Contents Chapter Five

CHAPTER FIVE

MULTI-CONTROL IN TELEPHONE EXCHANGES

5.1. INTRODUCTION

Much in line with **Leakey '77** we may distinguish between four different generations of telephone switching systems. The first generation then is the non-automated, operator-controlled system. The second generation is the electro-mechanical system (step-by-step systems and registor-translator systems; early types of distributed automated control). The third generation, which was introduced in 1960[1], is the automatic system with fully centralized control. The fourth generation, introduced in 1971, is the automatic system with multiprocessor control[2]. The third and fourth generation switching systems operate on a series of system programs, stored in memory modules. They are usually briefly indicated as the *stored program controlled* or *s.p.c.* systems.
Clearly, the telephone systems cannot always be sharply distinguished, and the developments have in general been more gradual than one may conclude from this classification. A great variety of systems has been investigated and is or has been in service. As the average life-time of the telephone exchanges is relatively long, it is not expected that the spc-systems will outnumber the electro-mechanical systems within the next 10 years. Still, these systems appear to have great advantages.
Hills & Kano '76, for instance, listed the following points:

(i) "Labour savings as a result of easier administration (for changing subscriber's information, etc.) and considerably reduced maintenance effort;
(ii) higher traffic capacity;
(iii) space savings (i.e. it will be possible to replace an existing exchange by one of higher capacity in the same building);
(iv) power saving;
(v) greater possibility of cost reduction in the future by sharing the fast developing technology of digital computers;
(vi) flexibility to changes that are bound to occur during a life-time of an exchange which is usually in the order of 20 - 40 years;
(vii) economical offering of new services to the subscriber." **(Hills & Kano '76, pg. 1/2).**

Further, **Leakey '77** noted that:

"(viii) the exchanges should be easier to plan, manufacture and install. It should be possible to reduce the procurement

— — — — —

(1) The Whitney-2 exchange of 600 lines, situated at Morris (Illinois, U.S.A.), developed by the Bell Laboratories (**Keister *et al.* 1960**). This system was in service from November 1960 to January 1962 (14 months).
(2) For instance, the Ericsson system AKE-13.

cycle time to something like 18 months for large exchanges, with the actual installation time being reduced to less than two months."

(ix) "Maintenance costs should be reduced by virtue of the powerful fault diagnosis arrangements that can be incorporated." **(Leakey '77, pg. 29).**

This somewhat euphoric view of s.p.c.-systems is shaded by the remarks of **Leakey & Sigrist '72; Leakey & Lawrence '72;** and **Lorétan '75.** We quote Lorétan:

"Leakey shows clearly that the introduction of simple automatic exchanges was in many respects no real progress as many facilities provided by the telephone operator were no longer available. In fact, a manual switchboard of moderate size can offer facilities like transfer, alarm calls, call back when free, call waiting and many more, depending on the skill of the operator. (-) Only with sophisticated central controllers it becomes now possible to restore some of these early facilities even for larger numbers of subscribers."
(Lorétan '75, pg. 3).

Lorétan further remarked that the early s.p.c.-systems, with their highly centralized control intelligence, have some proponent disadvantages, compared to the earlier systems:

(1) "The centralization of the control power may seem very attractive from many points of view. In practice, however, all the advantages cannot be realised. The savings on the peripheral equipment, which is degraded to a completely subordinate slave function, is somewhat compensated by the necessity to provide redundant equipment in the central control in order to guarantee the very high availability expected for a whole exchange."

(2) "The flexibility of the stored program has proved rather expensive in the implementation."

(3) "... the programs required for maintenance and fault location are usually very large ... in the same order of magnitude as the call processing and administration programs."
(Lorétan '75, pg. 4/5).

(4) The commonly used time-sharing techniques for the real-time processing is in Lorétan's view "inherently very vulnerable and relies completely on an operating system."

Lorétan's remarks relate mainly to the large monolithic types of computer control which are characteristic for the first s.p.c.-systems. Indeed, the forcing of all control tasks in one sequential processor yields a highly complex software structure. The observance of the real-time constraints, for instance, requires a division of tasks in priority classes and interrupt based execution cycles. (We return to this in section 5.2, point (1).) Thus, the real-time concerns, which one would like to treat on a low level of abstraction as they are really implementation details, not only *influence* but even *determine* the software structures on the highest levels of the design.
Okada and Kajiware confirmed these points when they noted that:

"... the input information processing function requires a fairly large amount of processing or software steps, caused by detecting arrival of input information within a permissible delay."

"This implies that the total efficiency of the control system is as-

> sumed to deteriorate because of the inefficient processing if all the different kinds of function are included in a single processor." **(Okada & Kajiwara '77, pg. 79).**

Elsewhere **Hills & Lorétan '76,** therefore, concluded that:

> "It is probable that in another decade the early monolithic s.p.c.-system and its direct derivates will be regarded as a dinosaur."

Today, the economical motivations for the concentration of all control intelligence in one large sequential processor have largely disappeared (see chapter 1). The cost of hardware is at present only a fraction of the cost of software development and maintenance. Consequently, there is a tendency to move the real-time considerations back to the lower 'software layers'. The introduction of the 'pre-processors' (also named 'front-end processors', 'peripheral processors', 'i/o processors') in many s.p.c.-systems, and the more recent introduction of microprocessors as logical building blocks, can be placed in this context. Especially, the development of the mini- and microprocessors has given new impetus to the research on alternative types of control.
This is where the multiprocessor systems come into view. It would be a mistake to presume that the introduction of multi-control in telephone systems makes the software structures more complex. On the contrary, it can lead to important improvements in clarity and structuredness. The introduction of multi-control leads naturally to a division of the software into logical units, which are processed concurrently, under the observance of clear coordination rules. Simplicity, modularity, and structuredness are, especially in systems as complicated as telephone exchanges, of prime importance.
For these reasons we study the application of multi-control to telephone systems in this chapter. In the next chapter we will analyze the coordination problems that may occur in a multi-controlled switching system, and we will study systematic ways to solve them. The coordination problems are essentially the same for all types of multi-control discussed here. In the analysis we will therefore abstract from the specific types of multi-control. The coordination problems *can* and should be solved independent of hardware concerns and implementation details.

5.2. TYPES OF MULTI-CONTROL

We discuss the following four types of multiprocessing control in telephone systems:

(1) processor multi-plexing;
(2) functional delegation;
(3) processor duplication, and
(4) processor multiplication.

The first two types of control are not necessarily based on the use of more than one processor.
In practice, one hardly ever uses either of these techniques in isolation, but applies them in combination. Still, it can be clarifying to consider the techniques and the related problems one by one.
The first three techniques are commonly used; the last one has only more recently acquired some attention.

(1) Processor multi-plexing
This is the traditional type of control organization for s.p.c.-systems. One processing unit is shared among a large number of processes, which execute alternately. Each task in the system is assigned a priority on the basis of the timing constraints. Line scanning for digit reception, for instance, may have to be executed once every 10 msec. per line, and as such it obtains a high priority. Other tasks, like path searching and call supervision, are less time critical, and still other tasks, like routine testing and subscriber administration, have hardly any timing demands, which implies a low priority. The rate at which the most urgent tasks have to be executed is controlled by means of real-time interrupts. Less urgent tasks are executed in order of importance in the remainder of the time. If faults are detected a fault tracing and/or system recovery program is started via an interrupt of the highest priority.

The processes in this type of system share the central processor, the network circuitry, and the data. When two processes of different priority levels are allowed to interrupt one another, the interrupting process may mutilate the data accessed by the interrupted process. Theoretically, there is even a danger of deadlocking when the interrupted process has set a mask on which the interrupting process is blocked. In practice, however, these problems can hardly occur as, clearly, one specific call (with the corresponding data fields) can only be in one call-phase at a time, and is in general only accessed by one process at a time. The calls are in a way 'pipe-lined' through the system, and this prevents many potential interaction problems.
The type of software structure which is characteristic for this form of control organization can however make it extremely difficult, if not impossible, to discover the situations in which interaction can lead to faults. There are no systematic methods available to design or analyze coordination schemes in these systems, or at least: they have to our knowledge never been described in the open literature. Only *ad hoc* solutions to occasional problems have been described. Some form of protection can, for instance, be obtained by interrupt masking and interrupt allotting techniques. The system of real-time interrupting is however intended to guarantee the strict observance of timing constraints, and therefore the masking of interrupts is only acceptable for very brief periods of time. Some amount of protection can also be built 'in the program layout'. One can, for instance, see to it that processes which can interrupt each other are always non-interfering. The interfering processes are then included in the same priority level, so that they cannot interrupt one another.
It goes without saying that these solutions are error-prone and hardly resis-

tant to changes in the software for system maintenance, extensions or adaptions. A slight modification of the programs may jeopardize the entire protective system.

(2) **Functional delegation**
To reduce the load of the central processor certain tasks can be delegated to autonomous peripherals, like autonomous scanners and i/o processors. The communication and cooperation between peripheral and central processor requires protection. The usual procedure is, however, to operate the peripherals in a master/slave mode. The peripherals are thus denied a full autonomy. The protections can again be built in the software structures. The central processor may, for instance, initiate the peripheral by a start command, and by supplying it with the addresses of the memory blocks in the central memory, where the peripheral may fetch its input and store its results (on a cycle stealing basis). When it has completed its task the peripheral generates an interrupt on the central processor. In the meantime the central processor will execute non-interfering processes.

(3) **Processor duplication**
Processor duplication is often applied to increase the system's reliability. Instead of 1 single central processor, one uses 2. The second processor can be used in three ways:

(a) Hot Standby. The second processor is kept in reserve to take over control when the first processor fails. The second processor will not execute instructions from the call processing software.

(b) Synchronous Operation. The second processor executes precisely the same instructions as the first (active) processor, but is denied access to the network circuitry and data lines. The results of both processors are compared by an independent circuit (a comparator), and whenever a difference is detected each processor runs a fault-tracing program until it can be determined which of the two processors is failing. The faulty processor is switched off-line and the other processor takes its place. The fault can then be repaired without disturbing the normal call processing. (Observe that only hardware faults within a processor can be detected in this manner.)

(c) Load Sharing. (See also section 5.3.) The traffic load is divided among the two processors. If one of the two fails, the other will take over all traffic. The system capacity is nearly twice as high as in the other two cases. (Not precisely twice as high, as some time will be lost due to the delays in synchronizations, for instance, for the access of shared memories.)

Processor duplication with hot standby is rarely applied. **Thompson '75** mentions only two examples in a list of nearly 70 systems (the No. 2B ESS and No. 3 ESS, intended for respectively, medium and small sized terminal exchanges).
Synchronous operation is more common. It was applied, for instance, in the Philips PRX 205 system, the No. 2 ESS system, and the D-10 and D-20 system of NEC. Load sharing modes of operation have only recently acquired some attention. They were applied in, for instance, the Metaconta 10C (BTM) system.
With each type of processor duplication there are special 'take-over' problems, i.e. problems to ensure the continuity of the call processing in case of processor failure. With hot standby and with load sharing this implies, for instance, that the active processor(s) must continuously update the memory of their colleague.
Take-over problems are, however, not investigated further in this study.

Observe that the coordination problems in a call processing program can only be caused by load sharing (disregarding the special take-over problems in the other systems).

(4) **Processor multiplication**
We discuss four types of control organization for systems which contain more than two processors:

(a) Synchronous Operation. (See (3b).) The synchronous operation of more than two processors can be applied when the system reliability and availability must be extremely high. All processors execute the same instructions. A comparator detects faults. By means of a simple 'majority vote' it can then be decided which processor(s) are faulty and which are not.
In telephone systems this type of control is not or rarely applied.

(b) (N + 1) Load Sharing. N processors share the traffic load. One processor is held on standby. If one of the active processors fails, the standby processor takes its place. Examples of this type of control have been described for the Metaconta 10C system **(Thompson '75)** and the Mark IIB system **(Leakey '77, pg. 24)**.

(c) (2N) Load Sharing. N processors share the traffic load, as in (4b). This time, each processor has been assigned one specific off-line processor with which it operates in Synchronous Mode (as described in (3b)). Examples are the AKE-13 and the AXE-10 system, and the multiprocessor extension of the PRX-205 system.

(d) Full Load Sharing. The traffic load is shared by all processors. There are no 'standby' or 'synchronously operating' processors at all. If one or more processors fail, the remaining processors will take-over its load (graceful degradation of service). This type of organization is rarely or not applied for more than two processors.

Again, the first type of control (Synchronous Operation) cannot yield coordination problems in the call processing software. In the other three cases, the nature of the problems may depend on the specific type of load sharing applied (see section 5.3).
With the last three types of control, the capacity of the telephone exchange can be increased step by step. (Note, however, that more load sharing implies more contention. See section 1.1.2, cf. appendix E.)
The coordination problems in the multi-controlled telephone systems are generally studied and solved on rather low levels of abstraction, that is: directly in terms of hardware facilities and operations on hardware circuits. One often uses hardware exclusion logic. The 'multi-plexor' in the AKE-13 system, for instance, guarantees that only one processor can gain access to a memory module at a time. One processor cannot gain access twice in succession, when other processors are waiting. Arbitration within this multi-plexor is performed by means of static priorities. On a slightly higher level one uses simple locking devices and/or test&set operations.
Occasionally one applies still higher level solutions, for instance, by concentrating the coordination concerns in an operating system **(Ward '72; Janssens '73)**. In the Metaconta 10C system nearly all interference problems are circumvented by a specific type of load sharing, and by an extensive use of software inhibition masks.
In the next section we will consider several ways to organize a load sharing.

5.3. LOAD SHARING STRATEGIES

We discuss four types of load sharing:

(1) **Functional division**
Every active processor executes a specific part of the control tasks in the call processing. During the life-time of a call, it will be served by several processors consecutively. The processors must be able to exchange information about the calls processed.
With this type of load sharing one can apply many different types of control-organization (e.g. hierarchical control, or pipe-lining). Hierarchical control with mini- and microprocessors was, for instance, applied in the French E-1 system. (See **Hills & Lorétan '76**; **Hills & Kano '76.**) In another form it was also applied in the AXE-10 system, where a *functional* division of tasks is combined with a *sectional* division on the lower hierarchical levels.

(2) **Sectional division**
Every active processor executes all control tasks for part of the network circuitry. If a call is switched through different parts of the network, it will be served each time by a different processor. The processors must again be able to exchange information about the calls processed.

(3) **Call division**
Every active processor executes all tasks for one specific part of the calls The calls can be assigned to the processors in three ways: at random, time-based or sectional.

(a) At Random. Every active processor handles as many calls as possible from any part of the network, in any time interval.

(b) Time-Based. The processors accept only new calls in specific time-intervals, with the aid of a real-time clock. Each active processor is assigned another time-interval.

(c) Sectional. An active processor accepts only new calls from a specific part of the incoming lines. The difference with method (2) is that an accepted call is served completely, and not only in parts of the switching network.

(4) **Full random division**
A theoretical fourth type of load sharing is to have each active processor select tasks from shared job queues at random. The processors are then not specifically assigned to calls, nor to sections or functions. This is a sort of 'worst case' in view of process coordination, as the interaction chances are maximal. (We study this case in more detail in chapter 6.) From another point of view, however, this type of load sharing is the 'most elegant' of the four methods discussed here, as all processors in the system are completely equivalent.
The full random load division strategy is clearly more flexible than the others discussed here. It leads almost naturally to optimal *load balancing*, which may increase the effective capacity of a computer system considerably. It also makes the *extension* of the system's capacity more straightforward. (Such extensions may present problems especially when a functional division is applied.)

5.4. COORDINATION TOOLS

In chapter 1, section 1.3.2.1, we have studied the criteria for coordination tools in general purpose multiprocessing systems: as programming tools they should be simple, flexible, and verifiable; in their implementation they should be efficient and robust. (For a precise interpretation of these terms we refer to chapter 1.) In this section we ask ourselves the question of what priorities we should assign to each of these requirements, in the 'special purpose' telephone system with multi-control. The priority of simplicity and verifiability has been argued repeatedly in this thesis, and will need no further emphasis. We will discuss flexibility, efficiency, and some robustness aspects.

(1) **Flexibility**
Flexibility is clearly no strong requirement in the special purpose telephone system. The system's major structure and goal are given, and they are not likely to change during the life-time of an exchange (at least not in so far as the coordination aspects are concerned). The tools should be adequate for the system's needs, and the needs are known a priori. These remarks do, however, not apply to documentation purposes. For documentation purposes the highest level tools may be used to clarify the working of the system (notably the section model).

(2) **Efficiency**
Efficiency is worth considering in any real-time system. In most computer-controlled telephone systems the call handling capacity is only 20% - 30% of the total system capacity. The larger part is used for maintenance and operating system software. Any improvement on the efficiency of the 70% - 80% overhead has a relatively large impact on the system's call handling capacity. But then, the efficiency of the system is only partly influenced by the type of coordination primitives used. Of a relatively much greater impact are the software structure, the type of load sharing, the memory organization[1], and the data structures. For these reasons we may conclude that efficiency can also in these systems be subordinate to simplicity and verifiability.

(3) **Robustness**
The observance of the integrity, determinance, and of the no-deadlock/no-starvation requirements may again be presupposed. Symmetry and fairness are, however, only relevant under very unfavorable traffic conditions (overload). The inclusion of elaborate scheduling facilities would be a luxury. The relevant scheduling techniques can be formulated a priori and fixed. In fact, many critical operations in the telephone system concern the setting, testing and releasing of protection bits (FREE/BUSY). These primitive actions require little execution time, so an implementation with busy waiting will be quite acceptable. If the circuit required is not unique (e.g. a tone-receiver) the proper reaction would not be to execute a busy-wait, but rather to withdraw the request for occupation and to test the next circuit in line.

- - - - -

(1) The memory organization should allow for maximal concurrency and minimal contention. There should then be a large amount of independently accessible memory modules, which thus secure a 'fine grain' of implicit exclusions.

5.5. CONCLUSION

We have divided the causes of coordination problems in multi-controlled s.p.c.-telephone systems into three main sections:

- functional delegation;
- processor multi-plexing, and
- processor duplication and multiplication with load sharing.

A structured approach to the coordination problems does not seem to be available. In practice one applies ad hoc solutions.
The coordination problems caused by functional delegation are solved, or counteracted by restricting the autonomy of the 'autonomous' peripherals to a slave-function.
Coordination problems caused by processor multi-plexing are counteracted with interrupt masking techniques, restricting the size of critical sections, and avoiding interference between processes from differing priority levels.
Coordination problems caused by processor duplication or muliplication with load sharing are counteracted with low level exclusion tools, and hardware exclusion circuitry. One rarely finds higher level solutions. There is an evident lack of systematic analysis and design tools for coordination schemes. The coordination problems are hardly ever treated in a uniform way. Exclusion problems, ordering problems, scheduling problems, and communication problems are all treated differently in non-standard ways. The different aspects are not recognized as parts of the same overall problem, which should require a simple, uniform and structured approach.

5.6. REFERENCES

Hills, M.T. & Kano, S. (1976), *Programming electronic switching systems*, IEE Telecommunications Series 3, J.E. Flood (ed.), P. Peregrinus Ltd., Stevenage, Herts., U.K., 1976, 207 pgs.

Hills, M.T. & Lorétan, R.P. (1976), *The future direction of s.p.c. systems*, Proc. Int. Switching Symp., 1976, Kyoto, Japan, paper 412-2, 7 pgs.

Janssens, J. (1973), *Metaconta 10C toll exchange: general description and software design*, Electrical Communication, Vol. 48, No. 3, 1973, pp. 239-247.

Keister, W., Ketchledge, R.W. & Lovell, C.A. (1960), *The Morris electronic telephone exchange* Proc. IEE, paper No. 3382 E, Nov. 1960, Vol. 107B, Suppl. No. 20, pp. 257-263.

Leakey, D.M. (1977), *Computer-controlled digital telephone switching systems*, GEC Journal of Science & Technology, Vol. 44, No. 1, 1977, pp. 22-30.

Leakey, D.M. & Sigrist, P. (1972), *The role of stored-program control in telephone switching systems*, GEC Telecommunications, No. 39, 1972, pp. 4-5.

Leakey, D.M. & Lawrence, G.N. (1972), *Stored-program control in telephone switching systems*, GEC Journal of Science & Technology, Vol. 39, 1972, pp. 131-136.

Lorétan, R.P. & Hills, M.T. (1975), *Control of telephone exchanges using multiprocessor computer systems*, Science Research Council Application, final report, Univ. of Essex, Dept. of Electr. Eng. Science, Telecomm. Systems Group, Report No. 134, 1975, 189 pgs.

Okada, K. & Kajiwara, M. (1977), *Functionally distributed telephone exchange control system using microprocessors*, Euromicro Newsletter, Oct. 1977, Vol. 3, No. 4, pp. 79-82.

Thompson, M.F. (1975), *Worldwide electronic switching systems overview*, Telecomm. Techn. Center, Stamford, Conn., ITT, Aug. 1975, 9 pgs.

Ward, M. (1972), *Total control of telephone exchanges by s.p.c.*, GEC Journal of Science & Technology, Vol. 39, No. 4, 1972, pp. 181-184.

Selected Bibliography of System Descriptions

1. AKE-13 system: Ericsson Review, Vol. 50, 1973, No. 2, pp. 34-57.
Ericsson Review, Vol. 50, 1973, No. 2, pp. 58-64.
Ericsson Review, Vol. 51, 1974, No. 2, pp. 34-47.
Het PTT Bedrijf, Vol. 18, 1973, No. 4, pp. 209-294.
IEE 2nd Int. Conf. on Softw. Eng. for Telecomm. Switching Systems, Salzburg, 1976, IEE Conf. Publ., paper No. 135, pp. 92-95.

AXE-10 system: Ericsson Review, Vol. 53, 1976, No. 2
Ericsson Review, Vol. 54, 1977, No. 1, pp. 2-6.

2. D-10 and D-20: Japan Telecomm. Review, Vol. 13, July 1971, pp. 107-115.
Japan Telecomm. Review, Vol. 13, Oct. 1971, pp. 161-170.
Japan Telecomm. Review, Vol. 14, Febr. 1972, pp. 78-86.
Review Electrical Comm. Lab., Vol. 22, 1974, pp. 761-764.
NEC, Research & Development, No. 43, Oct. 1976, pp. 91-99.

3. No. 1 ESS : Bell System Techn. Journal, Vol. XLIII, No. 5, 1964.
No. 2 ESS : Bell System Techn. Journal, Vol. XLVIII, No. 8, 1969.

4. Mark IIB : GEC Journal of Science & Techn., Vol. 39, No. 4, 1972, pp. 181-184.

5. Metaconta L : Electrical Comm., Vol. 47, No. 3, 1972, pp. 159-163.
Metaconta 10C: IEEE Trans. COM, USA, Vol. 17, No. 3, 1969, pp. 333-339.
Electrical Comm., Vol. 48, No. 3, 1973, pp. 239-247.

6. PRX-205 : Philips Telecomm. Review, Vol. 31, No. 2, Sept. 1973.
Philips Telecomm. Review, Vol. 34, No. 3, Oct. 1976.
Proc. Intern. Switching Symp., 1976, Kyoto, Japan, paper 422-4.