broadband network, sending ACC to only the adjacent nodes may allow the call setup to progress over several hops before being rejected. It appears that sending ACC to the source is more important in a broadband network than in a narrowband network.

3 Congestion Detection for GCC

This section briefly discusses how congestion severity levels are assigned to the three types of congestions. The use of these congestion levels in throttling traffic is discussed in the next section.

3.1 Signalling Link and SMH Congestion

The link congestion status is available to only MTP3 layer. At application (APPL) layer, the congestion information is by the point code, rather than the link. Assuming that MTP3 has some mechanism available to determine the point code on the other end of a link-set, the link congestion level can be passed to the APPL layer in appropriate format in the case of associated signalling. For quasi-associated signalling, the adjacent point code is a STP rather than the next SSP on the call path, and the necessary information is collected in two parts:

1. The indication from MTP3 layer specifies congestion status of the local linkset going to the STP. Since a message going any next user-part (UP) node will experience this congestion, it can be interpreted as congestion towards all next nodes via this STP. However, in order to avoid enumeration all such next nodes, the congestion can be viewed as occurring at the current node itself (for any message passing through this node).
2. The TFC mechanism informs the MTP3 layer about remote congestion towards the next UP node.

Similar mechanisms apply to the SMH congestion. That is, SMH congestion at the local UP node represents congestion towards all destinations², and towards the destination indicated by the TFC message in case of SMH congestion at a STP. As usual, the SMH congestion information is also passed to the APPL layer. Assuming that the same type of congestion control option is used for both link and SMH congestion,

²This assumes a centralized level 3 architecture at UP nodes; in case of a distributed architecture, the congestion is experienced only towards the destinations served by the congested level 3 processor. It may also be noted here that SMH congestion is unlikely at a UP node under quasi-associated signalling, but quite possible for associated mode signalling. For generality, we assume that SMH congestion control has been implemented at all nodes.

there is little reason to distinguish between them as far the APPL level traffic control is concerned. That is, the APPL layer simply retains the maximum congestion level towards the current or next node without regard to the type of congestion. We shall henceforth collectively refer to both types of congestions as “MTP congestion”.

As stated in section 1, the national option with priorities confines dealing with MTP congestion at MTP3 level only. However, in order to provide a unified GCC mechanism for all congestion types, the APPL layer must obtain some measure of congestion severity so that it could be passed towards the source for traffic control. Three simple schemes for doing so are as follows:

1. Use a priority based treatment across nodes as well. This option is unacceptable because of its poor granularity (only on-off control over new calls). When crossing network boundaries, a priority based scheme may not even work at all.
2. Consider the congestion priority level itself as the congestion severity level at APPL layer. In this case, the number of severity levels are necessarily limited to 3 and the thresholds are dictated by the MTP layer.
3. Determine the congestion severity level at the APPL layer using T29/T30 timers as in the international option. This offers much more flexibility in the choice of thresholds, but at a greater implementation complexity.

Although other schemes can also be devised, any significant deviation from existing SS7 mechanisms is unlikely to gain acceptance. The use of international option in this context is particularly attractive since it has been shown to work well in practice. In fact, results in [5] show that in most cases, this scheme works significantly better than the priority based MTP level congestion control scheme. In view of this, once the international option is implemented at the APPL level, message discarding at the MTP3 level may even be removed for lower complexity and better performance. However, we shall not do so in this study.

3.2 Call Processor Congestion

Call processor congestion detection is based on abatement and discard thresholds on the processor queue length for each congestion severity level. Our implementation uses a single FIFO queue for all messages, however, in case of priority queuing, the thresholds will still be on the overall queue length. At any time, the prevailing congestion severity level is used to throttle new call setup requests (i.e., incoming or locally originating IAM messages), but messages relating to ongoing calls are not throttled. The current ACC mechanism provides for at most 3 congestion severity levels, and we assume 3 levels for this study. It is, however, shown in [4] that more levels may be beneficial. The question of appropriate number of levels is intimately tied to the throttling percentage at each level and feedback delays, and is beyond the scope of this study. The distance

between the successive levels can be set using principles similar to those for setting SMH congestion thresholds and again is not the primary focus of this study.

If an IAM arrives at some node A with processor congestion, it may be dropped (always or probabilistically based on the throttling percentage assigned to that congestion severity level). In case of a drop, a REL is sent backwards indicating processor congestion at node A towards the final destination (i.e., the called switch).

3.3 Trunk Congestion

Unlike other types of congestions, detection of trunk (or more accurately bandwidth) congestion cannot be based on a queue length since any call that is unable to get the desired bandwidth must be dropped immediately. This property calls for devising a new method for measuring bandwidth congestion. Alternate routing further complicates this measurement, since it is not clear if the inability to get bandwidth on paths other than the final-choice path should be considered towards detecting trunk congestion. Restricting congestion detection to final-choice path may result in belated congestion control, particularly when feedback delays are large. On the other hand, considering rejections on the first-choice path may result in poor resource utilization. Congestion detection based on rejections on the last and last but one paths may provide a good compromise, but this aspect is not explored here. Yet another difficulty in measuring bandwidth congestion arises from the fact that the amount of bandwidth requested may span a large range. Thus, if the congestion measurement is in terms of number rejected requests, it may not accurately reflect the resource deficit. On the other hand, a measurement in terms of sum of requested bandwidths over rejected requests may also be inappropriate since the rejection of a few large requests may show a severe resource shortage and thus an overcontrol of the requests. Here we propose a measure that reflects both the number of unsuccessful requests and the amount of unallocated bandwidth. A comparative study of various alternate measures, although very important, is beyond the scope of this work.

To monitor bandwidth congestion over a trunk group, we choose a “measurement window” of N successive bandwidth allocation requests over that trunk group. We count the number of rejections, say r , over this window, which gives a count-based estimate of the *blocking probability*, denoted B_c , as r/N . We also accumulate the total bandwidth requested, say W_{req} , over the N requests, and also the total bandwidth rejected, say W_{rej} over the r rejected requests. Then, $B_a = W_{rej}/W_{req}$ gives the amount-based estimate of the blocking probability, denoted B_a . In order to avoid spurious detection of bandwidth congestion, we take the blocking probability for congestion detection purposes, denoted B , as $\min(B_c, B_a)$. The severity levels are then defined in terms of thresholds for B . Again, we choose 3 severity levels, and an abatement and onset threshold for each level. For the purposes of the experiments reported here, we do bandwidth congestion detection only on the final choice path.

In BISUP, when node A attempts to setup a connection towards the next node B,

either of these nodes may be designated as the “assigning exchange” for this call. If B receives an IAM from A without bandwidth allocation (i.e., node B is the assigning exchange) and no bandwidth is available, a REL must be sent out to A (in addition to updating trunk congestion level). This REL will carry an indication bandwidth congestion at node A (towards node B). On the other hand if A is the assigning exchange, and is unable to obtain bandwidth to node B even along the final choice path, it sends out a REL backwards indicating congestion from A towards B. Note that unlike processor congestion, the congestion is not towards the final destination.

4 GCC Propagation and Traffic Throttling

In order to deal with congestion situations, APPL level traffic throttling can be done at two points in the network:

1. Local throttling, i.e., at the node where the congestion is originally detected (either directly by this UP node or via TFCs sent by a STP on some path to the next UP node). This form of throttling is already specified in the SS7 standard, although the amount of throttling to be done is an implementation issue.
2. Remote throttling, i.e., at some node on the call path from the calling UP node (the traffic source) to the predecessor of the congestion detecting UP node. The throttling may occur at more than one such node, however, it is most beneficial to do the throttling as close to the traffic source as possible. Current SS7 standards have only a very weak form of this capability (e.g., the ACC which allows throttling at the neighboring node of the congestion detecting node).

In the next two subsections, we discuss issues related to local and remote traffic throttling. We then discuss some other issues concerning GCC, including throttling duration and GCC frequency.

4.1 Local Throttling

The amount of local throttling that can or should be applied depends on the type of congestion. With our earlier assumption that bandwidth congestion is recognized only when no bandwidth is available on the last choice path, the local throttling must necessarily be 100% irrespective of the severity level of the bandwidth congestion. In contrast, under processor congestion, the traffic (corresponding to new call originations) can also be thinned probabilistically based on the throttling percentage assigned to each severity level. However, a 100% local throttling could result in lower delays (perhaps at the cost of more failed calls) and needs to be studied.

In the case of MTP congestion, the appropriate throttling depends on the option deployed. For international option or national option without priorities, the APPL layer is entirely responsible for traffic throttling, and either probabilistic or 100% throttling

could be used. Again, 100% throttling will perhaps yield lower delays but more failed calls, and needs to be studied. For national option with priorities, a probabilistic throttling is not useful since an overwhelming percentage of IAMs carry a congestion priority of 0 and will be dropped by the MTP3 layer anyhow.³

Since this study concentrates on the national option with priorities, we have adopted a uniform policy for dealing with all types of congestion: Local throttling will always be 100% and probabilistic throttling is used exclusively for remote throttling.

The current SS7 standard is not very clear whether throttling should affect only the new calls (e.g., IAM messages only) or call setups in progress too (i.e., other call related messages such as ACM, ANM, etc. also). The general view appears to be that only new calls should be throttled and every effort should be made to complete call setups that are already in progress. We have taken this view here, however, it might be interesting to study the impact of throttling all messages. For the national option with priorities that we concentrate upon here, non-IAMs will be throttled by the MTP3 level according to their priorities.

One implementation note is in order with respect to the use of international option for assigning congestion severity levels at the APPL level. According to SS7 specifications, whenever a message encounters congestion locally, the user part (UP) is informed via MTP-STATUS message, which could be treated like a TFC for starting T29/T30 timers. However, for the sake of simulation efficiency, our implementation sends a status message only when the congestion status changes (upwards or downwards). Therefore, whenever T29 expires, we restart it immediately if the congestion status is still non-zero (instead of waiting for the messages to fly by). For this to work properly, we regard the congestion on a SSP/SCP to STP link as a local congestion at that SSP/SCP itself. This is a correct view since messages directed to any next UP node will encounter congestion in such a situation.

4.2 Remote Throttling

Given a mechanism for for establishing congestion severity level at the UP node that detects the congestion, the next task is to propagate this information backwards along the call path. The GCC mechanism does so just as in the ACC mechanism: whenever an IAM is dropped by the local throttling mechanism, the REL message sent backwards contains the necessary GCC information. This indication includes the following 3 pieces of information about the congestion: (1) Congestion type (MTP, processor, or bandwidth), (2) Current congestion severity level, and (3) Identity of the node that detected the congestion (and originated the GCC indication). In the case of signalling link or bandwidth congestion, the detecting UP node knows the identity of the next UP node towards which this congestion exists, however, a further propagation of this information does not appear to be very useful. Instead, as far the GCC is concerned, the congestion

³IAMs corresponding to special categories of calls such as essential services and HPC calls are typically assigned a priority 1. A probabilistic throttling will do little to give preferred service to these.

can be regarded to be occurring towards the called ISDN number. The identity of the GCC originator is useful so that traffic direct to (and possibly via) this node can be avoided.

Since the REL message must go all the way back to the calling node, a backward propagation of GCC indication all the way back to the source and remote throttling at the source are conceptually straightforward. In reality, however, there are serious difficulties in implementing the GCC concept effectively:

1. Routing Limitations: The routing used by BISUP (and ISUP) is based on routing tables that change only in response nodal failures/repairs. Thus, it is not possible to dynamically devise routes that avoid congested nodes.
2. Addressability Problem: A congested node can be conveniently identified via its point code within its own network, however, whenever a network boundary is crossed, such an identification becomes meaningless. That is, it may not be possible to precisely identify item (3) above for internetwork calls.
3. Lack of Knowledge: Although the routing table may provide multiple routes towards a destination, it only lists the next node for each route. That is, it is not possible to determine from the routing table which of the several routes avoid a node that is reported to be congested by the GCC.
4. Domain of Control Issues: The propagation of GCC may result in a node in one network controlling traffic going into another network. This may not be agreeable to the network operators on the grounds of its potential for improper control or deliberate abuse.

Since static routing is a fundamental feature of BISUP/ISUP, it is not possible to make use of the ID of GCC originating node for an intelligent rerouting of the failed call. Consequently, it may not be a good idea to keep retrying a failed call until either all alternate routes are exhausted or the call goes through. Instead, if the call fails once or twice, it might be better to drop it. That is, it is better to rely more heavily on traffic throttling than traffic rerouting for the purposes of congestion control.

The addressability problem can actually be exploited for structuring congestion notification across networks. Connections between two different networks are typically sparse and are provided by a number of “gateway nodes”. These gateway nodes are typically tandems, but may also have originating/terminating traffic. If a gateway node G connects two networks A and B , it must be identifiable by both either via a common point code or via a separate point code for each network. In either case, both networks should be able to address G via a point code. Now suppose that a call is attempted from a node $A \in \mathcal{A}$ to a node $B \in \mathcal{B}$ via some node $C \in \mathcal{B}$ which happens to be congested. Now the REL message back from node C to G will carry the GCC indicating congestion at node C towards node B . However, when the REL is propagated backwards from node G , it is no longer meaningful to retain the point codes of nodes C and B ; instead,

the GCC message will now indicate that there is a congestion towards the called ISDN number via the gateway G . Consequently, when node A receives the congestion indication, it can take two actions: (a) retry the call via another gateway, if any, and/or (b) cut down traffic towards the affected ISDN numbers. Since in most cases, there might be only one gateway between two networks, it is important for G itself to retry the call via a few alternate routes before sending the GCC indication backwards to network \mathcal{A} . Unfortunately, such a scheme increases the delay in informing the source of the congestion. The resulting delay in traffic throttling may well negate any advantages of retrying calls before backtracking beyond the network boundary.

This filtering of information at the gateway is in line with the fact that network \mathcal{A} typically does not, and perhaps should not, know about the internal routing within network \mathcal{B} . In fact, even if node A could somehow identify node C , the static routing would still prevent it from rerouting the call around node C ; the only advantage of this knowledge would be that traffic destined to node C could then be cut down. The only loss of information in going across network boundaries is that it is no longer possible to tell if the congested node is the same as the called node. Within the same network, this information can be derived from the called number and the point code of the congested node (although it does require a table lookup). Such an information is very useful, since if the destination node is congestion, alternate routing is of no use and should not be tried. Therefore, we propose a fourth piece of information to be carried by a GCC — essentially, a single bit indicating whether the congested node is same as the called node.

Since passing the point code of the called node is not meaningful across networks, node A can only rely on the called ISDN number for the purposes of cutting down traffic destined to node B . That is, node A should be able to somehow determine what portion of the called ISDN number corresponds to the called exchange. For national calls, this is a function of the numbering plan used, and may or may not be easy to determine. (In some countries such as USA, the exchange information is obvious from the called party number.) Notice that a precise identification of the congested exchange, although highly desirable, is not essential; the only requirement is that the set of ISDN numbers towards which traffic is cut-down be a good “approximation” to the set of ISDN numbers terminated by the exchange in question. We assume that such an approximation can be easily identified within any national network and is expressed in terms of which digits in the called number represent the exchange code. This information can be carried explicitly by the GCC indication. At a gateway joining two national networks, this information is passed back in the GCC indication so that it is available for international calls as well. The implementation used here, however, does not support this extension.

The control of traffic from one network into another is purely a problem of mutual agreements. If such a control is not allowed, the GCC information must stop at the relevant gateway, rather than being carried all the way back to the source. This can be handled easily via a table lookup and will not be addressed further in this document.

In general, a destination may experience more than one type of congestion simultaneously. Thus, successive GCC indications may indicate different congestion types, levels,

and throttling percentages. One simple way to handle this situation is to maintain only a single congestion indication at the node. Each time a new GCC indication arrives, its throttling percentage is compared against the current throttling percentage. If the new throttling percentage is higher, the nodal indication is updated to reflect the new congestion type, level, and throttling percentage. Such an approach would ensure that adequate throttling is always maintained. Of course, there are other ways too, but we shall not study them here.

Consider a situation where a REL containing a GCC indication arrives at a node P on its way from the congested node C to the source S. (We allow the possibility P=S, i.e., P could be the source as well.) Then two actions are possible at node P (a) attempt to reroute the failed call at node P, and (b) start throttling traffic bound from node P to the called (or destination) node D. If P is the immediate predecessor of node C, P can easily choose an alternate route that avoids C (if one exists); therefore, given the static routing of ISUP/BISUP, alternate routing from P should always be tried (provided that $C \neq D$, i.e., the destination itself is not congested). If $P \neq S$, we allow alternate routing from the source node as well, although the source has no way ensuring that node C is avoided. In order to avoid shifting bottlenecks, we avoid rerouting if the GCC level is greater than 1.⁴

If the alternate routing is possible, there is no reason to start throttling traffic. That is, action (b) above is taken only if action (a) is not possible. Action (b) itself is limited to the designated control points (e.g., the source or a gateway node). As stated above, in this study, remote throttling is done at the source node only. Thus, at nodes between P and S (including P but excluding S), the congestion information carried by GCC is not recorded. This means that the calls originating at these nodes and bound for node D, will not be throttled as a result of a GCC going from D to S. However, such calls will likely fail and result in GCC's explicitly delivered to these nodes and result in throttling. The main advantage of this scheme is that if the calls originating at an intermediate node do not go through C, they will not be affected.

In BISUP, the bandwidth assignment over a trunk group could be done by the node on either end of the trunk group. Let i and j denote the point codes of two nodes connected by a trunk group G. Then, for odd-numbered virtual paths, the node with point code $\min(i, j)$ assigns the bandwidth and virtual channel identifier (VCI) value, whereas for even-numbered virtual paths, the node with point code $\max(i, j)$ does the assignment. This mechanism avoids the problem of "glare" which is characteristic of ISUP; however, it does result in some minor complications. In particular, if an IAM travels from node i to j , node i is not necessarily the *assigning exchange* and thus may not know about potential lack of bandwidth or VCI values. If, subsequently, node j discovers inadequate resources, it will drop the call and send a REL back. In this case, node j will be the GCC originator, but the congestion is really from i towards j . This situation can be handled by avoiding any GCC-related action at node i , while allowing node i to retry the call as per BISUP specifications.

⁴The soundness of this approach should be checked experimentally, but is not done here.

4.3 Throttling duration and GCC Frequency

The duration of local throttling is directly governed by the congestion severity level; that is, so long as the congestion is at some severity level $i > 0$, the traffic is throttled (probabilistically or 100%). Because of feedback delays, such a scheme may lead to oscillations for remote throttling. Therefore, it is essential to introduce a timer, a more reasonable approach is the one used by ACC, henceforth denoted as T_{GCC} , as in the ACC scheme. However, the GCC scheme proposed here differs from ACC in two important respects, and may require a somewhat different way of setting the throttling duration.

The first difference is that the ACC scheme uses only congestion onset thresholds; no abatement threshold is needed for its operation. Consequently, the throttling node must “guess” as to when the congestion has abated. This is done by choosing a fairly large value for T_{ACC} , in the hope that if no congestion indication arrives for the time T_{ACC} , the congestion must have abated. This may result in traffic overcontrol. In contrast, the basic congestion detection mechanisms used to drive GCC all include abatement thresholds. For example, the T29/T30 mechanism for assigning MTP congestion severity levels would lower the severity level only after the abatement threshold has been crossed from above and subsequently the onset level is not crossed for at least the duration T30-T29. Thus, by the time a lower GCC level is reported (or, in case level 1, by the time frequent ACC indications stop), the congestion should have abated (with a high probability). In view of this, T_{GCC} can be chosen to be quite small (e.g., 0.5-1.0 seconds, as opposed to 5 seconds in case of ACC). Note that as in ACC, the GCC timer is restarted every time a new GCC arrives, even if that GCC reports *any nonzero* severity level. Thus, so long as the congestion remains above the abatement threshold, the timer is unlikely to run out even if it is chosen to be quite small.⁵

The second difference arises from the fact the GCC-induced throttling may occur at a node that is far removed from the congested node, where in the ACC scheme, throttling occurs at the neighbouring node. This means that a GCC may suffer much higher feedback delays than an ACC. It is well-known that congestion control becomes less effective as the feedback delay increases. For example, by the time the source starts throttling traffic in response to the first GCC, two unfavorable situations could develop:

1. If the congestion is short-lived, it may have already abated, which means that the source throttling is unnecessary and degrades the throughput. A small T_{GCC} is desirable in this respect.
2. If the congestion is long-lived, the control does not act fast enough. In this case, there will be a lot of call drops at the congested node because of the local throttling. Local throttling wastes resources and should be minimized.

The first observation implies that a small T_{GCC} is desirable (however, a too small a value may result in a premature withdrawal of throttling thereby leading to the possibility

⁵This is unlike the TFC procedure (for national option with priorities), where the T15 timer is restarted only when a higher congestion level is reported.

of oscillations). The second observation implies that sending GCC all the way to the source doesn't do much good if the feedback delays become quite large. Thus, for international calls between large national networks, it might even be preferable to send GCC only up to the national gateway nodes and throttle traffic there.

Note that a call attempt at node A going towards node B, may be dropped even if it does not encounter any local congestion. This happens if node A had earlier received a GCC which caused it to start shedding traffic towards the called node, say C. The necessary and sufficient conditions for the drop are: (a) The GCC timer for destination C must be currently running, and (b) Node A must be the node at which traffic bound for C and passing through A must be throttled. As stated above, we shall perform remote throttling at the source only; therefore, condition (b) amounts to saying that node A must be the calling node.

Until now, we have implicitly assumed that a GCC is sent out towards the source whenever a call encounters congestion and is dropped (excluding remote drops as a result of GCC-induced throttling). Although GCC indication does not result in additional traffic, setting and processing GCC indication for every dropped call might result in unnecessary overhead. Therefore, it might be desirable to set GCC indication on every n th failed call, where n is chosen experimentally. In terms of implementation, a single bit in the GCC portion of a REL could indicate whether the rest of the GCC field contains useful information or not. Clearly, a large n is undesirable as it could let T_{GCC} run out prematurely, but $n \approx 10$ should work well.

5 Experimental Setup and Results

In this section, we compare the performance of GCC control against the existing controls (assuming that ACC is implemented) for simple BISUP calls under associated signalling. Figure 1 shows the network used for this study. For simulation efficiency, the network has been kept rather simple and can be viewed to consist of 3 subnetworks, each with a "gateway" or a tandem connecting to other subnetworks. In a broadband network, signalling information is carried on certain virtual circuits; therefore, the signalling and trunk paths would usually be the same. We shall assume this to be the case in this study. The calls between the switches of the same subnetwork do not go via any intermediate switch, but those between subnetworks may go through 2 tandems (on the first choice path) and 3 tandems on the second and final choice path. The network is engineered such that under normal conditions the traffic between any two subnetworks is identical and all processors and links carry the nominal load. In order to study overload scenarios, we concentrate on the calls going from subnetwork 1 to subnetwork 3 such that the congestion is encountered either at tandem 3 (SMH and level 4 processor congestions) or from tandem 3 towards the destination switch (signalling link and bandwidth congestions). Congestion is induced by artificially reducing the capacity of the relevant element. In order to avoid confusion between various types of congestions, only one congestion type is induced at a time. In order to study the impact of loss of information across network

Figure 1: Network Model used in Simulation

boundaries, we shall consider two following two scenarios:

1. All subnetworks are a part of the same network, and thus the point-code information can be exchanged between them.
2. Each subnetwork actually represents a separate network and the tandems form the gateways between these networks.

In both cases, we shall allow traffic control to occur at the source, since it is clear that as the control moves away from the source, it will perform poorer.

6 Conclusions and Future Work

This paper has shown that overload performance of the signalling network can be improved considerably if traffic throttling can be moved to the traffic source, rather than done close to the congested element. The paper also studied how to detect bandwidth congestion in the BISUP context. The work to date does not deal with this congestion type since this type of congestion does not exist in ISUP. Finally, the paper studied the problems of propagating congestion indication across network boundaries. It was shown that despite the loss of important information when moving across network boundaries, propagating congestion indication to the source is beneficial.

We note here that the idea of propagating congestion indication to the traffic source is not new, and in fact has been advocated by several researchers. However, the standards bodies have been reluctant to accept it because of the difficulties involved and

the additional complexity. However, certain new capabilities proposed for ISUP (and applicable to BISUP) have a similar flavor. For example, the crankback facility allows the call to be retried further back along the call back based on the failure indication on the first try. We think that crankback paves the way for implementation of GCC as proposed here. As stated earlier, the more sparse connectivity in the BISUP environment and the use of associated signalling may also spur the standards bodies to accept a GCC like proposal. This paper shows some preliminary results towards such an acceptance, however, more extensive experimentation is perhaps required for proceeding further on this issue.

In previous sections, we raised a number of issues that are important to study in the overall context of GCC. These are restated in somewhat more detail in the following:

1. SCP congestion controls. A SCP uses ACG control instead of the ACC control to deal with level 4 processor congestion. ACG controls were originally designed considering the needs of simple SCP applications such as WATS or LIDB where the queries are atomic and concern only one SCP. In more complex AIN applications such as PCS, SCP interactions may include conversations (i.e., multiple query-response pairs) between the SSP and SCP, one-way SCP to SSP notifications, SCP to SSP queries, SCP to SCP queries to handle the original query from the SSP, etc. In such an environment, the SCP traffic could be very bursty in nature, and a precise rate control provided by the gapping mechanism may not be any better than the percentage throttling provided by ACC. Also, the ACG controls now no longer go to the traffic source (the overloaded SCP may have received a query from another SCP) and thus will need extension for effective control. It remains to be seen if the GCC mechanism proposed here can be used for SCPs as well. Such a scheme will be useful not only for level 4 processor congestion but for other congestion types as well. For example, in the case of congestion on an outgoing signalling link from a SCP (or from a STP to SSP), the current congestion controls simply drop the responses to be sent out, without doing anything about reducing the rate at which the source SSP is generating queries.
2. Combined use of ACC and GCC. Although GCC can be considered as an extension of the ACC controls, it might be difficult to simply replace ACC mechanism by the GCC mechanism in the existing signalling software. Thus, there might be a case for retaining ACC controls as they are and introducing GCC on top of that. (This is a likely scenario at least in terms of standards, since exercising an existing mechanism almost never happens in international standards.) The question to study in this context is whether the simultaneous presence of both controls would lead to any undesirable interactions and poor performance. For example, independent throttling of traffic both at the neighbor and at the source may result in overcontrol.
3. SCCP Congestion controls. SCCP congestion controls are currently not well developed. The ITU standard only states that when the local MTP receives a congestion

indication (including remote SCCP congestion), it passes it to the SCCP layer via the MTP-STATUS message. The use of this information by SCCP is not discussed at all. The use of complex AIN services considerably increases SCCP load, and an effective congestion control scheme becomes essential. There are several issues to examine in this regard: (a) if the SCCP layer uses a distinct processor, how do you detect congestion on this processor, (b) how do you determine the congestion severity level based on the congestion reports by the MTP-STATUS message, and (c) Can the GCC scheme be used for propagating congestion indication from SCCP?

4. **Detecting Bandwidth Congestion.** In this study, we did not explore many issues concerning bandwidth congestion, since the emphasis of the paper is on network level aspects. Some of the crucial questions concerning bandwidth congestion include: Is the current congestion measure (both the number of rejections and the total bandwidth demanded exceeding some thresholds) optimal? How do other simple measures compare against this one under various bandwidth demand scenarios? Does the correlation between successive bandwidth demands have a significant impact on the measure? Is it a good idea to declare bandwidth congestion before the bandwidth is actually exhausted (i.e., when the unallocated bandwidth falls below some threshold)? Since no REL will be sent out immediately in this case, is it okay to propagate GCC via end-of-call RELs going backwards (i.e., RELs for calls where the caller hangs up first)? It would be interesting to see if the question of appropriate congestion thresholds can be studied analytically. request arrivals.
5. **Number of severity levels and cut-down percentages.** In this study, we arbitrarily assumed 3 severity levels for all congestion types and the traffic throttling percentages of 40, 70, and 100%. Although the study in [4] showed that 10 levels provide a much better performance than the 2 levels available for ACC, this conclusion is by no means definitive. The problem is that the number of levels is intimately tied to the cut-down percentages used. There is little reason to believe that a constant increase in throttling percentage for each successive level (as assumed in [4]) is a good choice. In fact, under heavy overloads, such a scheme coupled with many levels will perform poorly because of a long delay in reaching the appropriate level of throttling. Perhaps a sharp cut-down in traffic at the first severity level, and then a more gradual increase in cut-down is a better choice and wouldn't need so many levels. A related question is that of the impact of congestion type on the number of levels and throttling percentages. It could well be that with high variability in demanded bandwidths, the number of levels for this congestion type is different from those for, say, SMH congestion. Since these problems are very difficult to address by simulation, a tractable analytic model would be highly desirable.
6. **Throttling Rules:** In this study, we assumed 100% local throttling for all congestion types, and reserved probabilistic thinning only for remote control. It is useful

to see if better performance can be achieved by a local probabilistic thinning whenever possible (except for the National option with priorities, where probabilistic thinning at APPL level does not make any sense). Also, throttling percentages that increase with severity level is just one of several ways of controlling the traffic. One attractive alternative might be to simply turn off the traffic completely for a period of time that increases with the congestion severity level. At the end of this period, the traffic could be increased slowly based on the reported severity level. This scheme begins to look somewhat like the flow-control scheme in TCP/IP, and the ideas from TCP/IP schemes could perhaps be used.

References

- [1] “Bellcore’s specification of Signalling System no 7”, Bellcore GR-246-CORE, chapters T1.113.1-5, BT1.113.1-5, Dec 1994.
- [2] “Broadband switching system SS7 requirements using BISUP”, Bellcore GR-1417-CORE, Issue 2, Nov 1995.
- [3] D.R. Manfield, G. Millsted, and M. Zukerman, “Congestion Controls in SS7 Signalling Networks”, IEEE Communications Magazine, June 1993, pp 50-57.
- [4] B. Northcote and M. Rumsewicz, “An investigation of CCS Tandem Overload Control Issues”, Proceedings of GLOBECOM, Nov 1995, Singapore, pp718.
- [5] M.P. Rumsewicz and D.E. Smith, “A Comparison of SS7 Congestion Control Options During Mass Call-in Situations”, IEEE Trans. on Networking, Vol 3, No 1, Feb 1995.