PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04Q 3/00 | A1

(11) International Publication Number: WO 98/57504

(43) International Publication Date: 17 December 1998 (17.12.98)

(21) International Application Number: PCT/GB98/01628

(22) International Filing Date: 4 June 1998 (04.06.98)

(30) Priority Data:
9712307.9 12 June 1997 (12.06.97) GB

(71) Applicant *(for all designated States except US)*: BRITISH TELECOMMUNICATIONS PUBLIC LIMITED COMPANY [GB/GB]; 81 Newgate Street, London EC1A 7AJ (GB).

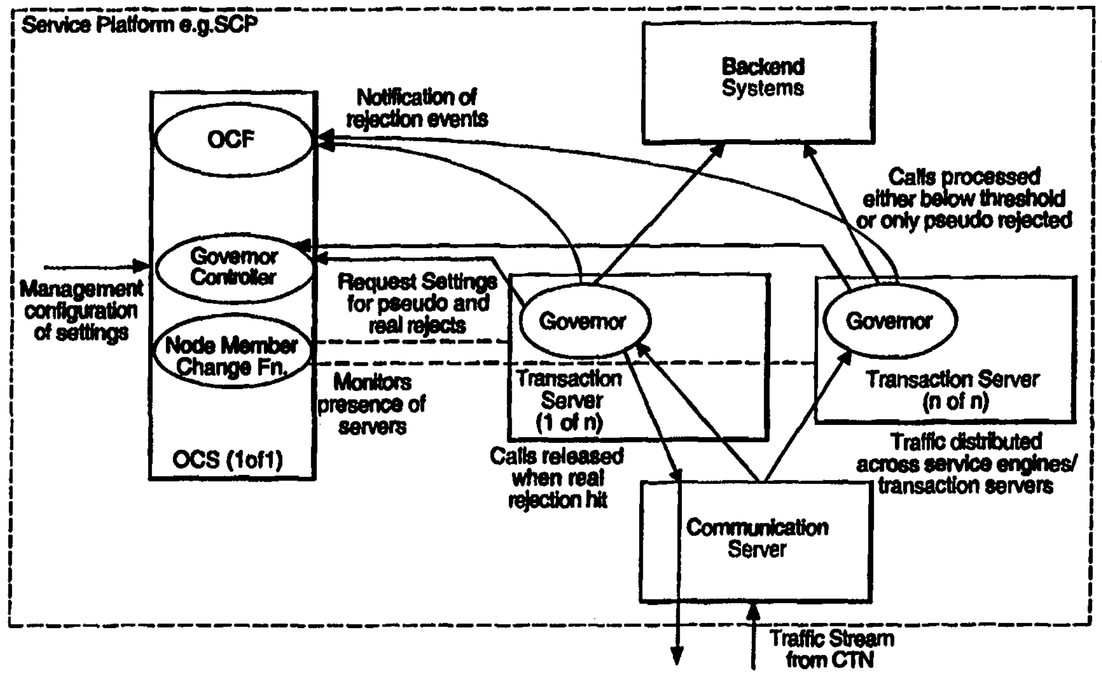
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: CRETCH, Philip, Paul, Bartlett [GB/GB]; 3 Crown Close, Martlesham, Woodbridge, Suffolk IP12 4UH (GB). SCOTT, Michael [GB/GB]; 33 Chainhouse Road, Needham Market, Ipswich, Suffolk IP6 8TB (GB). WILLIAMSON, Paul [GB/GB]; 58 Bury Hill, Melton, Suffolk IP12 1JD (GB). HUNT, Rowland, Geoffrey [GB/GB]; 11 South Close, Ipswich, Suffolk IP4 2TH (GB). COOKSON, Martin, David [GB/GB]; The Spinney, 11 Cambridge Road, Little Abington, Cambridge, Cambridgeshire CB1 6BL (GB).

(74) Agent: WELLS, David; BT Group Legal Services, Intellectual Property Dept., Holborn Centre, 8th floor, 120 Holborn, London EC1N 2TE (GB).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: OVERLOAD PREVENTION IN A SERVICE CONTROL POINT (SCP)


Service Platform e.g.SCP
Backend Systems
Notification of rejection events
OCF
Calls processed either below threshold or only pseudo rejected
Governor Controller
Management configuration of settings
Request Settings for pseudo and real rejects
Governor
Governor
Node Member Change Fn.
Monitors presence of servers
Transaction Server (1 of n)
Transaction Server (n of n)
OCS (1of1)
Calls released when real rejection hit
Traffic distributed across service engines/ transaction servers
Communication Server
Traffic Stream from CTN

(57) Abstract

A transaction processing system, which may be a telecommunications service platform, includes a network interface which in use is connected to a communications network, a transaction processor and a governor. The governor is programmed with a static limit value for the rate of transaction processing, and causes inbound transactions to be released once the limit value is reached. The governor may respond both to initial messages at the beginning of a transaction, and to intermediate messages during the subsequent processing of a transaction.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece	**ML**	Mali	**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**MN**	Mongolia	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MR**	Mauritania	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MW**	Malawi	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MX**	Mexico	**US**	United States of America
CA	Canada	**IT**	Italy	**NE**	Niger	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NL**	Netherlands	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NO**	Norway	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NZ**	New Zealand	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**PL**	Poland		
CM	Cameroon	**KR**	Republic of Korea	**PT**	Portugal		
CN	China	**KZ**	Kazakstan	**RO**	Romania		
CU	Cuba	**LC**	Saint Lucia	**RU**	Russian Federation		
CZ	Czech Republic	**LI**	Liechtenstein	**SD**	Sudan		
DE	Germany	**LK**	Sri Lanka	**SE**	Sweden		
DK	Denmark	**LR**	Liberia	**SG**	Singapore		
EE	Estonia						

OVERLOAD PREVENTION IN A SERVICE CONTROL POINT (SCP)

The present invention relates to a transaction processing system in a communications network, and in particular to the prevention of overloading of the transaction processing system by inbound traffic.

In designing communications networks, there is a need to provide efficiency in the use of network resources, whilst ensuring that the network functions reliably under widely varying traffic levels. The first of these criteria makes it desirable to operate service platforms at close to their maximum capacity. This however carries a risk of the platform failing if a sudden peak in traffic leads to the maximum capacity being exceeded. This is a particular problem for service control platforms (SCP's) in an intelligent network. SCP's are often connected to other components of the network by non circuit-related signalling systems. Although the limited number of available circuits acts as a constraint which protects other elements of the network from overload, no such constraint applies to the inbound signals directed to the SCP. Therefore, the SCP is particularly vulnerable to overload. Such overload may be manifested as a lack of processing power, lack of free memory, or exhaustion of available internal communication bandwidth.

According to a first aspect of the present invention, there is provided a transaction processing system for use in a communications network, the system including:

a) a network interface for communicating signalling for inbound traffic

b) a governor programmed with a static limit value for the rate of traffic-related signalling, and

c) a transaction processor which is responsive to the governor and which is arranged to release an inbound transaction when the static limit value is reached.

Preferably the transaction processing system comprises a service platform for use in a telecommunications network. The service platform may be a service control platform (SCP) used in an intelligent network. Alternatively the service platform might take the form of a computer-based system using CTI (computer telephony integration) technologies to provide intelligent services for a telephony network. The invention is also applicable, for example, to systems which carry out real-time processing of internet (TCP/IP) packets.

The present invention protects the service platform or other transaction processing system by providing it with a governor. The governor compares the rate of inbound calls to the platform with a static limit or threshold, and causes calls to be released when the limit is exceeded. The inventors have found that this approach is flexible and powerful in protecting the platform, and functions effectively even when it is uncertain which of the platform resources determines the point of overload. Surprisingly, the use of a static limit is found to give better results than directly monitoring the available resources on the platform to determine the point of overload. The operation of the governor can be made substantially independent of the detailed design of the service platform, and can protect network resources which may not be sited locally within the platform.

Preferably the transaction processing system further comprises:

d) an overload controller which is connected to the governor and which is arranged to output control signals onto the network to reduce the rate of inbound transactions to the transaction processing system when the static limit value is exceeded. Preferably the governor is programmed with a first, lower static limit value, and a second, higher static limit value, and is arranged to signal to the overload controller when the first, lower static limit value is reached, and is arranged to release an inbound call only when the second, higher static limit value is reached.

The combination of a rate governor and an overload controller is found to be particularly advantageous. Then call release events caused by the governor serve as an input to the overload controller. The overload controller may then, for example, reduce the rate of inbound calls by sending a call-gapping instruction to an originating switch. Still further advantages can be gained in this context by operating the governor with two limit values, or equivalently by using two governors each with a different respective limit value. Then, at call rates between the lower limit value and the upper limit value, the governor causes a pseudo-rejection of an inbound call, in which the call is still admitted, but a rejection event is signalled to the overload controller. As is further explained in the detailed description below, the use of pseudo-rejections enables more efficient handling of calls in circumstances where the rate-limiting network resource precedes the governor in the transaction processing chain.

Preferably the governor is arranged to respond to the total rate of both initial traffic-related signals and predetermined intermediate traffic-related signals. As is further discussed in the detailed description below, it is advantageous if the governor responds to the rate of intermediate traffic-related signals, such as those associated with the setting up of a call leg from an originating switch to a network intelligent peripheral, as well as responding to inital signals. The governor is then able to handle different transactions with widely varying processing overheads.

Preferably the transaction processing system further comprises:

e) a governor controller which is connected to the governor and which is arranged to set automatically the value for the static limit value. Preferably the system includes a plurality of processing systems, each of the processing systems comprising a transaction processor and a governor which is connected to the respective transaction processor,

and the governor controller is connected to the plurality of processing systems and sets independent respective values for the static limit values in the different respective governors.

These preferred features of the invention enhance the resilience and responsiveness of the transaction processing system by setting the static limit value automatically, and distributing call processing between several processors, each with their own governor. The governors do not necessarily refer to a governor controller on every call. The governor may refer to the controller periodically for a sample of calls. This the reduces the processing overheads.

Preferably the governor controller is programmed with a target transaction rate for the system, and is arranged to set the static limit values in the different respective governors so that the total admitted transaction rate substantially matches the said target transaction rate. Preferably the governor controller includes a monitor input which is connected to the plurality of processing systems and which receives status data for the processing systems, and the governor controller is arranged to amend the values for the rate limits in the different respective governors when the said status data indicates that the number of functioning transaction processors has changed. This enables the governor to balance the limit values of the different transaction processors so as to maintain as far as possible a desired global transaction rate for the overall system even in the event of one of the component processing systems failing.

Preferably, the governor controller is arranged to monitor the identity of network switches which, in use, direct inbound signalling to the transaction processing system, and the governor controller is arranged, when signalling is received from a switch which normally uses another transaction processing system, automatically to increase a target call rate for the transaction processing system.

This preferred feature of the invention extends the functionality of the governor controller, in such a way that it can respond automatically to changes in the condition of the communications network when, for example, another service platform fails. Monitoring the identity of the switches which communicate with the platform makes it possible to achieve this extended functionality without requiring additional inter-platform signalling.

According to a second aspect of the present invention, there is provided a communications network including a transaction processing system in accordance with the first aspect.

According to another aspect of the present invention, there is provided a method of operating a transaction processing system in a communications network, the method comprising:

a) communicating between the communications network and a governor located in the transaction processing system signalling for an inbound transaction;

b) in the governor, comparing the current rate of transactions processed by the transaction processing system with a static limit value; and

c) causing the transaction to be released when the static limit value is reached.

According to another aspect of the present invention, there is provided a service platform for use in a communications network, the platform including:

a network interface for receiving signalling for inbound calls

a static governor programmed with a predetermined call rate threshold, and

a call processor which is responsive to the static governor and is arranged to release an inbound call when the predetermined call rate threshold is reached.

According to another aspect of the present invention, there is provided a method of operating a communications network including:

a) comparing a rate of events in a network resource with a threshold rate by incrementing a leaky bucket counter in response to signalling events; and

b) applying different weights to the increments of step (a) for different signalling events, the weights depending on the degree loading of the network resource associated with the different signalling events.

This aspect of the invention is not limited to use in preventing overload of a control platform by inbound signalling, but is generally applicable to the prevention of overloading of network resources. The invention also encompasses a platform for use in such a method.

Systems embodying the present invention will now be described in further detail, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 is a schematic of a communications network embodying the present invention;

Figure 2 is a schematic of a service platform for use in the network of Figure 1;

Figures 3a and 3b are diagrams illustrating rejection events in the service platform of Figure 2;

Figure 4 is a diagram showing event flows in the platform of Figure 2;

Figure 5 is a diagram illustrating the balancing of governor thresholds across several sites;

Figure 6 is a diagram showing a shared memory network which is used in implementing the platform of Figure 2;

Figure 7 shows the physical architecture used to implement the platform of Figure 2; and

Figure 8 is a flow diagram for a governor embodying the invention;

Figure 9 shows an alternative embodiment.

A telecommunications network which uses an IN (Intelligent Network) architecture includes a service control point 1, which is also termed herein the Network Intelligence Platform (NIP). The service control point 1 is connected to trunk digital main switching units (DMSU's) 2 ,3 and to digital local exchanges (DLE's) 4,5. Both the DMSU's and the DLE's function as service switching points

(SSP's). At certain points during the progress of a call, the SSP's transfer control of the call to the service control point. The service control point carries out functions such as number translation and provides a gateway to additional resources such as a voice messaging platform.

As shown in Figure 2, the service control point 1 in this example includes an overload control server (OCS) 21, a number of transaction servers 22, and a communications server 23. The communications server 23 provides an interface to a common channel signalling network. The transaction servers are linked to a common back end system 24 which is responsible for management functions such as the collection of call statistics. In operation, when an initial detection point occurs, for example in a call made via DLE 4 and DMSU 2, then the SSP (DMSU) communicates an initial detection point (IDP) message via the C7 common channel signalling network to the service platform 1. The call is subsequently processed within one of the transaction servers 22, for example by translating an 0800 number into an equivalent destination number, and the results of the processing are returned via the signalling network to the DMSU 2 which may then. for example, route the call via the other DMSU 3 and DLE 5 to the destination.

As noted in the introduction above, there is a potential risk that the rate of call-related signalling reaching the service platform may exceed that which the platform is able to cope with. This might result in unacceptable degradation in the performance of the platform, or even complete failure of the platform. To prevent this, the transaction servers 22 include static governors 220. The static governors 220, which may be embodied as software modules running on the processors which implement the transaction servers, monitor the rate of in-bound call-related signalling at a respective transaction server and cause a call to be released once a predetermined static call rate threshold or limit value has been reached. The governors also signal rejection events to the overload control server. Ideally, the governor would be located as close as possible to the signalling entry point into the service platform, so that it has the earliest possible opportunity to detect a peak in traffic before preceding systems, such as communication servers, have become congested. In the present implementation, the governor is located within the call processing protocol layer. For some calls, the governor operates only on the initial message for the call. Positioning the governor in the call processing layer ensures that it has sufficient knowledge of the incoming message to distinguish initial messages from other messages generated by the same call. This position also facilitates generation of a call rejection message, since that is one of the functions of the call processing protocol layer. A further advantage is that the governor has access to data on offered calls and the identity of originating SSP's.

The Overload Control Server (OCS) responds to call release events signalled by the governor by sending call-gapping instructions to originating SSP's. Each such instruction carries an interval and duration timer. Admission of a call by the SSP results in an initial detection point (IDP) and starts the gap interval timer. When the interval ends, the next offered call is admitted. This process repeats until the duration time expires.

In the present example, the call processing layer is the INAP (Intelligent Network Application Protocol) layer specified in the ITU standard for Intelligent Networks. The initial message in this case is an INAP InitalDP operation. To release the call, a ReleaseCall operation is used, and a reason for the release is specified. This results in a tone or announcement being played to the originator of the call. A nodal global title is contained in a calling address which is retrieved from the TCAP (Transaction Capabilities Application Part) TC-Begin message. This title is used to identify uniquely the originating SSP for each call.

The governor in each transaction server limits the rate of admitted calls using what is termed a "leaky bucket" algorithm. As described in the present applicant's International Patent Application PCT/GB 94/02512, the leaky bucket is a counter which is decremented regularly at a fixed rate, the leak rate. In the present example, the governor adds a drip to the bucket, that is to say increments the counter, every time an initial message is received for a new call by the respective transaction server and the call is accepted. Once the bucket is filled to a level such that a reject threshold is reached, then all calls are rejected until the bucket has leaked. The bucket leaks at a rate which is equal to the governed mean arrival rate. The bucket leak rate and the reject threshold are both set by the governor controller in the OCS. The operation of the algorithm is then as follows:

1. The bucket starts with a fill level of 0 and receives a drip of value 1 every time that a new call is admitted.

2. All drips which would take the fill level above an "onset" level have no effect on the fill level.

3. When a drip would take the bucket above its onset level, the call should be rejected.

4. The bucket leaks at a rate equal to the governor threshold.

The effect of the algorithm is to ensure that when the offered traffic rate exceeds the governor threshold, then the admitted rate is limited to the threshold value.

In the present implementation, the governor triggers two types of rejection event, termed "real rejection" and "pseudo rejection". A separate call rate threshold is maintained for each type of rejection event, and separate respective leaky bucket counters are used. The real rejection event is used as described above. In this type of rejection event, as illustrated in Figure 3b, the governor generates a release call message so that calls are rejected once the predetermined call rate threshold is reached. This is appropriate if the bottleneck, that is to say the resources which would otherwise be overloaded, comes after the governor in the call processing chain. However, sometimes the bottleneck precedes the governor. This implies that once the call has got as far as the governor, there is likely to be sufficient capacity following the governor to handle the call. In this case, rather than causing the call to be released, it is more efficient to allow the processing of the call to continue. In this case a pseudo rejection event is used. The initial message is passed on to the post-governor systems for further processing. At the same time, a rejection event is signalled to the overload control function of the OCS, so that the OCS acts to reduce the incoming call rate, thereby protecting the potentially overloaded resources in the pre-governor systems. The pseudo rejection event threshold is set to a lower value than the real rejection event threshold. For example, the pseudo rejection event threshold might be set to 100 calls/sec and the real rejection event threshold might be set to 120 calls/sec. Then, if the current call rate is, say, 110 calls/sec then a pseudo rejection event is triggered. If the call rate reaches, say, 120 calls/sec, then a real rejection event is triggered.

Figure 4 illustrates the event flows in three types of situation. In region a, the current call rate is below both thresholds, and no rejection occurs. In region b, the current call rate is above the pseudo rejection event threshold and below the real rejection event threshold. In region c, the current call rate is above both the pseudo rejection event threshold and real rejection event threshold. The event locations shown in the diagram are, from left to right, the signalling network

(CTN), the governor, the governor controller, the overload control function (OCF), a node membership change function (NMCF), and the back-end system.

As illustrated in Figure 2, the service platform may in practice be implemented as a multiplicity of transaction servers, referenced 1 to n, each of which has its own governor. The use of several transaction servers both increases the capacity of the platform, and enhances its ability to withstand localised component failures. The governor controller is designed to maintain as far as possible the overall platform throughput in the event of one of the transaction servers failing. To this end, the governor controller includes a monitoring function. In the present implementation the governor controller communicates with the transaction servers using a memory channel network, and the monitoring function is implemented using a Node Member Change Function (NMCF) which is associated with the memory channel network. In use, the monitoring function determines how many of the transaction servers in the service platform site are active. The governor controller then takes a target allocated rate for the site, which target rate may have been set previously by a network management function, and divides it by the number of transaction servers available. The governor controller ensures that the resulting threshold values which are allocated to the transaction servers do not exceed the maximum rate which a transaction server is capable of handling. As an added safety measure, each governor is programmed with a maximum threshold value. The governor defaults to this maximum threshold value if it finds itself operating in isolation, for example as a result of the site governor controller failing.

In an alternative embodiment which is shown in Figure 5, a service platform is split between three sites, M,N,P. Each site has a number of transaction servers, together with a communications server and an overload control server. The transaction servers include respective governors, and a governor controller is located in each overload control server, as described previously with reference to the service platform of Figure 2. The different sites are connected to a common back-end system. In the Figure, four of the network SSP's are shown. The SSP'S have nodal global titles A, B, C & D. Each SSP connects to a single one of the service platform sites. In the Figure, the preferred connections are shown by arrows with solid lines. Alternative, second choice connections are shown by arrows with dashed lines. If a particular service platform site fails, all the SSP's currently using that site as their first choice switch to their respective second choice sites.

While the network might be provided with a central operations unit which balanced the threshold values across the different sites, the present preferred implementation uses governors and governor controllers in the sites which automatically balance the thresholds across the sites. The governor controller in each site registers all traffic offered to the site, and monitors the identity of each SSP which is directing signalling to the site. From the number of different SSP identities which are encountered in a predetermined period of time, the governor controller calculates how many SSP's are connected to it. The target threshold value for the site is then set accordingly. For example, if there are three sites with a total accumulated target of 900 calls/sec and one SSP fails, then assuming that the SSP's previously connected to the failed site are equally split between the remaining two, then each of the remaining sites sees calls from one additional call which it had not previously seen. This is detected by the governor controller in each remaining site. The governor controller raises the target governor settings in the transaction servers in the respective site to take account of the increased traffic handled by the site.

A manual override mechanism may be included in the SCP's for use, for example, to stop other sites increasing their traffic levels when one SCP is taken out of use.

Tables 1a and 1b below illustrate in further detail the balancing of thresholds in the network of Figure 5.

TABLE 1a

Global		*Variables at Site M*		*Variables at Site N*		*Variables at Site P*	
Max. whole NIP call rate	900	Active TS (out of 2)	2	Active TS (out of 2)	2	Active TS (out of 2)	2
Max. individual TS call rate	200	Requested Gov. Rate	300	Requested Gov. Rate	300	Requested Gov. Rate	300
Default TS rate (no OCS)	150	Active TS Rate	150	Active TS Rate	150	Active TS Rate	150
Active NIP sites	3	Achieved Gov. Rate	300	Achieved Gov. Rate	300	Achieved Gov. Rate	300
		Global Achieved Rate			900		

TABLE 1b

Global		*Variables at Site M*		*Variables at Site N*		*Variables at Site P*	
Max. whole NIP call rate	900	Active TS (out of 2)	2	Active TS (out of 2)	1	Active TS (out of 2)	N/A
Max. individual TS call rate	200	Requested Gov. Rate	450	Requested Gov. Rate	450	Requested Gov. Rate	N/A
Default TS rate (no OCS)	150	Active TS Rate	200	Active TS Rate	200	Active TS Rate	N/A
Active NIP sites	2	Achieved Gov. Rate	400	Achieved Gov. Rate	200	Achieved Gov. Rate	N/A
		Global Achieved Rate			600		

Table 1a shows the setting for the variables when the status of the entire NIP, that is the three sites and the common back end system is normal. Table 1b shows how these settings change when one site (site P) fails, and one transaction server fails at site N. Although for ease of illustration only a few SSP's are shown in the Figure, in a real network some hundreds of SSP's may be connected to three SCP sites. Accordingly, when one of the sites fails, every other SCP site will have a number of SSP's which switch to it as the second choice SCP site.

Figure 7 shows the physical implementation of the service platform of Figure 2. Of particular relevance to the invention are the DEC Alpha 4100 transaction servers and the DEC Alpha 4100 overload control servers. The governors and governor controllers are implemented as software modules running on these servers. These use a shared memory communications link termed

"Memory Channel" (MC). Memory Channel systems are available commercially from DEC. The memory channel is used to implement the node member change function NMCF. As noted above, this allows the governor controller to monitor how many active transaction servers there are. The memory channel also provides an efficient means for the transaction servers to communicate with central services on the OCS such as the governor controller and the OCF. Figure 6 illustrates the operation of the memory channel network. The Memory Channel technology provides a lock facility as part of its API (application program interface). MC guarantees (within a set time) to release a lock if the node (server) fails. The governor controller is pre-programmed with the identities of transaction servers. The governor controller regularly attempts to grab the lock belonging to each of the transaction servers. If it manages to get one of the locks it knows that a previously active transaction server has now failed. If it can no longer obtain a lock (while previously it could) it knows that a transaction server is now active.

In this implementation, the following information flows are communicated via the memory channel: the pseudo and real rejection thresholds; the count of an offered call and its associated SSP ID. The threshold values are simply data items held in the memory channel. Governor controller software on the OCS writes to the values while software methods on the transaction servers allow the values to be read. For each offered call, software on the transaction server accesses a common count of calls and then, while locking the count, increments the value of the count.

In the examples described above, it is assumed that each call made to the SCP results in approximately the same load for the SCP. This approximation gives acceptable results even where there is some variance in the loading associated with different calls. For example, some calls may involve simple number translation, in which case the SCP handles an initial DP followed by FurnishChargingInformation, SendChargingInformation and Connect signals, while other calls may combine number translation with a call statistics operation. (These events and signals, and other such events and signals referred to below, are formally defined in the ITU INAP standard, and the terminology used here is taken from that standard.) In this latter case, an extra RequestReportBCSMEvent operation is required. This results in an extra message, EventReportBCSM, being sent from the SSP to the SCP to indicate the state of the call. This involves a relatively small increase in the processing load on the SCP. As a result the governor can function effectively simply by adding a single drip to the leaky bucket counter for each InitialDP, irrespective of the call type.

In some situations, the SCP may be required to handle different calls which impose widely varying loads. This is the case when some of the calls involve multiple legs. Such calls involve arming detection points at the SSP, using the RequestReportBSCMEvent, as *interrupted.* In this case, when the SCP triggers a detection point, e.g. for busy, then the control of the call remains with the SCP. The SCP in this case may have multiple interactions with the SSP following a single InitialDP (initial Detection Point) event. Furthermore, the service may now involve the use of an Intelligent Peripheral, such as a voice messaging platform. The SCP then has to set a temporary leg to the Intelligent Peripheral to allow voice interaction. At the same time, other calls signalled to the SCP may still involve nothing more than simple number translation.

In the scenario described in the preceding paragraph, if the governor is programmed on the assumption that each initialDP is associated with the processing load of a simple number translation, then when the actual received calls have multiple legs the capacity of the SCP is likely to be exceeded. On the other hand, if the rate limits are set on the assumption that each InitialDP is associated with a multiple-legged call, then when the majority of the calls are for simple number translation the resources of the SCP will be under-utilised. To overcome these problems, in an alternative implementation of the invention, the governor is programmed to add a drip to the leaky bucket counter not only when an InitialDP message is received at the SCP, but also when other operations are sent to or from the SCP, e.g. when an EventReportBCSM message is received provided that the event report is an interrupted EDP (event detection point). Since the call has already been admitted at this stage, any resulting rejection event is of the pseudo-rejection type. When the service involves legs to an intelligent peripheral, then the occurrence of an EstablishTemporaryConnection operation is treated as a further trigger for incrementing the leaky bucket counter. In this case the trigger relates to a signal sent from the SCP to the SSP, rather than vice versa as in the other examples. Again, any reject event is a pseudo-reject.

A further feature of this alternative implementation uses a variable drip - that is to say the leaky bucket counter is incremented by different amounts depending on the nature of the event which caused the drip. A drip associated with an InitialDP may have a value of 2, while a drip associated with an EventReportBCSM may have a unit value of 1. Determining the appropriate drip values for different events, and the threshold for the governor, may be carried out by prototyping and testing a particular platform and/or from modelling using representative service times for different types of call. In principal any or all of the intermediate signalling events occurring at the service platform may be used to trigger the adding of a drip to the leaky bucket counter. For ease of implementation, all such events may be treated as potentially adding a drip, but for selected events the weight of the drip may be set to zero.

Figure 8 is a flow diagram for a governor which use variable-weight drips, as described in the immediately preceding paragraphs. For the sake of clarity, the periodic dripping from the leaky bucket counters is omitted from the illustrated event flows. This may be viewed as a concurrent external event. At step s1, a message event occurs at the governor as it receives or transmits a signal via the network interface. The governor determines the event type, step s2. If the event is determined to be an InitialDP (IDP) then the value of the drip weight is set, e.g., equal to 3, step s3. The value of a variable r_currentlevel, which is the current level of the leaky bucket counter for real rejection events, is read, step s4. The drip weight is added to r_currentlevel and the result is written to a temporary register "temp", step s5. In step s6 the value of temp is compared with real_l, that is the limit value for real rejection events. If the limit value is exceeded then in step s7 the governor sends notification of a real rejection event to the OCS. In step s8 the governor causes the call to be released. If the limit value is not exceeded, then in step s9 (Figure 8iii) the drip is added to the bucket as the variable r_currentlevel is set equal to the value held in the temporary register. The value p_currentlevel of the level of the leaky bucket counter for pseudo rejection events is read, step s10, and the drip weight is added to that value and is written into the temporary register, step s11. In step s12, the value of temp is compared with pseudo_l, that is the limit value for pseudo-rejection events. If the limit value is exceeded, then in step s13 the governor sends notification of a pseudo rejection event to the OCS. If the limit value is not exceeded, then in step s14 the drip is added to the bucket as the variable p_currentlevel is set equal to the value held in the temporary register.

Intermediate events, subsequent to an IDP event and the admission of a call, are used to add to the level in the leaky buckets for real and pseudo rejection. For example, if the intermediate event is an EstablishTemporaryConnection (ETC) signal sent from the SCP to the SSP, then the drip weight is set equal to, e.g., 2, step s15 (Figure 8ii). For another intermediate event, EventReportBCSM (ERBE) the drip weight is et equal, e.g., to 1, step s16. Similarly other weights may be set for other intermediate events. In steps 17 to 20, the appropriate drip values are added to both the leaky buckets. The governor, in this example, adds drips to the leaky buckets for all event flows between the SCP and SSP. These events comprise:

From SSP to SCP

- InitialDP
- EventReportBCSM
- ActivityTest result

and from SCP to SSP

- FurnishChargingInformation
- SendChargingInformation
- Connect
- ActivityTest
- EstablishTemporaryConnection
- RequestResportBCSMEvent

Every one of these operations has an assigned drip value. The InitialDP is a special case as it can be rejected without dripping. As already noted, some events may be assigned a zero value for the drip.

Figure 9 shows a preferred implementation of the invention using different drip weights. The SCP in this example includes a transaction server call processing module 91, a Transaction Capabilities Application Part (TCAP) server 92, a TCAP back end processor 93 and a communications server 94. A transfer module 95 links the TCAP server 92 to an Advanced Transaction Server 96 which is responsible for certain advanced transactions. As in the previously described embodiments, the TCAP server includes a governor which uses a leaky bucket algorithm.

The load on the transaction server is due to he processing of mesages crossing the six interfaces labelled 1 to 6 in the figure, and hence that load can be modelled as a weighted sum of message rates across these interfaces. Similarly signalling network load can be modelled as a weighted sum of messages across the 2 interfaces labelled 1 and 4 in the figure. Drip weights in the governor are set to be proportional to the amount of load implied by each message. There is a configurable drip size defined for each message type at each interface. Table 2 below lists the different messages for each of the interfaces (i/f) shown in the Figure.

Table 2

i/f	**Messages**
1	IDP, EventReport, ETCFail, Abort, Error
2	IDP, EventReport, INAP Control
3	FCI, SCI, Connect, RequestReport, INAP Control
4	FCI, SCI, Connect, RequestReport, ETC, ReleaseCall, CallGap, Error, Abort
5	IDP, EventReport, ETCFail, INAP Control
6	FCI, SCI, Connect, RequestReport, ETC, ReleaseCall, INAP Control

This requires the provision of 5+3+5+9+4+7=33 configurable drip sizes. In a relatively simple implementation of the invention only the IDP (initial detection point) at interface 1 contributes to the level in the governor leaky bucket and its drip size is unity. In an implementation where the only permissible way for the TS to shed load is to return ReleaseCall for an incoming IDP, rather than passing the IDP to TS Call Processing, then the real-rejection Governor leaky bucket cannot fill beyond the level at which IDPs are rejected, because only IDPs which are accepted result in a drip into the bucket. However in preferred implementations, follow-on messages for a call at any interface are not rejected, even if the governor leaky bucket level is greater than the level at which incoming IDP's ("new load") are rejected.

As an example, in an implementation as shown in Figure 9, for a service which, e.g., is handled by the ATS, drip weights (termed here "sizes") for different messages and different interfaces may be set as follows: Using the notation <message>(<interface_number>) to specify a drip size, the drips added have the following weighting matrix IDP(1), IDP(2), INAP Control(3), IDP(5), SCI(6), FCI(6), Connect(6), SCI(4), FCI(4), Connect(4). Here the INAP Control message is the message which indicates to the TCAP Server that the message should be passed to an ATS.

As a second example, consider the drips into the governor bucket as a result of an IDP which is rejected because the Governor bucket level is above the threshold level at which IDPs are rejected. Then the drips added have sizes IDP(1), ReleaseCall(4). This models the load more accurately than implementations which in which rejected calls are assumed to result in zero loading of the Transaction Server.

In implementing a platform in accordance with the invention and using a leaky bucket in the governor, it is not necessary to leak the bucket every time a message-related drip is added to it. It is only necessary to determine the correct level when an IDP arrives and a decision must be made whether to process, or to release, the call. Thus the bucket should be leaked immediately before this decision is to be made, but need not be leaked at any other time.

The bucket fill level, drip sizes, and the amount drained out at each leak time are all intrinsically real numbers, not integers. For maximum flexibility in setting drip sizes and for clarity and maintainability of code, it is desirable to represent them as *float* or *double* types, rather than using integers with scaling. With floating-point hardware this is very unlikely to incur a performance penalty. Only tests including an inequality (>, <, >=, <=) are then used when comparing two floating point numbers: tests for equality are not reliable.

In the implementation discussed above with respect to Figure 9, a weighting matrix is used with two dimensions corresponding to interface location and message event identity. The weights in that example are determined in relation to a single assumed bottleneck location, for example at the TCAP server. The capability of the governor can be further extended by considering a number of different potential bottlenecks, any one of which may become the true bottleneck depending on call mix. There are then two weighting matrices to be used in determining drip weights (or equivalently a single three-dimensional matrix) one having dimensions of message-type and interface which relates to the load implied at the local box (here the Transaction Server) by messages across the on-box and off-box interfaces, the other having dimensions of message-type and bottleneck, which relates to load implied at remote bottlenecks by messages across the interfaces which communicate with that bottleneck.

For example, a reporting engine may be the bottleneck if all calls require real-time reports on their success or failure; signalling network processing for routing may be the bottleneck if there are large numbers of short signalling messages; signalling link bandwidth may be the bottleneck if average signalling message length is large.

Multiple bottlenecks are protected by the governor as follows: for each potential bottleneck, evaluate the loading at that bottleneck implied by each message. As an example, the loading at a reporting engine may be due only to INAP Event Reports received in a NotifyAndContinue mode, thus only this message type would have a nonzero weight; loading of signalling routing processors would be equal for all messages; and loading of signalling links would be in proportion to the length of the message.

The absolute values depend on the chosen (arbitrary) leak rate of the Governor bucket and on the capacity of the bottleneck: for example, if the reporting engine can handle 200 call outcome reports per second, and the Governor leak rate is 1.0 per second, each Event Report should have a weight of 0.005. If a routing processor in the signalling network can handle 250 messages per second and there are 4 such processors in a load-sharing scheme, each message should have a weight of 0.001.

The result is a matrix of governor drip weights, with (say) columns indexed by bottleneck and rows indexed by message type. To protect against congestion at all potential bottlenecks, choose for each message the maximum weight appearing in its row, i.e. the weight corresponding to the bottleneck at which that message implies the greatest loading. It is then this value which is substituted in the interface/message-type matrix previously described. Accordingly different entries in the interface/message matrix are then determined by different potential bottleneck locations.

Using the simple example above, if the reporting engine and the signalling routing processors were the only two potential bottlenecks, then Event Reports received in NotifyAndContinue mode should be assigned a weight of 0.005 (the maximum of their two weights of 0.005 and 0.001) and all other messages should be assigned a weight of 0.001 (the maximum of their two weights of 0 and 0.001). The reporting engine bottleneck in the example would have a load implied by event reports across the interfaces from TCAP Server to TSCP and TS to ATS. The signalling network bottleneck would have a load implied by all the messages across the network interface to TCAP Server. These different weights may be implemented by a three-dimensional weighting matrix with dimensions of message type, interface index, and bottleneck index.

Although described above in relation to leaky bucket counters, the principle of using different weights in counting signalling events in a transaction processing system may also be used in implementations in which other forms of rate monitoring are used. Also differential weighting may be used with leaky buckets to protect network resources in other contexts, apart from their use in the governor of the present invention. For example they may be used in outbound rate control for prevention of focussed overload at destination number in a public switched telephone network (PSTN), as described in the present applicant's European patent EP-B-334612, granted 4 Aug 93, priority date 21 March 1988, the contents of which are incorporated herein by reference.

CLAIMS

1. A transaction processing system for use in a communications network, the system including:

a) a network interface for communicating signalling for inbound traffic

b) a governor programmed with a static limit value for the rate of traffic-related signalling, and

c) a transaction processor which is responsive to the governor and which is arranged to release an inbound transaction when the static limit value is reached.

2. A transaction processing system according to claim 1, further comprising:

d) an overload controller which is connected to the governor and which is arranged to output control signals onto the network to reduce the rate of inbound transactions to the transaction processing system when the static limit value is exceeded.

3. A transaction processing system according to claim 1 or 2, in which the governor is arranged to respond to the total rate of both initial traffic-related signals and predetermined intermediate traffic-related signals.

4. A transaction processing system according to claim 3, in which the said predetermined intermediate traffic-related signals comprise signals associated with the setting of an interrupt detection point.

5. A transaction processing system according to claim 4, in which the said intermediate traffic-related signals comprise interrupt *EventReportBSCM* messages.

6. A transaction processing system according to any one of claims 3 to 5, in which the governor is arranged to weight differently the contribution to the total call rate of the said initial traffic-related signals and the contribution to the total call rate of the said intermediate traffic-related signals.

7. A transaction processing system according to 2, or any one of claims 3 to 6 when dependent on claim 2, in which the governor is programmed with a first, lower static limit value, and a second, higher static limit value, and the governor is arranged to signal to the overload controller when the first, lower threshold is reached, and is arranged to release an inbound call only when the second, higher threshold is reached.

8. A transaction processing system according to claim 2 or 7, in which the said control signals comprise call gap messages which are addressed to a switch which, in use, is connected to the transaction processing system.

9. A transaction processing system according to any one of the preceding claims, further comprising:

e) a governor controller which is connected to the governor and which is arranged to set automatically the value for the static limit value.

10. A transaction processing system according to claim 9, comprising a plurality of processing systems, each of the processing systems comprising a transaction processor and a governor which is connected to the respective transaction processor,

and in which the governor controller is connected to the plurality of processing systems and sets independent respective static limit values in the different respective governors.

11. A transaction processing system according to claim 10, in which the governor controller is programmed with a target transaction rate for the system, and is arranged to set the static limit values in the different respective governors so that the total admitted transaction rate substantially matches the said target transaction rate.

12. A transaction processing system according to claim 11, in which the governor controller includes a monitor input which is connected to the plurality of processing systems and which receives status data for the processing systems, and in which the governor controller is arranged to amend the static limit values in the different

respective governors when the said status data indicates that the number of functioning transaction processors has changed.

13. A transaction processing system according to any one of claims 9 to 12, in which the governor controller is arranged to monitor the identity of network switches which, in use, direct inbound signalling to the transaction processing system, and in which the governor controller is arranged, when signalling is received from a switch which normally uses another transaction processing system, automatically to increase a target transaction rate for the transaction processing system.

14. A communications network including a transaction processing system according to any one of the preceding claims.

15. A method of operating a transaction processing system in a communications network, the method comprising:

a) communicating signalling for an inbound transaction between a communications network and a governor;

b) in the governor, comparing the current rate of calls processed by the transaction processing system with a static limit value; and

c) causing the inbound transaction to be released when the static limit value is reached.

16. A method according to claim 15, further comprising outputting control signals onto the network, and thereby reducing the rate of inbound calls to the transaction processing system, in response to a call rejection event triggered by the governor.

17. A method according to claim 16, including communicating a rejection event signal from the governor to an overload controller when a rejection event occurs, and subsequently outputting the said controls signals from the overload controller.

18. A method according to claim 17, including communicating the said rejection event signal to the overload controller when a first, lower static limit value is

reached, and both signalling to the overload controller and releasing the inbound call only when a second, higher static limit value is reached.

19. A method according to any one of claims 15 to 18, in which in step (b) the governor compares the total rate of both initial traffic-related signals and intermediate traffic-related signals with the said threshold.

20. A method according to claim 19, in which the said predetermined intermediate traffic-related signals comprise signals associated with the setting of an interrupt detection point.

21. A method according to claim 20, in which the said intermediate traffic-related signals include interrupt *EventReportBSCM* messages.

22. A method according to any one of claims 19 to 21, in which the contribution to the total call rate of the initial traffic-related signals and the contribution to the total call rate of the intermediate traffic-related signals are weighted differently.

23. A method, system or network according to any one of the preceding claims, in which the transaction processing system is a service platform for use in a telecommunications network.

24. A method according to any one of claims 15 to 23, in which the governor increments a leaky bucket counter in response to signals communicated between the system and the communications network.

25. A service platform for use in a communications network, the platform including:

a) a network interface for receiving signalling for inbound calls

b) a static governor programmed with a predetermined call rate threshold, and

c) a call processor which is responsive to the static governor and is arranged to release an inbound call when the predetermined call rate threshold is reached.

26. A method according to claim 24, in which the governor applies different weights to leaky bucket increments ("drips") associated with different respective message types passed in connection with the processing of the call.

27. A method according to claim 24 or 26, in which the governor applies different weights to leaky bucket increments ("drips") associated with the passing of messages over different respective interfaces in the platform in connection with the processing of the call.

28. A method according to claim 27 when dependent on claim 26, in which a weighting matrix having dimensions of message-type and interface identity is applied to the said leaky bucket increments , and in which different entries in the said matrix have a value determined for different bottlenecks in the processing of the signalling for the call.

29. A method according to any one of claims 15 to 28, including determing the current rate of calls processed by applying different weights to different signalling events associated with the processing of calls by the transaction processing system.

30. A method according to claim 29, in which different weights are applied for signalling events across different signalling interfaces in the transaction processing system.

31. A method according to claim 29 or 30, in which different weights are applied to signalling events corresponding to different message types.

32. A method of operating a communications network including:

a) comparing a rate of events in a network resource with a threshold rate by incrementing a leaky bucket counter in response to signalling events; and

b) applying different weights to the increments of step (a) for different signalling events, the weights depending on the degree loading of the network resource associated with the different signalling events.

33. A platform for use in a communications network including:

a) a leaky bucket counter arranged to be incremented in response to signalling events, in use the counter comparing a rate of events in a network resource with a threshold rate; and

b) means for applying different weights to the increments of the leaky bucket for different signalling events, the weights depending on the degree of loading of the network resource associated with the different signalling events.

34. A method according to any one of claims 24, and 26 to 32, including applying a three-dimensional weighting matrix having dimensions of message-type, interface index and bottleneck index.

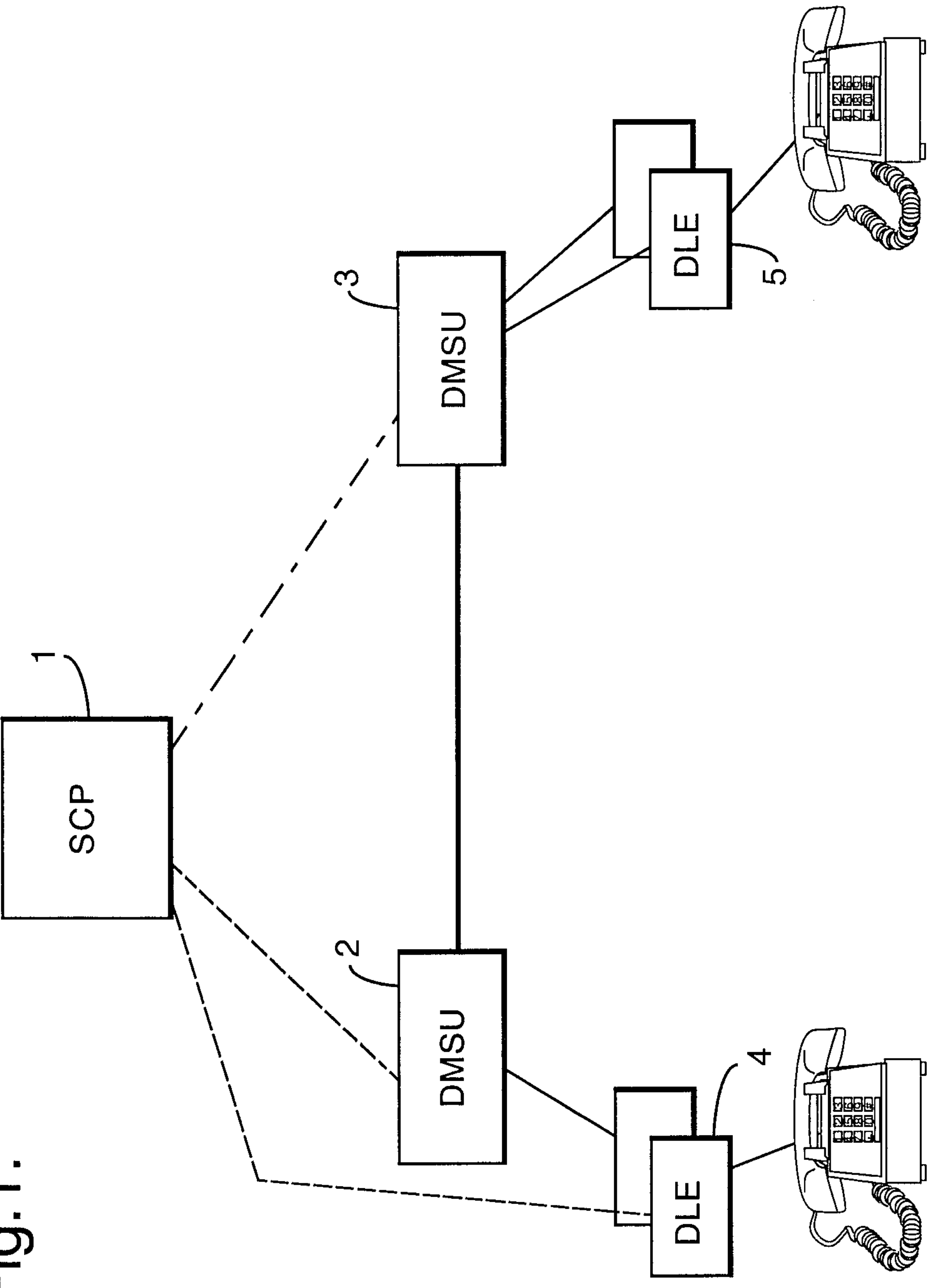


Fig.1.

Fig.2.

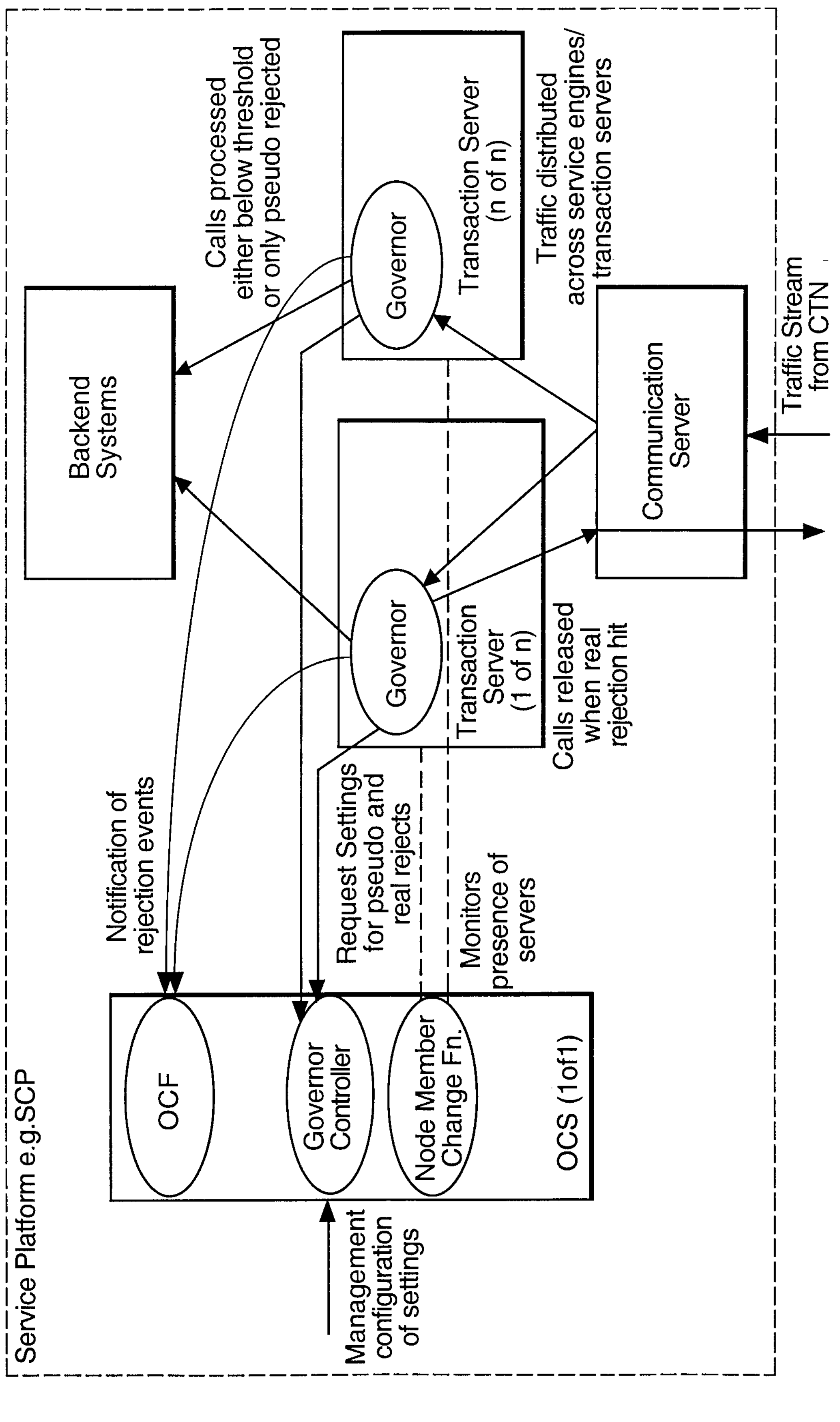
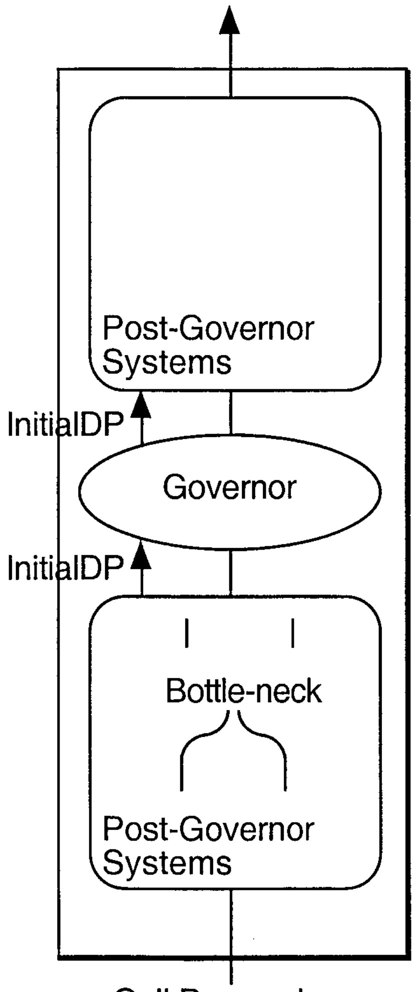

Service Platform e.g.SCP
Backend Systems
Calls processed either below threshold or only pseudo rejected
Governor
Transaction Server (n of n)
Traffic distributed across service engines/ transaction servers
Communication Server
Traffic Stream from CTN
Governor
Transaction Server (1 of n)
Calls released when real rejection hit
Notification of rejection events
Request Settings for pseudo and real rejects
Monitors presence of servers
OCF
Governor Controller
Node Member Change Fn.
OCS (1of1)
Management configuration of settings

Fig.3a.

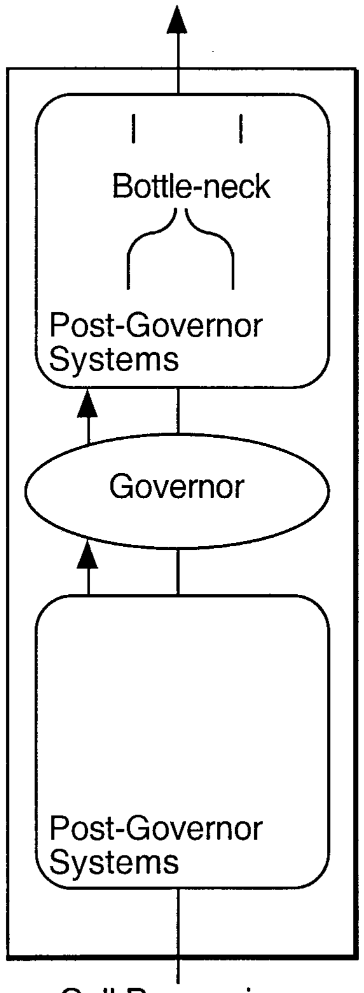
Pseudo-Rejection



Call Processing
call chain

Fig.3b.

Real-Rejection



Call Processing
call chain

Fig.4.

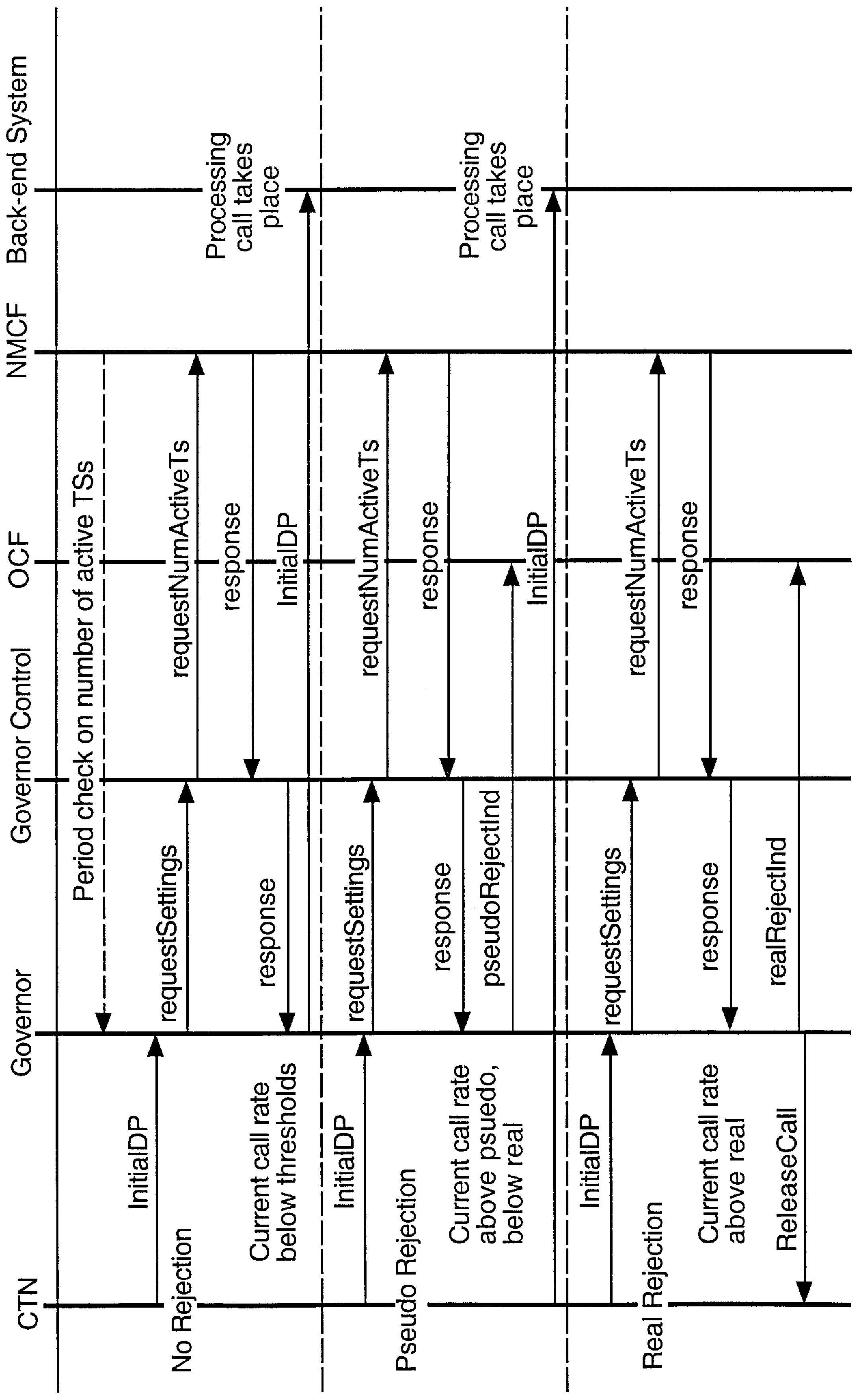

CTN
Governor
Governor Control
OCF
NMCF
Back-end System
Period check on number of active TSs
InitialDP
requestSettings
requestNumActiveTs
response
response
InitialDP
Processing call takes place
No Rejection
Current call rate below thresholds
InitialDP
requestSettings
requestNumActiveTs
response
response
pseudoRejectInd
InitialDP
Processing call takes place
Pseudo Rejection
Current call rate above psuedo, below real
InitialDP
requestSettings
requestNumActiveTs
response
response
realRejectInd
ReleaseCall
Real Rejection
Current call rate above real

Fig.5.

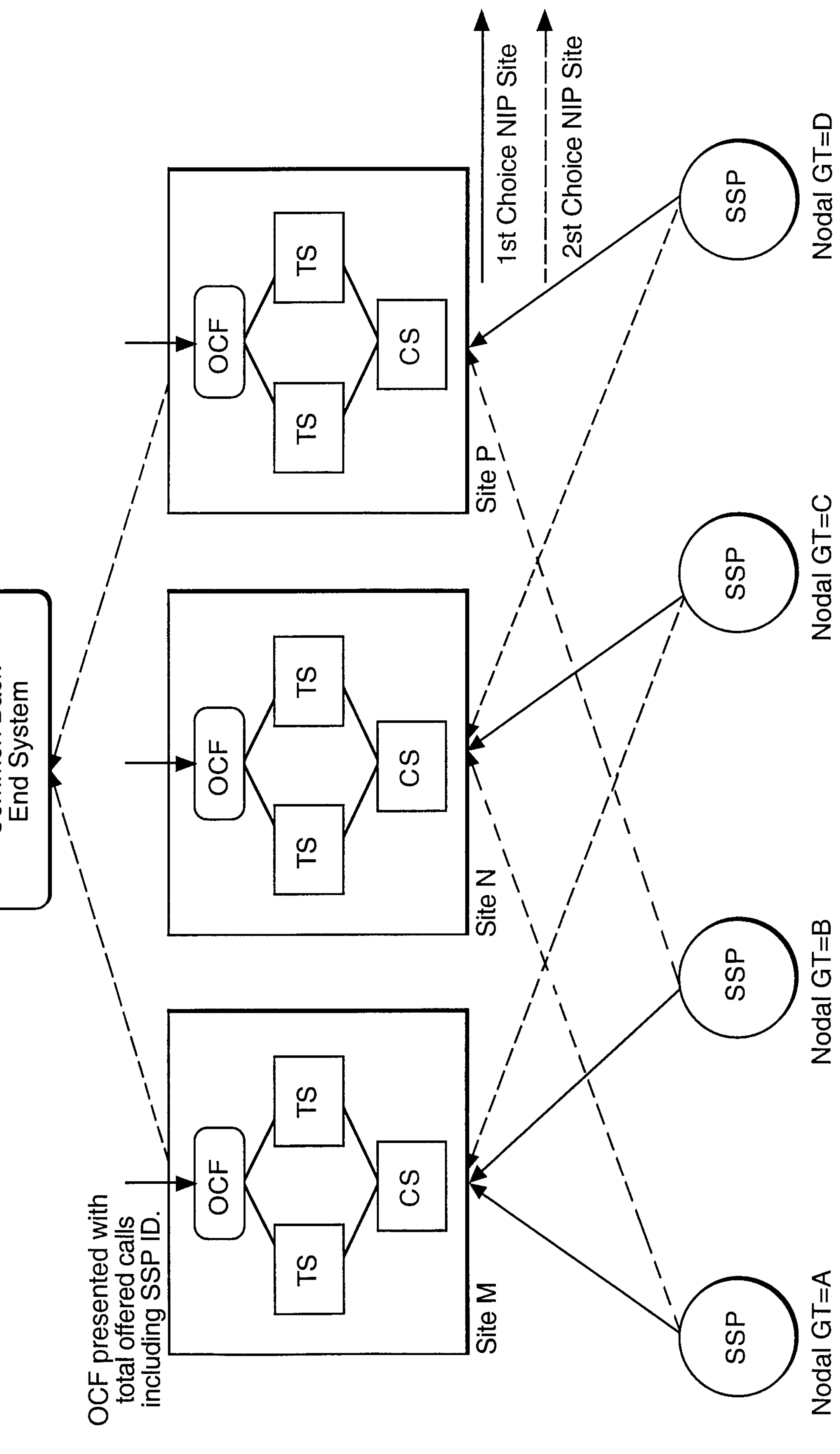
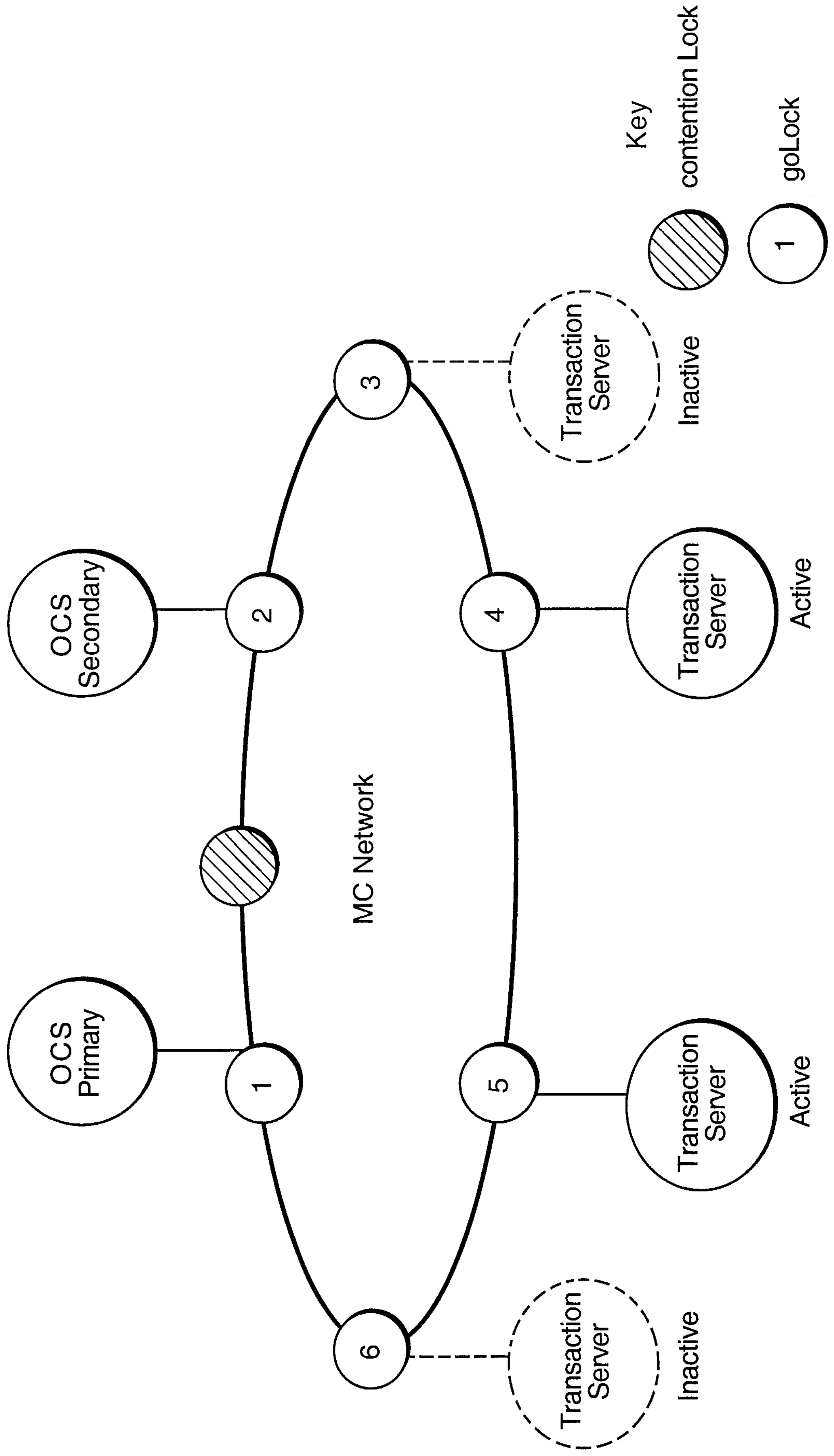

Common Back End System
OCF presented with total offered calls including SSP ID.
OCF
TS
TS
CS
Site M
Site N
Site P
1st Choice NIP Site
2st Choice NIP Site
SSP
Nodal GT=A
Nodal GT=B
Nodal GT=C
Nodal GT=D

Fig.6.


OCS Primary
OCS Secondary
MC Network
1
2
3
4
5
6
Transaction Server
Active
Inactive
Key
contention Lock
goLock

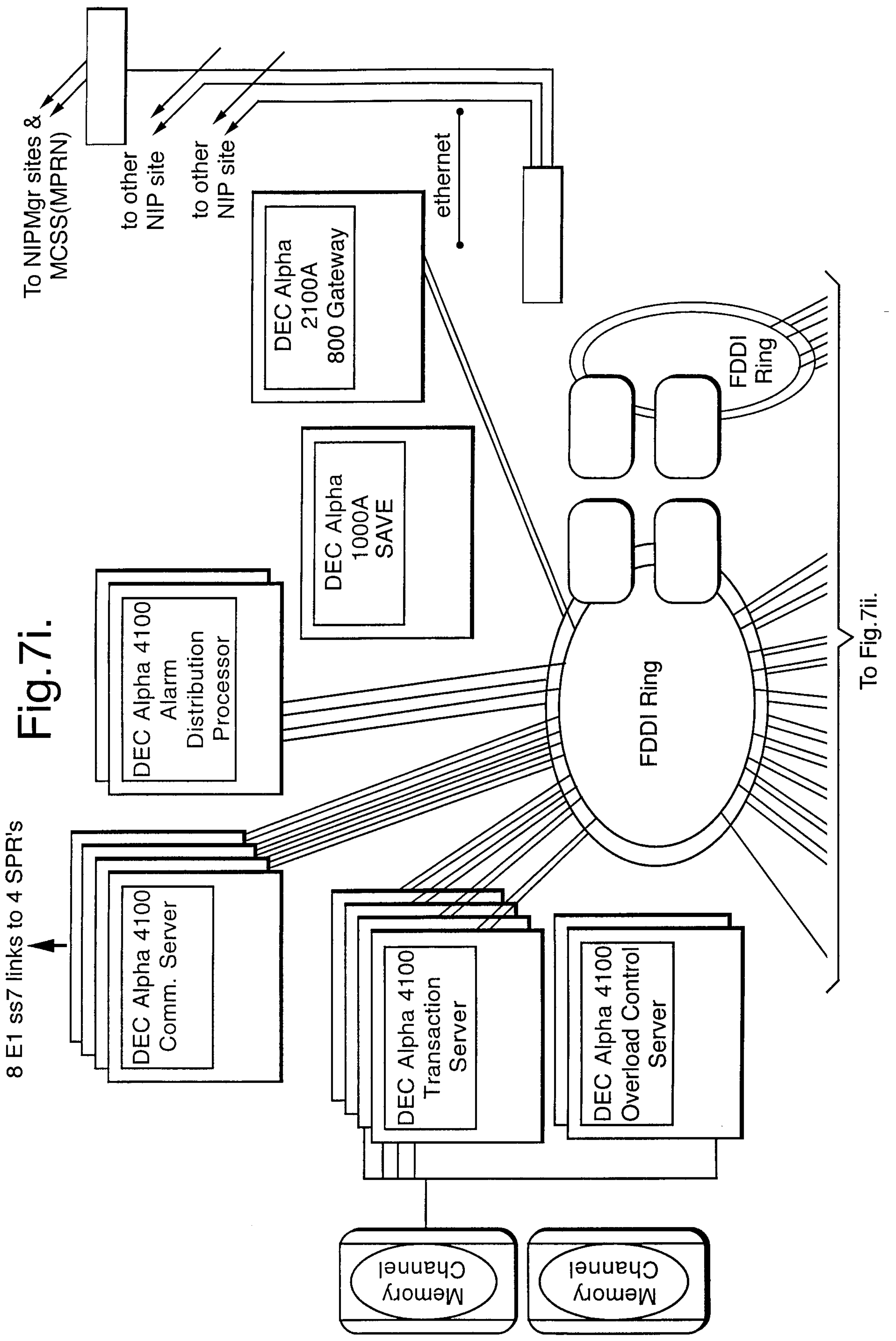

To NIPMgr sites &
MCSS(MPRN)
to other
NIP site
to other
NIP site
ethernet
DEC Alpha
2100A
800 Gateway
DEC Alpha
1000A
SAVE
FDDI
Ring
DEC Alpha 4100
Alarm
Distribution
Processor
FDDI Ring
To Fig.7ii.
8 E1 ss7 links to 4 SPR's
DEC Alpha 4100
Comm. Server
DEC Alpha 4100
Transaction
Server
DEC Alpha 4100
Overload Control
Server
Memory
Channel
Memory
Channel

Fig.7i.

Fig.7ii.

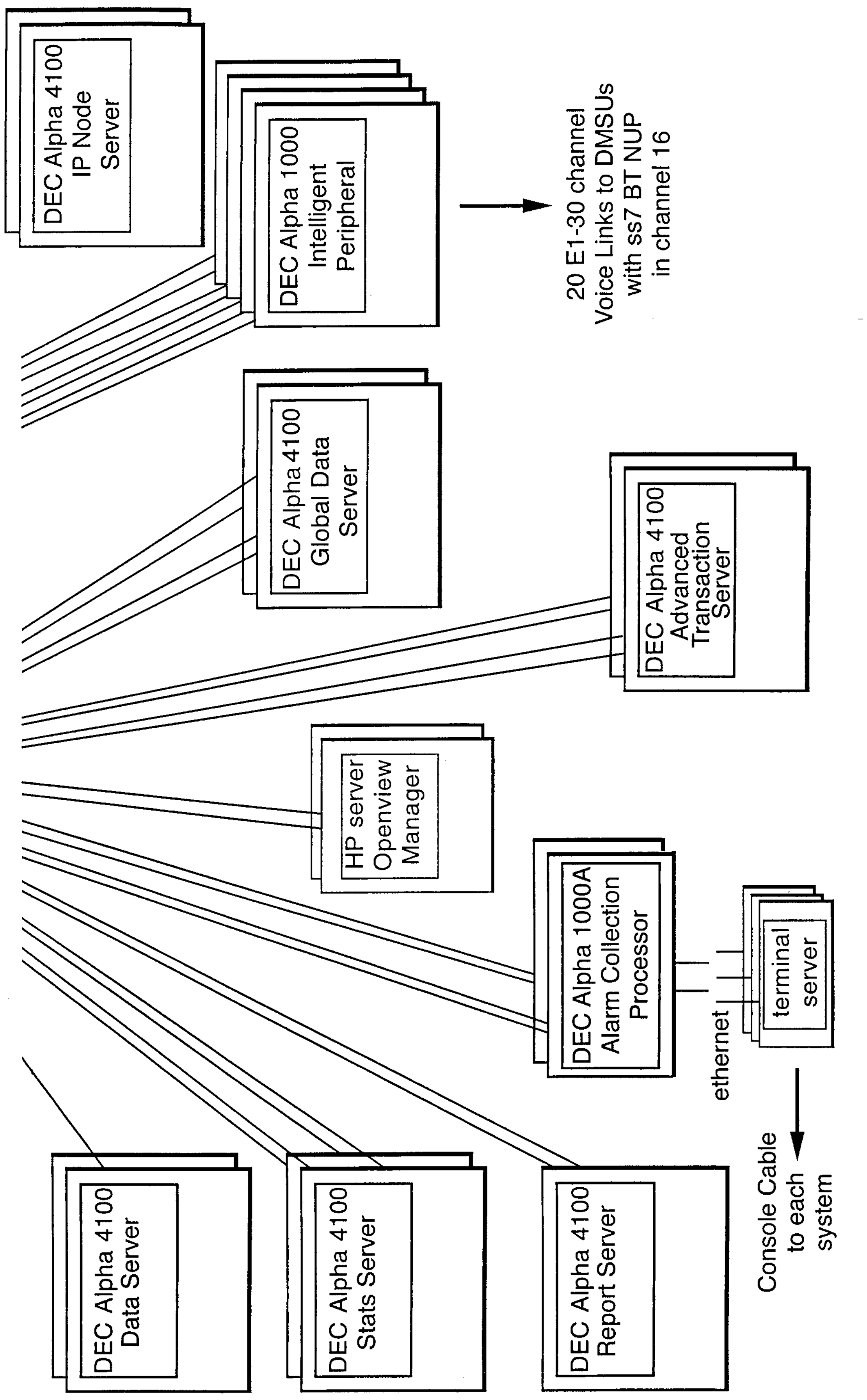
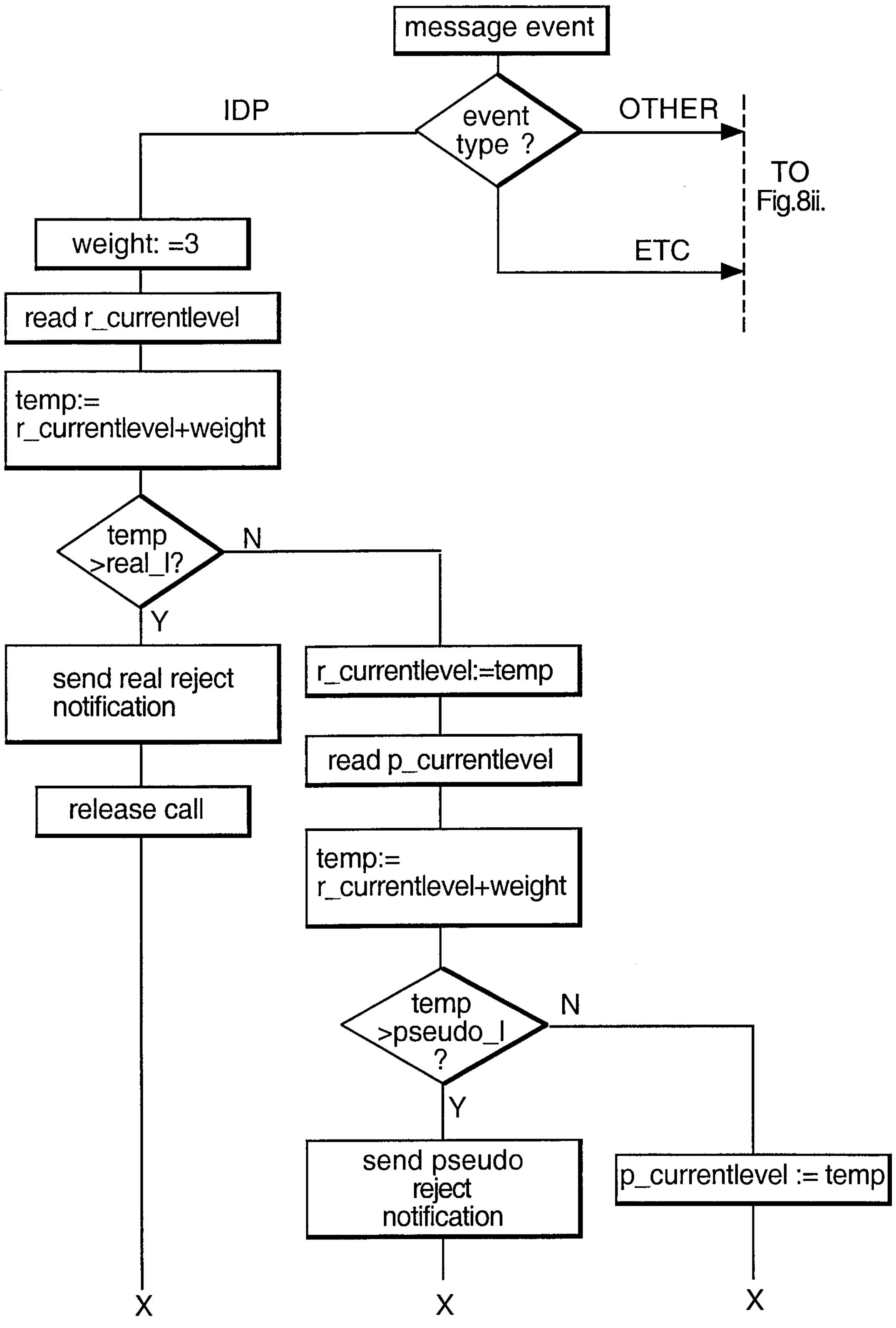

DEC Alpha 4100 IP Node Server
DEC Alpha 1000 Intelligent Peripheral
20 E1-30 channel Voice Links to DMSUs with ss7 BT NUP in channel 16
DEC Alpha 4100 Global Data Server
DEC Alpha 4100 Advanced Transaction Server
HP server Openview Manager
DEC Alpha 1000A Alarm Collection Processor
ethernet
terminal server
Console Cable to each system
DEC Alpha 4100 Data Server
DEC Alpha 4100 Stats Server
DEC Alpha 4100 Report Server

Fig.8i.



message event
event type ?
IDP
OTHER
ETC
TO Fig.8ii.
weight: =3
read r_currentlevel
temp:= r_currentlevel+weight
temp >real_l?
N
Y
send real reject notification
release call
r_currentlevel:=temp
read p_currentlevel
temp:= r_currentlevel+weight
temp >pseudo_l ?
N
Y
send pseudo reject notification
p_currentlevel := temp
X
X
X

Fig.8ii.

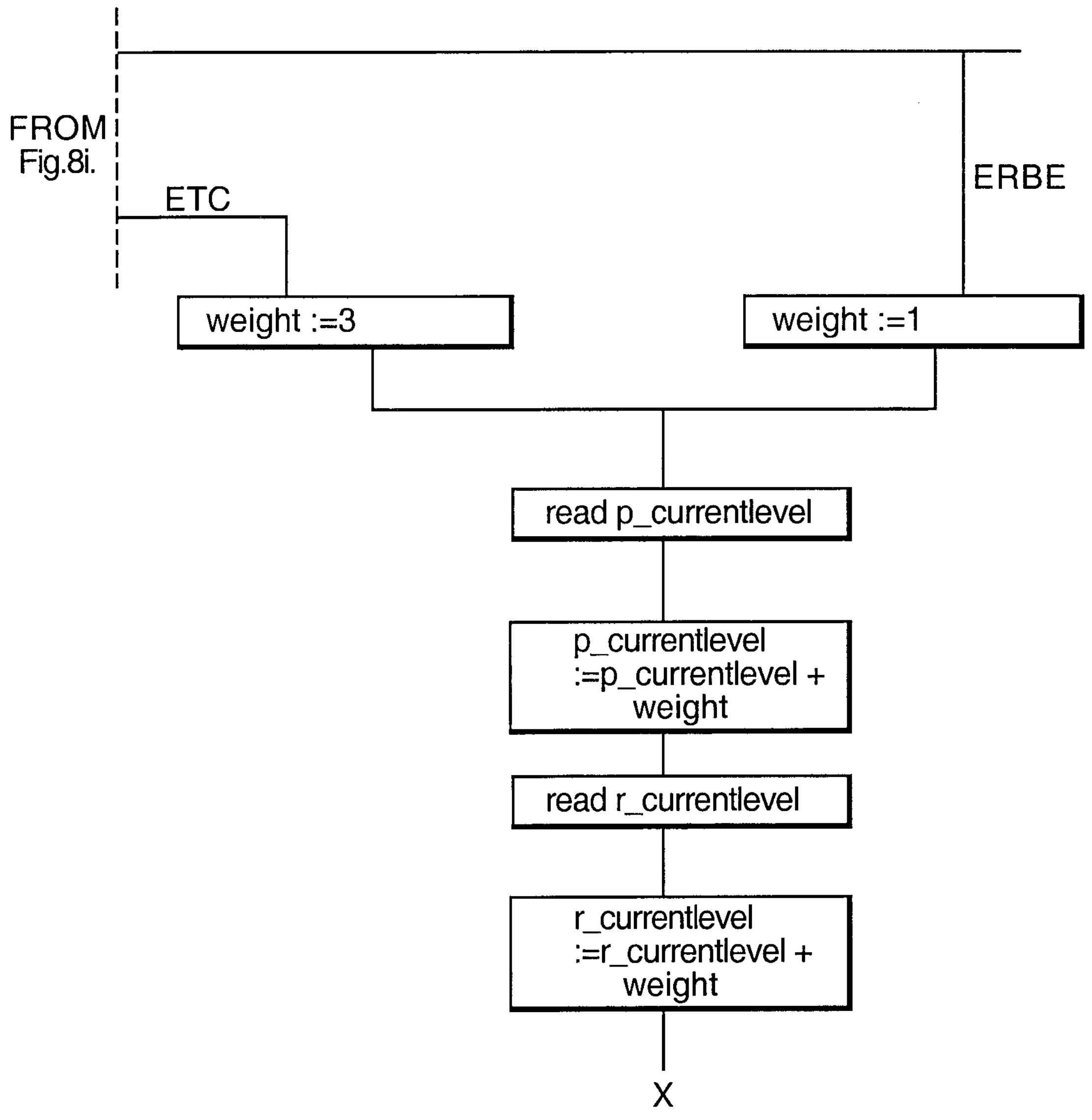
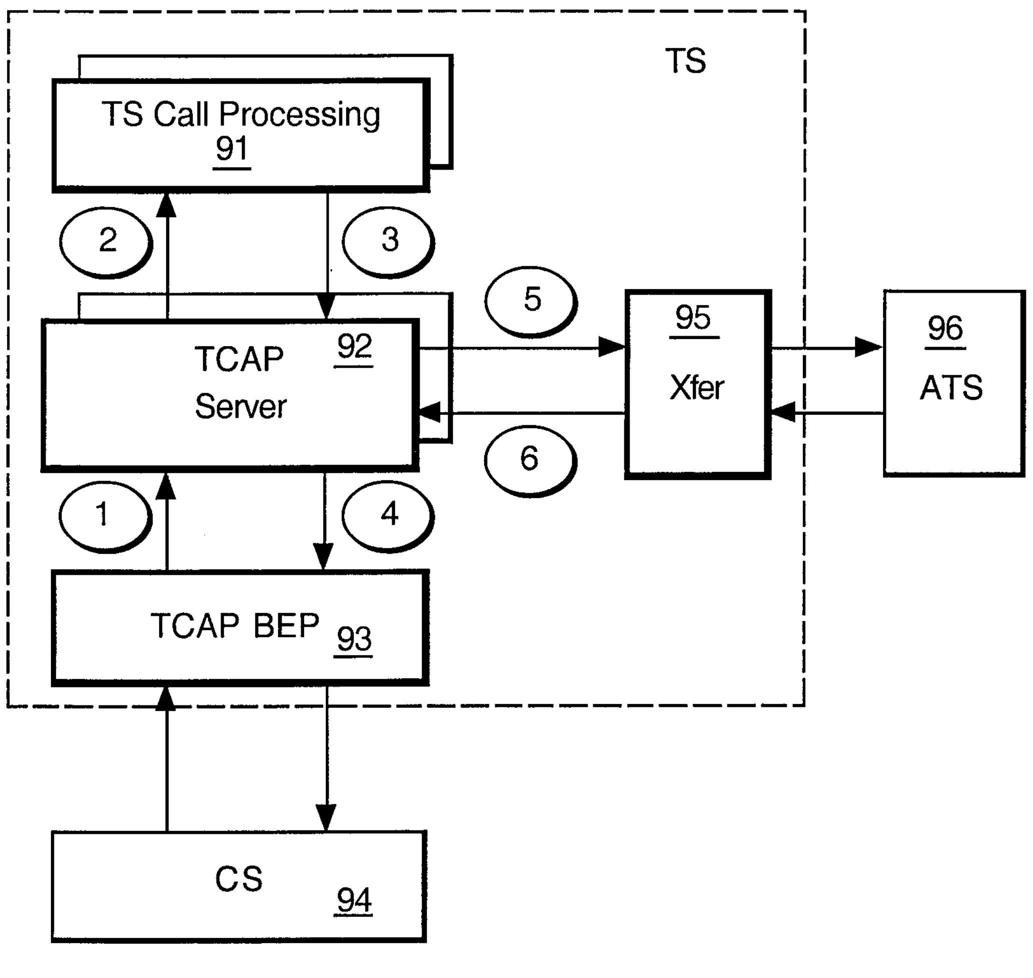

FROM
Fig.8i.
ETC
ERBE
weight :=3
weight :=1
read p_currentlevel
p_currentlevel :=p_currentlevel + weight
read r_currentlevel
r_currentlevel :=r_currentlevel + weight
X

Fig.9.


TS
TS Call Processing 91
2
3
5
TCAP Server 92
95 Xfer
96 ATS
6
1
4
TCAP BEP 93
CS 94

INTERNATIONAL SEARCH REPORT

ıational Application No
PCT/GB 98/01628

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04Q3/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H04Q

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 96 15634 A (NOKIA TELECOMMUNICATIONS OY ;GINZBOORG PHILIP (FI)) 23 May 1996	1-3,8,9, 14-17, 19,23,25
Y	see page 8, line 5 - page 9, line 20 see page 12, line 7 - page 13, line 10 see page 22, line 19 - line 28 ---	10-13
X	US 5 425 086 A (HIDAKA TAKASHI ET AL) 13 June 1995	1,2,9
Y	see page 1, line 67 - page 2, line 57 ---	10-13
X	WO 97 09814 A (ERICSSON AUSTRALIA PTY LTD ;MELBOURNE INST TECH (AU); UNIV MELBOUR) 13 March 1997 see page 2, line 4 - page 4, line 17 see page 22, line 24 - page 23, line 3 --- -/--	1-3,8,9, 14-17, 19,23,25

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search 4 September 1998	Date of mailing of the international search report 14/09/1998
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Chassatte, R

1

INTERNATIONAL SEARCH REPORT

I. .ational Application No
PCT/GB 98/01628

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 735 786 A (SIEMENS AG) 2 October 1996	1-3,8,9, 14-17, 19,23
Y	see claim 1	24-27, 29-33

Y	KOSAL H ET AL: "A CONTROL MECHANISM TO PREVENT CORRELATED MESSAGE ARRIVALS FROM DEGRADING SIGNALING NO. 7 NETWORK PERFORMANCE" IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, vol. 12, no. 3, 1 April 1994, pages 439-445, XP000458689 see tables I,II see abstract see paragraph I. see paragraph II.	24-27, 29-33

A	US 5 581 610 A (HOOSHIARI ALIREZA) 3 December 1996 see column 4, line 35 - column 7, line 44 see column 11, line 29 - line 30	1-34

A	RUMSEWICZ M P: "CRITICAL CONGESTION CONTROL ISSUES IN THE EVOLUTION OF COMMON CHANNEL SIGNALING NETWORKS" FUNDAMENTAL ROLE OF TELETRAFFIC IN THE EVOLUTION OF TELECOMMUNICATI NETWORKS, PROCEEDINGS OF THE 14TH. INTERNATIONAL TELETRAFFIC CONGRESS - ITC 1 JUAN-LES-PINS, JUNE 6 - 10, 1994, vol. 1A, 6 June 1994, pages 115-124, XP000593405 LABETOULLE J;ROBERTS J W (EDS) see the whole document	

A	SMITH D E: "ENSURING ROBUST CALL THROUGHPUT AND FAIRNESS FOR SCP OVERLOAD CONTROLS" IEEE / ACM TRANSACTIONS ON NETWORKING, vol. 3, no. 5, 1 October 1995, pages 538-548, XP000543255 see the whole document	

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

Iational Application No
PCT/GB 98/01628

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9615634 A	23-05-1996	FI 945332 A	12-05-1996
		AU 3872695 A	06-06-1996
		BR 9509656 A	14-10-1997
		EP 0791276 A	27-08-1997
US 5425086 A	13-06-1995	JP 5073371 A	26-03-1993
		JP 5145959 A	11-06-1993
		CA 2078497 A	19-03-1993
WO 9709814 A	13-03-1997	AU 6782996 A	27-03-1997
EP 0735786 A	02-10-1996	CN 1135133 A	06-11-1996
US 5581610 A	03-12-1996	AU 691332 B	14-05-1998
		AU 3436395 A	02-05-1996
		DE 19538804 A	15-05-1996
		FR 2726142 A	26-04-1996
		SE 9503641 A	20-04-1996