A computer controlled system.

Technical Field of the Invention.

The present invention generally relates to a computer controlled system, e.g. of the kind included in a network element in a telecommunication network.

A telecom network consists of network elements i.e. systems, that cooperate to provide the network users with network services. The many different services supported by a network element, is to a large extent realized by software, which is stored and processed by a computer system within the network element. The computer system stores data relevant to network and services, controls services, interacts with cuustomer premises equipment and with other network elements, and controls the local transmission and switching hardware.

Looking at the computer system from the view of physical equipments, the computer system hardware comprises mainly a number of communicationwise interconnected and cooperating processors and mass storage devices, divided between a number of different equipment assemblies.

Description of Related Art.

EP 405,829 relates to a telecommunication system in which the software is implemented by means of independent software components in the form of objects. A function runtime linker in a runtime system records the objects and stores a data pointer to the data of the objects. To communicate with another object, a source object transmits a message to the runtime system. The message includes name and identity of the method of the destination object.

The runtime system serves only a single processor or group of objects and calls the destination object by means of the identity of the method and the data pointer if the destination object belongs to a group of objects served by the runtime system. In case a destination object is located on another

processor, the runtime system broadcasts the message to other processors. In each of the receiving processors, the runtime system searches its linker table for the symbolic name of the destination object of the message and, if found, calls the destination object on the basis of the method identification in the message and the data pointer information in the runtime linker. Interprocessor messages include a source processor designation and the runtime system of each of the processors stores the name of the source processor and the symbolic name of the source object when an interprocessor message is received.

In US 4,901,231 there is described a multiprocessor system executing over a plurality of processors. A user process in one processor may access system resources in the other processors. When a user process accesses a local file the access is made i.a. by means of a user file table. When the user process accesses a remote file the access is made via a port table, over a virtual channel identified by means of the port table to a part process and then via the user file table and system file table of the part process.

US 5,187,790 relates to a computer system with a plurality of simultaneously running processes, including at least one server process and a plurality of client processes. Each process has a list of identities representing object access rights. Each object has an access checking list with identities to be used for determining the processors which may access the object.

Summary.

Any computer controlled system has a control program executing on at least one processor. Today a great number of requirements are made with respect to such a system among which can be mentioned that the application development should be easy, the system should have extremely high availability, and it should be easy to upgrade its capacity etc. A way of complying with such requirements is to have it designed with a layered architecture, where applications are programmed on a

control system platform, which is supposed to care for most common issues, leaving only the problems of the application domain to the application programmers.

In such an architecture, application programs, and much of the operating system software as well, are executed as cooperating processes.

Fitting processors into the process architecture is a bit cumbersome. When looking at the processors merely as resource providers the operating system is wanted to hide the individual processors from the application programs and give an image of just one large processor. On the other hand, the fact that there are different processors - with their own characteristics and their specific abilities to control the hardware devices that connect the system to the outside world - is the most important condition with respect to modelling of application programs into different processes. When dealing with processors it is also necessary to configure the system in a way that the resources provided by the processors are utilized as efficiently as possible. This should, however, not stop the control system from hiding the individual processors as far as possible to the application programmers.

Therefore it is desirable, from a programmer's point of view, that the control system can be seen as a homogenous platform, offering execution and memory resources to the processes, but also different services, such as

- a database for persistent data storage,
- a framework for implementing managed objects,
- normal operating system services like timers, and a clock,
- management of the application programs,
- control of start and restart procedures, error recovery etc.

The computer controlled system according to the invention includes a processing architecture, comprising a control system platform composed of a number of control system processors for handling different parts of said database, an operating system which manages resources provided by the processors and

implements abstractions provided by the control system, and representations of process resources. The processing architecture provides for application programs to be executed as cooperating processes interacting with each other, with the control system and with the world outside this architecture. Instances of the process resource representations are distributed over the control system processors by means of distribution units, each distribution unit forming a group of process instances. The distribution units belong to specific distribution unit types, of which each distribution unit type describes common properties of distribution units being of this type.

By a process is here meant an independently executing unit, meaning that it executes program code regardless of what other processes in the system might be doing at the same time. Each process has its own, private state, that is, a process cannot directly operate on data belonging to another process.

A process can be of a specific type, forming an instance of its process type. The process type determines what function the process has, processes of different types do different things, while instances of the same type do the same thing but operate on different data or on different hardware units.

The process instances can include physical process instances that execute program code, and logical process instances in the form of an abstract concept. The kind of logical process instances in a system is determined by the process types existing in the system, and their properties, and by the configuration of the system in terms of hardware devices.

Distribution units may group both logical and physical process instances. Logical instances are grouped because the relation between instances and distribution units is expressed in terms of the logical instances, and physical instances because when a distribution unit is allocated to a processor all physical process instances related to the distribution unit are created on that particular processor.

The processes can furthermore include central process

types, there being one logical process instance for each such process type. There can also be process types having instances identified by a key, there being one logical process instance for each key value. Different process types are able to have different types on their respective key, and thus have a different number of logical process instances. Process types having instances installed by software in the system itself, can have a database object type developed closely together with this process type. An instance of that database object type corresponds to one logical process instance.

The relation between distribution units and process instances of the kind just described can be expressed as follows for the process types. For a central process type it is trivial, since there is only one logical instance, so there can only be one distribution unit. For a process type the instances of which are identified by a key, the number of distribution units are specified as a constant together with the process specification, and the relation is expressed as a programmed algorithm mapping each key value onto one of the distribution units. For a process type, the instances of which are installed by software in the system itself, process instances are grouped in distribution units as expressed by relations between the database objects.

Processes can furthermore include static process types for which the control system ensures that there is always a physical instance for each logical instance, and dynamic process types for which the control system only ensures that a physical instance is created when some other process wants to communicate with it.

Still further process types are unique process types, for which there is one physical instance for each logical instance, except during certain system activities when two instances cooperate, but act as one process to other processes in a system, and replicated process types, for which there may be several physical instances for one logical instance.

For unique processes, a distribution unit is uniquely allocated to a single processor, and for a replicated process

type, one distribution unit may be replicated on several processors.

Several process types can also be allowed to share the same distribution unit type, provided that such types are of a certain common kind. Such kinds comprise a central, static and unique process type, a keyed, dynamic and unique process type, a central, static and replicated process type, a central, dynamic and replicated process type, and an installed, static and unique process type.

An instance of the keyed, dynamic and unique process type and a database object instance having the same primary key as the process instance are preferably made to refer to the same distribution unit type for at need forcing such a process instance to execute on a same processor holding a primary replica of such a database object with the corresponding key.

Brief Description of the Drawings.

Embodiments of the invention will now be described below with reference to the accompanying drawings on which

Fig. 1 schematically illustrates a telecommunication network,

Fig. 2 is a view schematically illustrating the software structure of a network element, e.g. of the kind shown in Fig. 1.

Fig. 3 is a view schematically illustrating the structure of a control system platform included in the network element according to Fig. 2,

Figs. 4-8 are schematic views of processor pools for illustrating different kinds of process types,

Fig. 9 is a schematic hardware view of a network element,

Fig. 10 is a schematic management view of the network element according to Fig. 9,

Fig. 11 is a schematic resource view of the network element according to Fig. 9,

Fig. 12 schematically illustrates distribution of process instances intended for execution on control processors included in the network element according to Fig. 9,

Figs. 13 and 14 illustrate addressing of a process instance in a control processor from a process instance located in another control processor,

Fig. 13 being a schematic view of three processors with distribution units included therein, and

Fig. 14 being a flow chart illustrating a search course.

Detailed Description of Embodiments.

Fig. 1 schematically illustrates a telecom network according to a basic TMN scenario. The network, indicated at 102, consists of network elements 104 that cooperate to provide network users with network services. The many different services supported by a network element, is to quite a large extent realized by software, which is stored and processed by computer system within the network element. The computer system stores data relevant to network and services, controls services, interacts/communicates with customer premises equipment, indicated by telephone subscriber equipment at 106 and 108, and with other network elements, and controls local transmission and switching hardware. Operations systems for the network are indicated at 110 and 112, as well as an operations system 114 for management of a local network element 116.

Looking at the computer system from the view of physical equipments, the computer system hardware comprises mainly a number of communicationwise interconnected and cooperating processors and mass storage devices, divided between a number of different equipment assemblies.

Any computer controlled system has a control program executing on at least one processor. Today a great number of requirements are imposed on such a system among which can be mentioned that the application development should be easy, the system should have extremely high availability, and it should be easy to upgrade its capacity etc. A way of complying with such requirements is to have it designed with a layered architecture, where applications are programmed on a control system platform, which is supposed to care for most common issues, leaving only the problems of the application domain to

the application programmers.

In such an architecture, application programs, and much of the operating system software as well, consist of code executed in processes, interacting with each other, with the control system, and with the world outside the system through hardware devices. This is generally illustrated in Fig. 2 which is a logical view of the software execution environment in a network element 202. The network element 202 is shown to receive lines 204, e.g. subscriber lines, ending in incoming hardware devices 206, e.g. in the form of line circuits for connection of the subscriber lines. Outgoing hardware devices are indicated at 208. Trunks 210 consisting of a number of transmission channels leave the network element 202 via the hardware devices 208. At 212 a switch is indicated. A software system in the network element 202 is indicated by a block 214 defined by dashed lines. The block 214 is shown to include a control system platform indicated by a block 216, and mutually interacting processes, indicated at 218, 220 and 222. As an example it can be imagined that the process 218 controls one of the line circuits 206, indicated by arrow 224, and the process 222 controls one of the line circuits 208, indicated by arrow 226. The process 220 controls, indicated by arrow 228 pointing to the switch 212, through-connection between these two line circuits.

With reference to Fig. 3, the control system platform 214 can be composed of a number of control processors 302, providing execution and memory resources, and operating system software 304, which manages the resources provided by the processors 302 and implements the abstractions provided by the control system. At 306 processes are indicated as being a common resource.

Henceforth below, when talking about an operating system, this will refer to an operating system of the kind indicated in Fig. 3 and will be referred to as "operation system in question". A process can be of a specific type, forming an instance of its process type. The process type determines what function the process has, processes of different types do

different things, while instances of the same type do the same thing but operate on different data or on different hardware units.

Process instances can be characterized as logical or physical process instances in order to describe different aspect of the process concept. A logical process instance is related to a specific entity outside the processes itself e.g. a "function", a line circuit, a subscriber etc. A physical process instance represents a logical instance, has a state in memory and executes program code.

The term "logical process instance" is an abstract concept. The kind of logical process instances in a system is determined by the process types existing in the system, and their properties, and by the configuration of the system in terms of hardware devices. It is the question of a constant condition, regardless of the system's execution state. A physical instance is a "materialization" of a logical instance, but what physical instances there are in a system depends not only on what logical instances there are but also on the current run-time situation. Over time there can be different physical instances representing the same logical instance, and for process types with certain properties, there may even be several physical instances representing the same logical instance at the same time.

A process type can also be static or dynamic. For static processes the control system ensures that there is always a physical instance for each logical instance. For dynamic processes the control system only ensures that a physical instance is created when some other process wants to communicate with it.

Furthermore, a process type can be "central". For processes of this central type there is exactly one logical process instance for each such process type.

Another type of process has instances identified by a key. For this process type there is one logical process instance for each key value. Different process types can have different types on their respective key, and thus have a different number

of logical process instances. For a kind of process type being keyed and dynamic, there can also be "co-allocation" of data base and process. The concept of "co-allocation" has the following background.

When a database object is updated, the copies in all replicas of some form of database object distribution unit containing the instance must be updated. This means that information must be transferred to several control processors. If the process instance performing the update executes on the same control processor as one of the database distribution unit replicas, the information transfer is however minimized.

If a process instance is of a keyed process type, and it frequently updates a database object instance with the same primary key as the process instance, it can be forced to always execute on the control processor that holds the primary replica by referring to the same distribution unit type from both the process type and the database object type. This co-allocation requires that the types share the same key type, and that the process type is of the keyed, dynamic kind.

Still another process type can have instances related to database objects. For a process type, the instances of which are installed by software in the control system itself, there is thus a database object type developed closely together with the process type. An instance of that database object type then corresponds to one logical process instance.

Finally, a process type can be unique, or have replicated instances. When the type has the "unique" property, there is at most one physical instance for each logical instance, except during certain system activities when two instances cooperate, but act as one process to other processes in the system. When the type has the "replicated" property, there may exist several uncoordinated physical instances for each logical instance at the same time. This property can only be combined with the "central" property. For a static such process type, it implies that one physical instance is created on each of a number of processors. For a dynamic such type, it implies that a new physical instance is created each time the logical instance is

addressed.

When one process wants to invoke a remote operation on another process, it addresses the other process, either by identifying a particular logical process instance or a particular physical process instance.

To be able to identify a logical process instance, it is necessary to know the process type and the difference between different instances, a key value of some type, or a reference to a database object. This is often not a problem as processes of one type are designed to cooperate with processes of a particular other type. It is however also possible to identify an "unknown" logical instance through a generic logical instance identifier, for instance by retrieving such an identifier from the database. This identifier will then have been written from some other process having knowledge of the instance to identify.

For identifying a physical process instance, there exists a data type for generic physical process identifiers. These identifiers, or process instance references, are only valid while the physical instance exists, and can only be used by other processes if the physical instance itself makes the reference available, by storing it in the database, or passing it to other processes through remote operations.

A "central" process instance, or one identified with a key, can be addressed from a process in another network element, which must then supply a network address identifying the network element where the addressed process executes. By default, only processes in the same network element are addressed. Process instances identified by database objects can only be addressed within a network element.

The properties of process type discussed above should be specified in a design language in a process type specification. This specification should thus specify:

- The name of the type.
- Whether it is static or dynamic.
- The type of the primary key.
- Whether the instances may be replicated or not.

- For types with a primary key: a distribution limit or a reference to a distribution unit specification.

- For an installed type; a reference to a distribution unit specification.

Examples of process type specifications specifying three different process types are given below.

```
PROCESS TYPE ResourceHandler IS
  STATIC;
END PROCESS TYPE ResourceHandler;

PROCESS TYPE TrafficControl IS
  DYNAMIC;
  PRIMARY KEY key: SubscriberNumber;
  DISTRIBUTION LIMIT 1000;
END PROCESS TYPE TrafficControl;

PROCESS TYPE LIC_Handler IS
  STATIC;
  PRIMARY KEY key: REFERENCE TO LIC_Data;
  DISTRIBUTED AS LIC_Board;
END PROCESS TYPE LIC_Handler;
```

A physical process instance always executes on one, and only one processor. When an instance is to be created, the control system must decide on what processor the instance is to be created, in such a way that the processors are evenly utilized. The control system must also keep track of the created instances to direct remote operations etc. to the proper process instance. Doing this on a process by process basis would be impossible, there are just too many instances.

The above problem is solved by means of distribution units, which are used to keep a group of process instances together on the same processor. Distribution units group both logical and physical process instances. They group logical instances because the relation between instances and distribution units is expressed in terms of the logical instances, but they also group physical instances because what actually happens when a distribution unit is allocated to a processor is that all physical process instances related to the distribution unit are created on that particular processor.

The way the relation between distribution units and process instances is expressed depends on the properties of the process type.

- For a central process type it is trivial. There is only one logical instance, so there can only be one distribution unit.

- For a process type the instances of which are identified by a key, the number of distribution units must be specified as a constant together with the process specification, and the relation is expressed as a programmed algorithm mapping each key value onto one of the distribution units.

- For a process type where the instances are installed, distribution units are installed as well, and the relationship between process instances and distribution unit instances are set up during installation.

Whether distribution units are replicated or not on several processors, depends on the property of the processes they group. For unique processes, a distribution unit is also uniquely allocated to a single processor, while for a replicated process type, one distribution unit may be replicated on several processors.

An application on the operating system in question consists of a number of cooperating processes that execute code, have a state in memory of the processes and store data persistently in the database. A process instance executes on one control processor only, but it can communicate with other processes regardless of the control processors on which they execute.

To attain functionality, the designer of an application on the operating system in question does not need to think about any distribution aspects other than physical limitations when hardware devices, including device processors, are to be operated upon. Communication between processes is handled transparently by the operating system in question.

The application designer has to consider the hardware distribution in order to attain good characteristics, both when designing process structures and database usage, and when giving directions to the network engineer regarding configuration of the software. Cooperation between functions on control processors in different subnets is of course more

costly and takes longer time than cooperation within the subnet or within a control processor, as information must be passed over several communication channels.

The operating system in question implements the distribution transparency referred to above by knowing on which control processor to find process instances, and, hidden from the application programs, directing inter-process communication and database operations to the appropriate control processors.

As indicated above, if the operating system in question was to actually keep track of where every single process instance was allocated, there would be a lot of overhead. The overhead is lowered through the distribution unit concept described above. The advantage with this approach is that the operating system in question only needs to keep track of the significantly fewer distribution units.

A distribution unit is of a specific distribution unit type. A distribution unit type describes the common properties of the distribution units of that type, namely the types of process instances and co-allocated data base object instances, if any, it groups and a few other characteristics, in most cases the exact number of distribution units of that type that exist in a system.

How the distribution units are distributed over the control processors is stated when the system is configured in terms of allocating a distribution unit type to a distribution pool of control processors. The distribution pool is a theoretically arbitrary grouping of control processors. There may be any number of control processors in a distribution pool and a control processor may be part of any number of pools.

For each distribution unit type there should be a corresponding specification, in which the properties of the distribution unit type should be specified in the same design language as for the process type specification. This specification should specify:

- The name of the type.
- The type of process instances.
- The type of co-allocated database object instances.

- The number of distribution units of the type.

Examples of three different distribution unit types are given below.

The central process type:

```
DISTRIBUTION UNIT TYPE Central_DUT IS
   PROCESS TYPE Central_PT;
END;
```

Process type with key:

```
DISTRIBUTION UNIT TYPE Keyed_DUT IS
   PRIMARY KEY key : KeyType;
   DISTRIBUTION LIMIT 100;
   PROCESS TYPE Keyed_PT;
END;
```

Co-allocation data base and process

```
DISTRIBUTION UNIT TYPE Coallocated_DUT IS
   PRIMARY KEY key : KeyType;
   DISTRIBUTION LIMIT 100;
   PROCESS TYPE Coallocated_PT;
   OBJECT TYPE Coallocated_OT;
END;
```

Specifictions co-existing with the above specification:

```
PROCESS TYPE Central_PT IS
   STATIC;
   DISTRIBUTED AS Central_DUT;
END;

PROCESS TYPE Keyed_PT IS
   DYNAMIC;
   PRIMARY KEY key : KeyType;
   DISTRIBUTED AS Keyed_DUT;
END;

PROCESS TYPE Coallocated_PT IS
   DYNAMIC;
   PRIMARY KEY key : KeyType;
   DISTRIBUTED AS Coallocated_DUT;
END;

OBJECT TYPE Coallocated_OT IS
   PRIMARY KEY key : KeyType;
   DISTRIBUTED AS Coallocated_DUT;
END;
```

There may be several process types that share the same distribution unit type, provided that the types are of the same kind. For each kind of process type the allocation of a dis-

tribution unit type to a distribution pool means slightly different things. This will be described more in detail with reference to Figs. 4-8 for the kinds of process types that have been mentioned earlier.

A first kind, being of the central, static and unique process type, is indicated in Fig. 4. Fig. 4 schematically illustrates three control processors 402, 404 and 406 belonging to a distribution pool. For this type there is only one process instance 408 and therefore only one distribution unit 410. The operating system in question chooses one of the control processors in the pool, viz. the processor 404 in the example shown, to which the distribution unit 410 is allocated and on which the process instance 408 executes. If that processor 404 malfunctions, the operating system in question will choose another one of the control processor, i.e. one of the processors 402 and 406 in the example, from the pool and restart the process on that one.

A second kind, being of the keyed, dynamic and unique process type, is illustrated in Fig. 5. Fig. 5 schematically illustrates three processors 502, 504 and 506, belonging to a distribution pool. The processor 502 is shown to include two distribution units 508 and 510. The processor 504 is shown to include two distribution units 512 and 514. The processor 506 is shown to include two distribution units 516 and 518.

Each one of the distribution units includes three logical process instances, of which some may have been "materialized" to a physical instance, because some other process wanted to communicate with it to reach its services. This has been exemplified in Fig. 5 in the following way. For the distribution unit 508 three "purely" logical instances are indicated at $508_{l1\text{-}l3}$. For the distribution unit 510 two logical instances are indicated at $510_{l1,l2}$ and one physical instance at 510_{ph1}. The distribution unit 512 is indicated to include one logical instance 512_{l1} and two physical instances $512_{ph1,ph2}$. The distribution unit 514 is indicated to include two logical instances $514_{l1,l2}$ and one physical instance 514_{ph1}. The distribution unit 516 is indicated to include two logical

instances $516_{11,12}$ and one physical instance 516_{ph1}. The distribution unit 518 is indicated to include two logical instances $518_{11,12}$ and one physical instance 518_{ph1}.

For this type the application programmer has to provide a distribution limit that fixes the number of distribution units, this distribution limit being 6 in the example, and a function that translates the key that identifies the process instance into a number that identifies a specific distribution unit. The operating system in question spreads, as indicated in Fig. 5, the distribution units evenly over the control processors in the pool, and if one control processor fails, the distribution units on that control processor are moved to the remaining control processors in the pool and moved back when the control processor functions again. As the process instances are dynamic, the operating system in question will only start a physical instance when a process instance is addressed by setting up a dialogue to it. An instance executing on a failing control processor is not automatically restarted.

A third kind, being of the central, static and replicated process type, is indicated in Fig. 6. For such a process type, there is only one distribution unit which the operating system in question will replicate to all control processors in the distribution pool and start one process instance on each control processor. This is indicated in Fig. 6, in which three control processors 602, 604 and 606, belonging to the same pool, each contain a respective replicated distribution unit 608, 610 and 612, the respective process of which being indicated at 614, 616 and 618, respectively. A fourth processor indicated at 620 does not belong to the pool of the other processors and has therefore not received any replicated distribution unit.

Should one of the control processors 602, 604 or 606 fail, the instance executing on it just disappears until the control processor functions again. When a replicated process type is addressed, the sender can only point out the replicated process type as such, not one of the instances specifically. The operating system in question chooses the instance on the

control processor "nearest" to the sender, that is, on the same control processor if there is one, otherwise in the same subnet, otherwise anywhere. If several control processors are equally "near", the operating system in question chooses at random. Note that the internal states of the instances, of the process thus replicated, are independent of each other. The operating system in question will not keep them consistent with each other.

A fourth kind, being the central, dynamic and replicated process type, is illustrated in Fig. 7. Fig. 7 illustrates three control processors 702, 704 and 706, being the only ones of the same pool. The operating system in question replicates a single distribution unit, indicated at 708, 710 and 712, on all control processors in the distribution pool, and starts a new process instance on one of the control processors in the pool whenever such a process is addressed. Process instances 714 and 716 are indicated in the replicated distribution unit 708, and the replicated distribution unit 710 is indicated to include three process instances 718, 720 and 722. The replicated distribution unit 706 includes a single process instance 724.

The control processor that is chosen depends on the location of the sending process in the same way as with the static, replicated process type.

A fifth kind, being of the installed, static and unique process type, is indicated in Fig. 8. Fig. 8 indicates three control processors 802, 804 and 806 belonging to a pool. The control processors 804 and 806 form a group. Allocating the distribution unit type to a distribution pool only means that process instances of the types connected to that distribution unit type may be installed on any of the control processors in the pools. There are, however, no distribution units and no process instances until they have been installed in run-time.

In order to install a distribution unit, a processor group must be selected, such as the one indicated in Fig. 8. A processor group is a group of control processors that is built from configuration data. When a distribution unit, indicated at

808, is installed, the operating system in question chooses one of the control processors, being the processor 804 in the example, in the selected processor group that is also part of the distribution pool to which the distribution unit type is allocated. Within the distribution unit 808 thus installed, process instances, indicated at 810 and 812 as an example, are added, also in run-time. The operating system in question starts the process instances in the distribution unit on the selected control processor, and selects another one and restarts the processes there if the currently selected control processor fails.

Creating the appropriate processor groups is a matter for the application designers and the network engineer, as the operating system in question does not know of the actual physical limitations.

Installed process instances are normally closely related to managed object instances representing hardware resources and when a hardware resource is added to a system, a process instance is usually needed to control or handle that resource. The hardware resource probably needs to be managed in some way, and therefore a managed object instance representing the resource is created. The installation of the process and its distribution unit is therefore often done in conjunction with the creating of a managed object. From relations between the managed object representing the hardware resource and managed objects representing other hardware resources, the proper processor group to select can be obtained.

Referring to that just stated above, and with reference to Figs. 9-12, an examplary installation case will now be described, Fig. 9 illustrating a hardware view of a network element, Fig. 10 illustrating a management view of the network element, Fig. 11 illustrating a resource view of the network element, and Fig. 12 schematically illustrating distribution of process instances intended for execution on control processors included in the network element.

The purpose of a processor group is to make it possible to restrain on which control processors the installed process

instances are to execute in order to cope with physical limitations in the hardware configuration. In the network element in Fig. 9 devices 902 and 904 are shown included in each a block D1 and D2, respectively. The blocks D1 and D2 are intended to represent each a number of devices connected to a switch 906. These devices including the devices 902 and 904 can only be reached from control processors 908 and 910 connected to the same switch 906 as the devices. Another switch 912 has connected thereto two control processors 914 and 916 and two devices 918 and 920 shown as included in each a block D3 and D4, respectively. The blocks D3 and D4 are intended to represent each a number of devices connected to the switch 912, including the devices 918 and 920.

In the corresponding management view of Fig. 10 the processors 908, 910, 914 and 916, the devices 902, 918 and 920, as well as switches 906 and 912 in Fig. 9 have been represented by managed objects given the same reference numerals as a corresponding element in Fig. 10 with an added lowered MO. Analogously, in the corresponding resource view of Fig. 11 the same processors, devices and switches as have been represented by managed objects in Fig. 10, have been represented by resource objects given the same reference numerals as a corresponding element in Fig. 10 with an added lowered RO.

Fig. 10 illustrates how a managed object 1002 representing a distribution pool groups the processor managed objects 908_{MO}, 910_{MO} and 914_{MO}, 916_{MO}. To the distribution pool a distribution unit type is allocated, which will be described more in detail with reference to Fig. 12 further below. The managed objects 908_{MO}, 910_{MO} are connected to the switch managed object 906_{MO}, which has also connected thereto the device managed object 902_{MO}. The managed objects 914_{MO}, 916_{MO} are connected to the switch managed object 912_{MO}, which has also connected thereto the device managed objects 918_{MO} and 920_{MO}.

In the corresponding resource view of Fig. 11 resource objects representing the respective resources of Fig. 11 have been organized in a similar way as the managed objects of Fig. 10. The resource view of Fig. 11, however, also includes a

resource object 922_{RO} representing the control processors 908 and 910 as a group, as well as a resource object 924_{RO} representing the control processors 918 and 920 as a group.

To the distribution pool represented by the managed object 1002 a distribution unit type, schematically indicated at 1202 in Fig. 12, is allocated which, as an example includes distribution units of this type indicated by blocks 1204-1218. The distribution units each include a number of process instances indicated by rectangular shapes in the respective blocks.

Thus, the distribution units 1204 and 1206 each form a group of process instances for handling the devices D1, and installed for execution on the control processor 908. A process instance for handling the device 902 is indicated at 1220. The distribution units 1208 and 1210 each form a group of process instances for handling the devices D2 and installed for execution on the control processor 910. A process instance for handling the device 904 is indicated at 1222.

The distribution units 1212 and 1214 each form a group of process instances for handling the devices D3 and installed for execution on the control processor 914. A process instance for handling the device 918 is indicated at 1224. The distribution units 1216 and 1218 each form a group of process instances for handling the devices D4 and installed for execution on the control processor 916. A process instance for handling the device 920 is indicated at 1226.

A process instance that handles a device connected to any of the switches 906 or 912 must execute on the respective control processors 908, 910 or 918, 920, respectively, connected to that switch, due to the above described groupings of the control processors supported by the resource objects 922_{RO} and 924_{RO}. Thus, if a new device 928 is added to the switch 906, a process instance that handles the new device should analogously be installed, selected to execute on any of the control processors 908 and 910. Therefore, a resource object 928_{RO} representing the new device should be installed, connected to the resource object 922_{RO} as indicated by arrow 930, and the new process instance should be allocated to a dis-

tribution unit forming a group of process instances intended for execution on the control processors 908 and 910. As an example, such a new process instance is e.g. indicated at 1228 in Fig. 12 as installed for execution on the processor 910.

Addressing a process instance in a control processor from a process instance located in another control processor will now be described with reference to Figs. 13 and 14.

Fig. 13 illustrates three control processors CP1, CP2 and CP3. The control processor CP1 is indicated to contain two distribution units A and B. The control processor CP2 is indicated to contain distribution units C, D, R and S. The control processor CP3 is indicated to contain distribution units E, F, R and S. The distribution units R and S are assumed to have process instances of any of the replicated third and fourth kinds allocated to them.

For a process in a control processor to be able to invoke an operation on another process, which may be located in another control processor, addressing information is needed. Such addressing information is provided by tables as follows.

Each control processor has a first table associated therewith, common to all control processors, indicating the distribution of all distribution units on the control processors, and a second table pointing to its own distribution units. For each of the processors CP1, CP2 and CP3 the first table is indicated as T1 and the second table as $T2_{CP1}$, $T2_{CP2}$ and $T2_{CP3}$, respectively. As can be seen, the first table T1 on each line in a first column states distribution units, and in a second column a processor on which the distribution units in the first column are located. Thus, from line 2 of the table T1 it can be seen that the distribution units C and D are located on the control processor CP2. The second table T2 has a first column stating on each line a distribution unit, and a second column providing on the same line a pointer to that distribution unit. This can be easily realized from the arrows pointing from each line of table 2 to a corresponding distribution unit, each arrow being indicated by the same identification letter as the distribution unit to which it

points.

Furthermore, each distribution unit has its own table T3 containing pointers to its process instances. In Fig. 13 this table is illustrated only for the distribution unit D on the processor CP2, indicated as table $T3_D$. As can be seen the distribution unit D contains three process instances P1, P2 and P3. Table 3 contains in a first column three lines stating a respective one of these process instances, and a second column from the respective lines of which an arrow indicates a pointer to the corresponding process instance. In the same way, there are also tables, not shown on the drawing, associated with the distribution units R and S which identify process instances allocated to these distribution units R and S. The tables T1, T2 and T3 form part of the operating system.

Some examplary addressing cases will now be described. It is assumed that a process instance Sender in the distribution unit B on the control processor CP1 in Fig. 13 desires to invoke a remote operation either on one of the process instances existing in the control processor CP2, e.g. the process instance P2, or on a process instance allocated to any of the distribution units R and S in the processors CP2 and CP3. Fig. 14 is a flow chart illustrating a search course performed by the operating system according to which the process instance in question can be found.

A message from the process instance Sender, located to the distribution unit B in Fig. 13, looking for the process instance to be forwarded to appears in the starting point 1402 of the flow chart. The search is started in step 1404 by means of the table T1. If the process instance to be searched for is the process instance P2 located in the distribution unit D, line 2 of table T1 reveals that this distribution unit can be found on the processor CP2. According to this, step 1406 eliminates the case that the process instance could have been located to a distribution unit on the own processor. If the process instance to be searched for is of the replicated type, table T1 in the last line reveals that it can be found on any of the processors CP2 and CP3.

In step 1408 it is determined whether the process instance is replicated or not. If yes, the operating system in step 1410 chooses either the one of the control processors CP1 or CP2 "nearest" to the sending process Sender, or anyone of them if they are equally "near", as has been described earlier for the third and fourth kinds of processes. If no, step 1410 is by-passed.

If the process instance searched for is P2, step 1412 forwards the message to step 1414, in which table $T2_{CP2}$ associated with the processor CP2 in line 2 identifies a pointer to the distribution unit D. Step 1416 forwards the message to step 1418, in which table $T3_D$ in line 2 identifies a pointer to the process instance P2.

In step 1420 it is determined whether the process instance is dynamic or static. If dynamic, step 1422 determines whether the process instance is keyed or replicated. If keyed, the process type is determined to be keyed, dynamic and unique. In step 1424, questioning "started?", it will be determined whether the logical instance is materialized to a physical one or not. If it is not, the process instance is started in step 1426, and the message is forwarded in step 1428. If there is a physical instance, the flow proceeds to the message forwarding step 1428, as indicated by arrow 1430.

If it is found in step 1420 that there is the question of a static process instance, the flow by-passes steps 1422, 1424, 1426 proceeding to step 1428 as indicated by arrow 1432.

If the answer in step 1422 is "replicated", this means that we now have a central, dynamic and replicated type of instance, and that a new physical type has to be materialized. The flow therefore by-passes step 1424, proceeding to step 1426 according to arrow 1434, i.e. the operating system starts the addressed process instance.

Claims.

1. A computer controlled system including

a processing architecture comprising a control system platform composed of a number of control system processors for handling different parts of a database, an operating system which manages resources provided by the processors and implements abstractions provided by the control system, and representations of process resources,

the processing architecture providing for application programs to be executed as cooperating processes interacting with each other, with the control system and with the world outside this architecture,

instances of the process resource representations being distributed over the control system processors by means of distribution units, each distribution unit forming a group of process instances,

the distribution units belonging to specific distribution unit types, of which each distribution unit type describes common properties of distribution units being of this type.

2. A system according to claim 1, in which the process instances include

physical process instances that execute program code, and

logical process instances in the form of an abstract concept, the kind of logical process instances in a system being determined by the process types existing in the system, and their properties, and by the configuration of the system in terms of hardware devices.

3. A system according to claim 2, in which distribution units group both logical and physical process instances, logical instances because the relation between instances and distribution units is expressed in terms of the logical instances, and physical instances because when a distribution unit is allocated to a processor all physical process instances related to the distribution unit are created on that particular processor.

4. A system according to any of claims 1-3, in which the processes include

central process types, there being one logical process instance for each such process type,

process types having instances identified by a key, there being one logical process instance for each key value, different process types being able to have different types on their respective key, and thus have a different number of logical process instances,

process types having instances installed by software in the system itself, there being a database object type developed closely together with this process type, and an instance of that database object type corresponding to one logical process instance.

5. A system according to claim 4, in which the relation between distribution units and process instances is expressed as follows for the process types:

- for a central process type it is trivial, since there is only one logical instance, so there can only be one distribution unit,

- for a process type the instances of which are identified by a key, the number of distribution units are specified as a constant together with the process specification, and the relation is expressed as a programmed algorithm mapping each key value onto one of the distribution units,

- for a process type, the instances of which are installed by software in the system itself, process instances are grouped in distribution units as expressed by relations between the database objects.

6. A system according to any of claims 1-5, in which the processes include

static process types for which the control system ensures that there is always a physical instance for each logical instance,

dynamic process types for which the control system only ensures that a physical instance is created when some other process wants to communicate with it.

7. A system according to any of claims 1-6, in which the processes include

unique process types, for which there is one physical instance for each logical instance, except during certain system activities when two instances cooperate, but act as one process to other processes in a system,

replicated process types, for which there may be several physical instances for one logical instance.

8. A system according to claim 7, in which,

for unique processes, a distribution unit is uniquely allocated to a single processor,

for a replicated process type, one distribution unit may be replicated on several processors.

9. A system according to claims 4 or 5, 6, and 7 or 8, in which several process types are allowed to share the same distribution unit type, provided that such types are of a certain common kind, these kinds comprising:

a central, static and unique process type,

a keyed, dynamic and unique process type,

a central, static and replicated process type,

a central, dynamic and replicated process type,

an installed, static and unique process type.

10. A system according to claim 9, in which an instance of the keyed, dynamic and unique process type and a database object instance having the same primary key as the process instance are made to refer to the same distribution unit type for at need forcing such a process instance to execute on a same processor holding a primary replica of such a database object with the corresponding key.