# HIGH LAYER PROTOCOL STANDARDISATION FOR DISTRIBUTED PROCESSING

**V. Decreuse**

**Rapporteurs:** Mr. B.C. Hamshere
Miss M. West

## A. Introduction

- The present lecture is aimed at formulating some thoughts related to the upper layer protocols in the following two respects:

  - A good knowledge of the applications, or end user requirements, is key in achieving an efficient system architecture.

  - To which extent one can hope to define standardized upper layer protocols taking into account that they would have to cope with existing applications.

- A network is a part of a broader entity which could be referred to as a "System" encompassing both the connectivity and interworking capabilities. It seems that the education in computing and data processing science has emphasized for the last years the study of what is related to the connectivity functional set (routing algorithms, queuing theory applied to the transmission, data flow mechanisms... etc) putting aside what is likely becoming the most important part of the system i.e. the interworking capabilities.

  As a matter of fact, the connectivity facilities are to be designed to work on behalf of the end users (terminals end programs) which are the source and the sink of the conveyed information. As a consequence, it is of a tremendous interest to understand in depth what the end user requirements are in terms of communication facilities as they lead to make some critical design choices at the underlying networking functional level.

- The relationships between end users evolved during the last years in such a way that the usual operator-terminal-program dialogue, even if remaining the main part of the network traffic, has been replaced in part by more sophisticated exchanges supporting program to program communications. What is important to point out here is that a terminal operator, as a human being, is capable of handling from his own complex operations whereas a program does not have the same flexibility.

  - As a first consequence, the advent of program to program communication needs enforces the system designers to invent more and more sophisticated upper layer protocols permitting two coupled processes to speak with each other in a consistent way.

  - As a second consequence, the advent of the so called "Distributed Processing" permitting several processes to cooperate in order to achieve an atomic unit of work, involving complex Data Base structures and other resources, has introduced the requirement of powerful and efficient synchronization/resynchronization protocols which are likely today the keystone of the system software design and development.

- Furthermore, the recent requirements addressing the world of "Open System Interconnection" have emphasized the need of additional protocols so as to allow the coupling in a more flexible way of processes belonging to different systems from a manufacturer point of view. Such cooperating processes have to dynamically understand common semantic sets conveyed by a common agreed upon transfer syntax.

  An important question arises now. Is an open system architecture (i.e. applying to heterogeneous systems) quite achievable with respect to the various application requirements, their various system implementations unless defining itself all the processing environment (i.e. not only what is related to the communication part of it) what is obviously beyond its scope ? To answer such a question is certainly a matter of debate in so far as existing applications (implementations), involving system and user coding, are to be changed so as to comply with them.

  According to their own objectives, on the one hand the OSI architecture has been defined in a down-top approach while the SNA (System Network Architecture) on the other hand has been mainly designed by relying on the application needs. The ECMA's DIPE (Distributed Interactive Processing Environment), though claiming for its compliance with the OSI architecture, steadily states that the purpose of processing is processing itself not communication. The communication protocols are included because the processing is distributed but they are not fundamental to processing itself. In this respect the DIPE structure, which is user oriented should be designed by starting from the corresponding top-down viewpoint specifically that of the designers of distributed applications not that of the communication engineer.

  It would not be very advisable to conclude as far what is the right approach.

- Some additional thoughts about system architecture.

In many respects, any system architecture can be thought of as an axiomatic (for instance the Euclidean geometry) which is in essence more concerned with the relationships between the objects it defines than with the properties of the objects themselves. But unlike an axiomatic, a system architecture has to cope with the real world what means in the data processing environment that the structural concepts have to take into account the end user requirements.

In this critical examination of the KANT's "Critique of the pure Reason" BERTRAND RUSSEL denies the fact that "synthetic judgments a priori" are possible. On the contrary, according to him, any empirical proposition relies on some experience and depends on observational data. In other words, any "synthetic judgment" is necessarily "a posteriori".

## B. An overview of distributed processing environment

## B.1 : Some preliminary definitions

- Distributed Processing is the cooperative execution of computer programs which are dispersed into separate computers. The prime concern of distributed processing is processing not communication.

- Distributed Data Processing is a Data Processing in which some or all of the processing, storage and control functions, in addition to input/output functions are situated in different places and connected by transmission facilities.

- Interaction Processing is, in its broadest sense, a processing which depends on the transfer information between two or more participants by means of "synchronous" conversations.

## B.2 : Some structural concepts

Distributed Processing involves coordinated relationships between two or more processes aimed at achieving a given piece of work.

B.2.1 : The basic entity is the relationship between two cooperating transaction programs which is referred to as a "conversation" in the SNA world or an "association" in the OSI world. The "association" and "conversation" main attributes are:

- The common agreed upon dialogue rules between the two cooperating entities

- The commonly understood application semantic and the underlying transfer syntax.

- The common agreed upon synchronization rules in order to maintain the involved processing resources in a coherent state.

All those attributes are to some extent negotiated between the two partners at conversation/association set up time.

A set of related conversations/associations aimed at achieving, in a consistent way a given piece of work constitutes a "Processing Tree".

### B.2.2 : The underlying entities

The associations/conversations rely on underlying entities providing them with the required services (see figure 1).

What is of interest here to point out is as follows:

- Association/conversation is aimed at allowing two coupled transaction program instances pertaining to the same or different node(s) to speak with each other.

- Sessions/Session connections are aimed at allowing two different coupled nodes to communicate with each other.

- The SNA session is serially reusable whereas the OSI session connection is not what is a strong drawback from a performance and usability standpoint.

- The OSI architecture, as an heterogeneous system oriented architecture has to comply with flexibility requirements in the program to program communication area. It therefore emphasizes at association and presentation connection level the role of protocols permitting two transaction programs to handshake at application semantic level as well as at transfer syntax level.

- SNA, as a private system architecture, has not to face such needs. The transaction programs are designed to work together even if some common functional capabilities are negotiable at conversation and session set up time.

- The OSI transport connection provides the session with an end-to-end node reliable transmission pipe (what is termed here "Non disruptive route switching") relieving the upper layers from any concern with failures occurring at transport and network level. On the contrary, the SNA virtual route failures are, in the state of the art, propagated to the upper layers which are responsible for managing the recovery.

  It appears here that the OSI down-top design approach led to split up the communication system facilities into two parts which are one the one hand all what is related to pure transport functional set, on the other hand what is close to the application functional set.

  The Top-Down SNA design approach led to emphasize the role of what is related to the application functional set while interesting, from an historical perspective, step by step to what is related to pure networking functional set.

## B.3 Distributed Processing in the perspective of the OSI reference model

As got from figures 2, 3, 4, 5 the OSI architecture clearly states which part of a process belongs to its sphere of interest (OSI environment) and which part it is not concerned with (Local System Environment). Such a cut is quite understandable in the perspective of an heterogeneous system architecture whose main objective is to permit end-to-end system reliable communication (theoretical point of view).

In fact, one can ask whether the pure information processing (L.S.E) could not have any impact on the underlying layers. Performance and usability requirements, claiming for such capabilities as serially

reusable session connections, transmission priority, symmetric responsibility for setting up a new session connection on an available transport connection undoubtedly advocate it would be desirable to go a little beyond the border.
Moreover we can guess from our knowledge of some implemented applications that the local system environment resource protection mechanisms could have a strong impact on the architected synchronization/resynchronization mechanisms. But, what is meant really by local system environment ?

## B.4 Asynchronous and synchronous distributed processing

It is useful so far to distinguish the asynchronous distributed processing from the synchronous one as their requirements are, from a system point of view, not the same.

The asynchronous distributed processing is required by many applications and system services including office system, network management, file transfer and job networking. It is characterized by the fact that it relies on a delayed delivery transmission mode permitting in most cases the requestor to retry at will its work in so far as there are no resource concurrency control requirements. There is an interesting analogy to the postal system.

The synchronous distributed processing is characterized by the fact that the communicating partners converse in real time, responses if any being synchronized with requests. The different processes compete in sharing processing resources what implies that complex resource concurrency control mechanisms are required. An interesting analogy here, is to the telephone system.

Figure 6, though the comparison is not exhaustive, shows what are the basic requirements of asynchronous and synchronous distributed processing. As the most stringent requirements, from a system standpoint, come from the synchronous distributed processing we will focus on what is referred to as the distributed interactive processing environment.

## B.5 Distributed Interactive Processing Environment: An overview

### B.5.1 : Conversation and processing tree

- The conversation is a relationship established between two cooperating transaction programs which can be homed in the same or different system(s) in order to achieve a piece of work. A user TP can schedule the execution of its partner TP either explicitly or implicitly by using appropriate protocols. The target system will schedule the required TP which is identified by a transaction program name (TPN) or transaction code according to its own, locally defined, triggering mechanisms. A basic concept is that the establishment of the dialogue invokes a fresh instance of the destination TP, that is to say whatever the properties of the destination program are (either reentrant or serially reusable or not reusable at all), it will

run under control of a devoted task what insures the uniqueness of the couple driving task -running program. It is obvious that many tasks can, at any given point in time, run the same program each instance of it owning a unique running environment.

- According to the processing requirements, the location of the involved resources, the destination TP can in turn schedule one or many destination TP(s) what leads, step by step, to build the so called "processing tree". The processing tree consists of a root (he initiator TP) coupled to its direct subordinate(s) by conversations which are the "branches" or "arcs" of the tree. Each first level subordinate can in turn have its own subordinates... etc. (see figure 7). The "fresh instance" concept precludes a tree masked structure.

B.5.2 : Intercommunication types of facility

- Function request shipping: this facility enables a TP to access a processing resource (Data Base, queue, file... etc) owned by an other system in an implicit way. By implicit way is meant here that the TPs that access the remote resources are designed and coded as if the resources were owned by the system in which the transaction is to run. During execution, system functions are responsible for shipping the request to the appropriate target system (see figure 8).

- Asynchronous processing: this facility enables a TP to initiate a TP in a remote system and to pass data to it. The data can include the name of a local TP that is to be initiated by the remote system to receive the reply if any. The requesting TP and the destination TP are running independent from each other and no direct correlation between requests and replies is possible (see figure 9).

- Transaction routing: this facility enables a terminal that is owned by one system to run a TP in another system (see figure 10)

- Distributed Transaction Processing: this facility enables a TP to communicate with a TP running in another system. The TPs are designed and coded explicitly to communicate with each other and thereby to utilize the intersystem link with maximum efficiency. The communication is in this case synchronous in that requests and replies can be directly correlated (see figure 11).

Unique requirements from distributed interactive processing

The distributed interactive processing emphasizes in a tremendous way the needs related to response time and consistency of the resources involved in a processing tree.

Response time :

Response time, as previously seen, depends in a part on the capabilities of the lower layers. It depends too on the efficiency of dialogue management and the related features aimed at optimizing the transfer of informations. An important contributor to the efficiency of the dialogue man agement, in a two way alternate (TWA) transmission mode which is usually the best suited to the DIPE, is the design of information transmission features permitting the partners to work in "deferred output mode".

The involved systems have to implement deferred output processing of SEND commands so as to optimize the use of intersystem coupling conversation/session what means that, in general, a message is not sent to the remote system until the next TP command that refers to the conversation is executed. Deferred output enables the system to add dialogue control indicators to waiting data before it is transmitted thereby decreasing the number of useless transmissions on the session. The addition of dialogue control indicators to deferred output is referred to as "piggy-backing". Further optimization can be achieved by accumulating as much data as possible in an internal SEND buffer before actually then across the link. Then the data from a series of SEND commands are transmitted only when the buffer becomes full or when the transmission must be forced according to a TP command.

The strong efficiency of the dialogue protocols based on the utilization of deferred output mechanisms relies on the fact that most services are to be not confirmed as any TP requesting a confirmed service is put in WAIT state till the corresponding response (as opposed to reply) has been received. The TP is unable to send other requests and therefore to take advantage of the "piggy backing" optimization. In this context the availability of dialogue management providing a synchronization facility using exception response mode is very useful as the requesting program can send requests in a continuous way while being aware of any abnormal condition when receiving an exception response.

Deferred output mechanisms impact the TP logic to the extent that exceptional conditions are reported in an asynchronous way. In other words, an event resulting form some request is not directly correlated to this request as it will not be taken into account prior other requests, permitting to schedule the transmission to the partner ,have been issued.

## C.2 : Resource consistency

The consistency of resources involved in a processing tree is the main concern of the system design in the state of the art. Prior looking at the complex mechanisms designed to meet those requirements, it is useful to introduce the basic concepts on which they rely, namely the concurrency control of recoverable resources, the logical or atomic unit of work (atomic action in the OSI world ) and the unit of work.

### C.2.1. Concurrency control

- Problem statement (see figure 12)
  Concurrency control mechanisms are required in an interactive processing environment to the extent that many TPs, even in a single system, compete with each other in order to utilize shared processing resources. Looking at figure 12, we can infer that concurrency control mechanisms provided by basic system facilities (like exclusive control at access method level) are not capable of warranting the consistency of changed resources. As of this figure, it appears that the backout process resulting from the T1 transaction abnormal end will put record n of file A in its state prior T1 was started thereby discarding the change made by the T2 transaction.

  As a result, there are:

  = A lost of information
  = An inconsistency exposure at T2 level as T2 could have changed other resources during its processing
  = A pollution exposure to the extent that record n information, as resulting from the T1 change, could be propagated by T2 in changing other resources.

- Proposed solutions:

  Systems have to design more sophisticated (powerful) mechanisms, involving resource protection managers so as to face the above stated problem.

  = First level: any changed resource by a TP is made available to other TPs in GET mode only. It will not be available in UPDATE mode till the owning TP ends a "Logical Unit of Work" either explicitly (synchronization point scheduled by the TP itself) or implicitly (point of synchronization scheduled by some resource protection manager in some circumstances or by the overall resource protection manager at the end of the TP which is also the end of a Logical Unit of Work). The Logical Unit of Work (LUW) or Atomic Unit of Work can be viewed as a logical sequence of operations which cannot be broken from a logical point of view end of which being either normal (changes are COMMITTED and the relevant resources are unlocked) or abnormal (the consistency requirement implies that a BACKOUT process is to be scheduled leading to put the changed resources in their initial state and to free the relevant locks). Such a mechanism in fact in not quite reliable for it permits a TP other than the "owning" one to access, before the end of the LUW, information which has been changed, then to propagate this information by making some changes based on it thereby not removing the pollution exposure in case of an abnormal end of the owning transaction.

  = Second level: any changed resource by a TP is not made available to other TPs in either (GET or UPDATE mode), till the owning TP ends its logical unit of work. The owning TP has the exclusivity of the changed recoverable resources as long as it

has not reached the end of the current logical unit of work. ("Program Isolation" in the IBM IMS DB/DC subsystem terminology").

The efficiency of such a mechanism can be viewed in different aspects :

= From a resource integrity point of view it looks like an "iron" mechanism.

= From a performance point of view it is obvious that the "program isolation" feature tends to increase the logical contention level which depends on the instantaneous transaction rate, the granularity of shared resources and the LUW lifetime.

The concepts of LUW and program isolation have been summarized on figure 13.

C.2.2 : Resource consistency in DIPE - Overview of synchronization and resynchronization processes (syncpoint level).

From the knowledge of the basic principles on which rely the data integrity mechanisms we can infer that, even in a single system environment, the actions of the different resource protection managers are to be coordinated by an overall local syncpoint management entity which is responsible for ruling the whole process and has been referred to as a "Syncpoint Manager".

The functions of the local Syncpoint Manager are to be extended in a distributed environment so as to insure the consistency of all resources involved in a processing tree. According to figure 14 we can see that, besides the relationships between a syncpoint manager and its subordinate local resource protection manager, there are relationships between all the syncpoint managers belonging to nodes participating to the execution of a "Unit of Work" (U.O.W) in order to coordinate the so called "Conversation Resources".

The relationships between syncpoint managers belonging to a processing tree are ruled by architected protocols conveyed by particular message entities flowing across the so called "Synchronization Tree".

Like the processing tree, the synchronization tree is made up of a root, which is responsible for the overall management of the synchronization tree, subordinates, branches and leaves organized in a pure hierarchical way.

The root is homed by the node in which a TP issues a primitive triggering the synchronization process. Usually the root of the synchronization tree is the same as the root of the processing tree and the structure of the trees are therefore identical.

Nevertheless, it can sometimes happen that the root of the synchronization tree is a subordinate in the processing tree. It is of interest here to note that the opportunity to build synchronization tree structures different from the underlying processing tree structure is to be carefully assessed by the application designers to the extent that the synchronization involved mechanisms do not warrant, in the state of the art, it will properly work in all the cases (collision exposures... etc). (See figure 15).

- Synchronization protocols: overview (Syncpoint level). The synchronization process relies on the basic concept of "two phase commitment". The Unit of Work consists at any given point in time of a vector of inflight Local Unit of Work (LUWAn, LUWBn, LUWCn...) each of them running in a system participating to the overall process.

  At a given point in time, a TP which is usually the root of processing tree issues a primitive aimed at scheduling the synchronization process. The involved syncpoint managers are responsible for generating the appropriate protocols, propagating the requests to their adjacent partners and providing the TP(s) they are coping with the relevant indications.

  = The PREPARE protocol element, is forwarded by the syncpoint manager of the initiator TP, which has issued the SYNCPT request, to all its direct subordinates (cascading). This command asks the subordinate to place its protected resources in a state that allows them to be fully committed to the change that have been accumulated during this LUW but that also allows the changes to be backed out. The choice to commit or backout the inflight LUWs is made by the initiator after interaction with all agents. The syncpoint managers upon receipt of the PREPARE command have to propagate it to their subordinates, if any and to pass the indication "syncpoint requested" to the TP they are coping with. The TP can either issue a SYNCPT request or a BACKOUT request if it has to face any unrecoverable processing error.

  = The SYNCPT request will be translated by the syncpoint manager in a REQUEST-COMMIT protocol element which says that the issuer has succeeded in preparing all its protected resources.

  = The BACKOUT request will be translated by the syncpoint manager in a protocol element BACKED-OUT which says that the issuer has backed out its current LUW and will be propagated to the overall syncpoint manager which in turn will ask all the participants to the U.O.W to backout their current LUW.

  = The COMMITTED protocol element is propagated to all the participants after the syncpoint manager linked to the initiator

TP has received a REQUEST-COMMIT protocol element from all the participants. It asks them to really commit.

= The FORGET protocol element informs the syncpoint manager that sent COMMITTED that its log records for this LUW can be erased. It tells also the initiator's syncpoint manager that the syncpoint is complete and that control can be returned to the initiator TP (see figure 16).

- Resynchronization protocols (see figure 17)

The consequences of the "in doubt period" phenomenon are as follows:

= In case of failure during the "in doubt period" each conversation partner does not know the actual status of the other one. As the TP cannot indefinitely hold locks for protected resources, it has to take a decision and must either commit or backout.

= The system is required to provide an architected resynch. process allowing two resynch. system defined TPs to set up a conversation (by using any available session on the ISC link) in order to compare the status of the two partners and send a report to the involved operators.

The resynchronization process is based at least on:

= The logging of the appropriate information about the U.O.W state changes by each syncpoint manager.

= A mechanism enabling the two system resynch. TPs to compare the status of the partners in order to identify potential mismatches leading to an inconsistency of the U.O.W.

= The participation of the system operators who after receiving the reports from the system resynch. TPs may have to schedule some installation designed "reconciliation process". The "reconciliation process" will be likely a run of programs using some kind of user logging and permitting either to inhibit local changes or, on the contrary, to put the involved resources in their state prior backout process was triggered.

Further some systems can provide some more powerful mechanisms permitting the two partners to take over the broken conversation (reconnect support).

## D. Conclusions

Remembering now our initial objectives, we can answer in part the questions as stated by the introduction to this presentation.

D.1 : It is obvious that the knowledge of the behaviour and requirements of applications is key in designing a System Architecture. As pointed out by the above presentation, the application needs impact the functional capabilities provided by upper layers as well as lower layers. The session or session connection properties, the information transfer defined rules... etc, are to be designed to meet the application requirements. Moreover, the most complex part of the system design which is related to the synchronization/resynchronization mechanisms relies on concepts which in fact are defined by the application themselves rather than by a system network architecture. We can, for example, say that the concepts on which is based the architecture of the IBM so called "Logical Unit type 6.2" have been got in a large part from the CICS (Customer Information Control System) implementation which is today the most pervasive teleprocessing control system in the IBM world.

Such concepts were in fact defined prior the advent of networking and are aimed at meeting the pure processing requirements while putting aside the communication part.

The advent of the distributed interactive processing led later the designers to define architected protocols permitting the participants to a processing tree to synchronize with each other while still using the local defined basic mechanisms and entities like syncpointing, LUW...etc.

It is therefore of interest to note that the upper layer system protocols were in this case defined according to pure local implemented mechanism. We pointed out also that, though the SNA synchronization protocols theoretically permit the synchronization tree structures tobe different from the processing tree one, there are many restrictions which enforce the application designer to be careful and use private protocols in order to take advantage of such a capability.

As a consequence, it appears that more powerful protocols are to be invented what is obviously a large scope of investigation and study.

D.2 : The other basic question is related to the capability to define standardized upper layer protocols.

Such an achievement is undoubtedly very desirable but one can ask whether it is workable. We have primarily to distinguish asynchronous distributed processing from the synchronous one.

- One can think that as asynchronous distributed processing uses simple facilities from a dialogue and synchronization point of view, it would be possible to standardize upper layer protocols.

But it is still a matter of debate and we have to go more in depth in the knowledge of the involved application requirements.

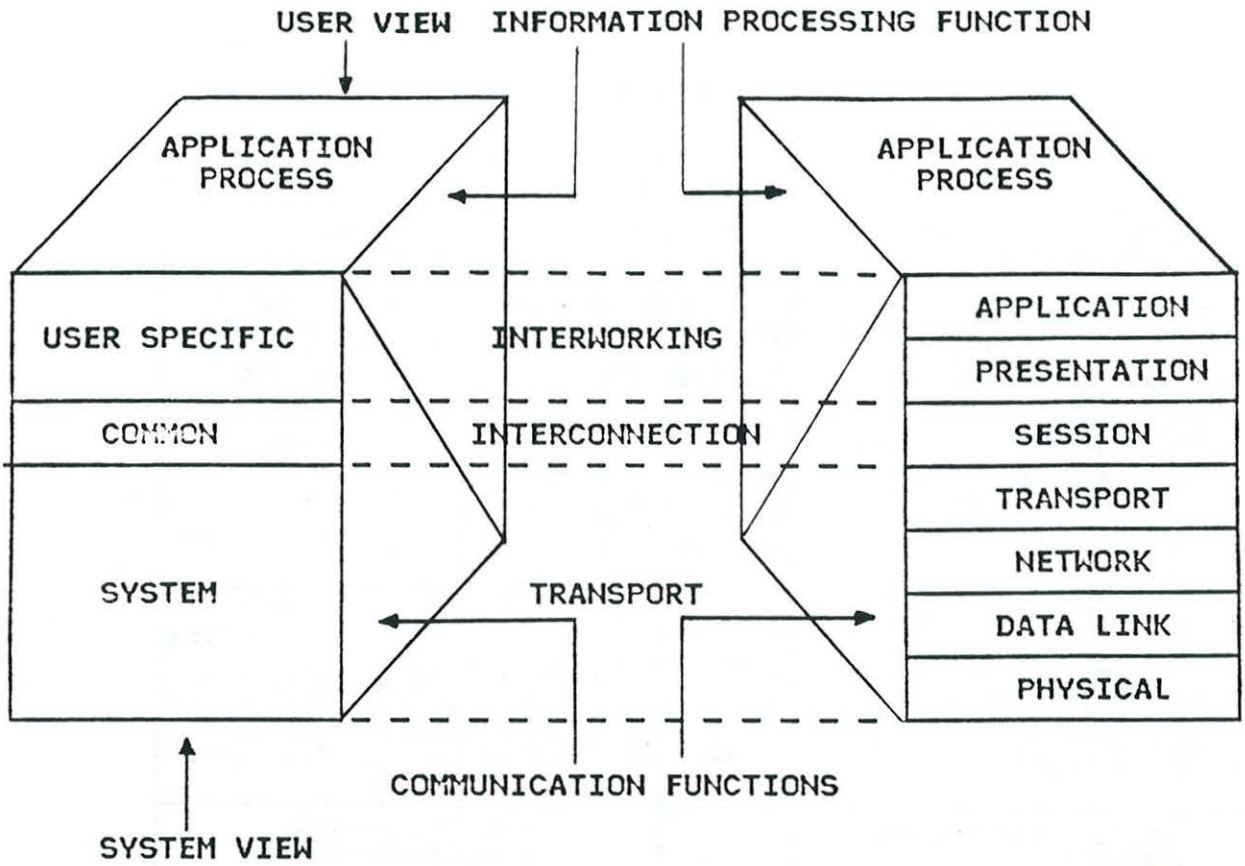
- Looking now at the synchronous distributed processing, it seems that we have to deal with two different situations:

  = Should standardized higher layer protocols defined, they would not cope with the existing applications (see the above discussion). As a consequence, any open system architecture should in this case not only define the communication environment at application level but also the pure processing environment, what is obviously beyond the scope of OSI today. In other words, besides defining standardized communication protocols, such an architecture would have to define also, for instance, Data Base managers, standardized syncpointing mechanisms (suitable for local resources) and their underlying concepts.

  = To succeed in coping with existing applications seems, in a first approach, to be a dream or a "lost paradise" in that it would be necessary to impact not only the system functions but also the user TPs to take into account the new capabilities. Once again, in this area, the synchro/resynchro mechanisms are the corner stone of upper layer protocols design.

| SNA ENTITY | MAIN PROPERTIES | OSI ENTITY | MAIN PROPERTIES |
|---|---|---|---|
| CONVERSATION | .DIALOGUE RULES<br>.TRANSFER SYNTAX<br>.SYNCHRONIZATION MECHANISMS | ASSOCIATION | .DIALOGUE RULES<br>.COMMON SEMANTIC<br>.TRANSFER SYNTAXES |
| SESSION | .ONE TO ONE MAPPING AT A GIVEN POINT IN TIME<br>.SERIALLY REUSABLE<br>.SUPPORTS DIALOGUE HANDLING<br>.PROVIDES END TO END DATA FLOW CONTROL MECHANISMS END SOME SYNCHRONIZATION PROTOCOLS (α) | PRESENT CONNECTION | .ONE TO ONE MAPPING (α)<br>.SUPPORTS TRANSFER SYNTAXES HANDLING |
| | | SESSION CONNECTION | .ONE TO ONE MAPPING<br>.NO REUSABLE<br>.SUPPORTS DIALOGUE HANDLING<br>.PROVIDES SYNCHRONIZATION PROTOCOLS |
| VIRTUAL ROUTE (VR) | .END TO END NODE PIPES AT LOGICAL LEVEL<br>.PROVIDES GLOBAL DATA FLOW CONTROL PROTOCOLS<br>.SESSION MULTIPLEXING<br>.DISRUPTIVE ROUTE SWITCHING | TRANSPORT CONNECTION | .ONE TO ONE MAPPING AT A GIVEN POINT IN TIME<br>.SERIALLY REUSABLE<br>.PROVIDES END TO END DATA FLOW CONTROL AND RESPONSE MECHANISMS<br>.NON DISRUPTIVE ROUTE SWITCHING |
| EXPLICIT ROUTE (ER) | .N TO 1 MAPPING CAPABILITY<br>.END TO END NODE PIPE AT PHYSICAK LEVEL<br>.IS MADE UP OF A COLLECTION OF ADJACENT NODES LINKED EACH OTHER BY MEANS OF TRANSMISSION GROUPS | NETWORK CONNECTION | .N TO 1 MAPPING (MULTIPLEXING CAPABILITY)<br>.GOES ACROSS ANY NUMBER OF SUBNETWORK |
| TRANSMISSION GROUPS | .N TO 1 MAPPING CAPABILITY<br>.LOGICAL LINK BETWEEN TWO ADJACENT NODES CONSISTING OF ANY NUMBER OF PHYSICAL PARALLEL LINKS | SUBNETWORKS | |

# COMMUNICATION FUNCTIONS AND OSI MODEL

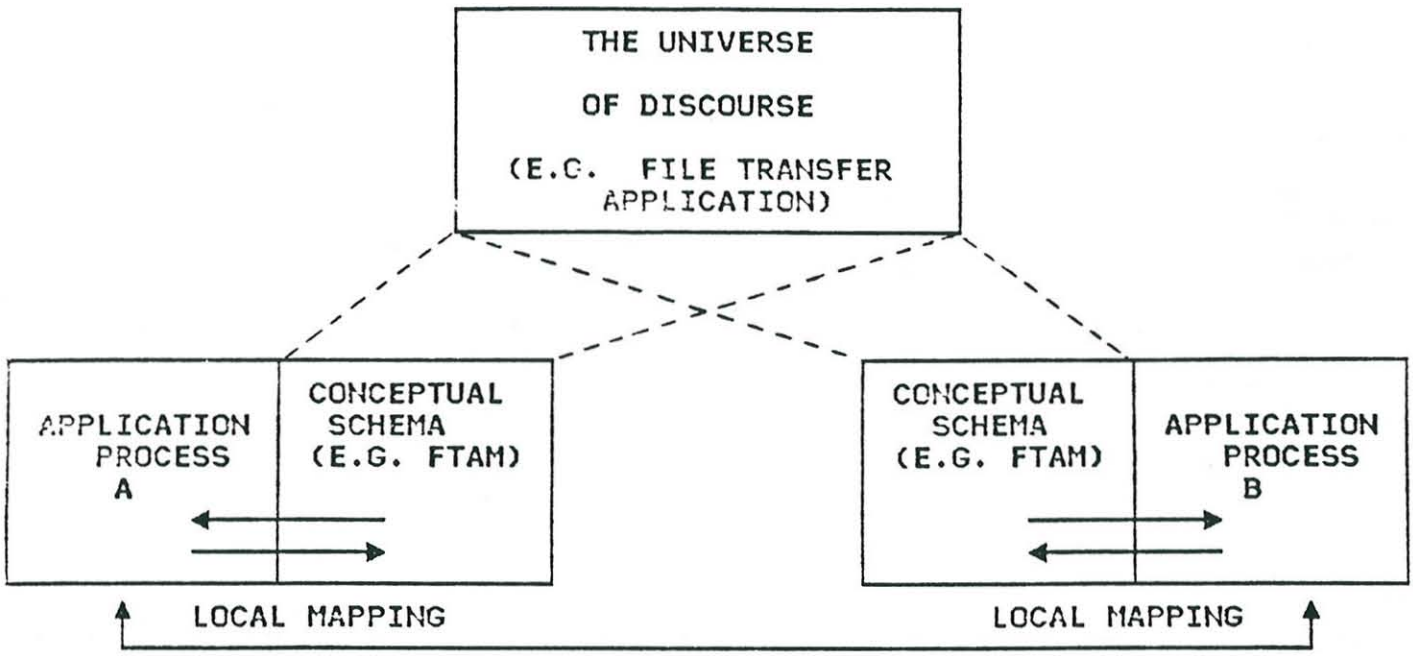



- DATA TRANSPORT COMMUNICATION FUNCTIONS NECESSARY TO TRANSPORT REPRESENTATION OF INFO. (DATA) FROM AN END SYSTEM TO ANOTHER WITH AN ACCEPTABLE LEVEL OF ERROR FOR THE APPLICATION PROCESS.

- INTERCONNECTION COMMUNICATION FUNCTIONS THAT ENABLE APPLICATION PROCESSES TO INTERCONNCT AND TO ENGAGE A DIALOGUE.

- INTERWORKING COMMUNICATION FUNCTIONS THAT ENABLE APPLICATION PROCESSES TO CARRY OUT A MEANINGFUL COMMUNICATION AND TO CARRY OUT PROCEDURES NECESSARY FOR DISTRIBUTED INFO. PROCESSING.

# OSI INTERWORWINK: SOME CONCEPTS

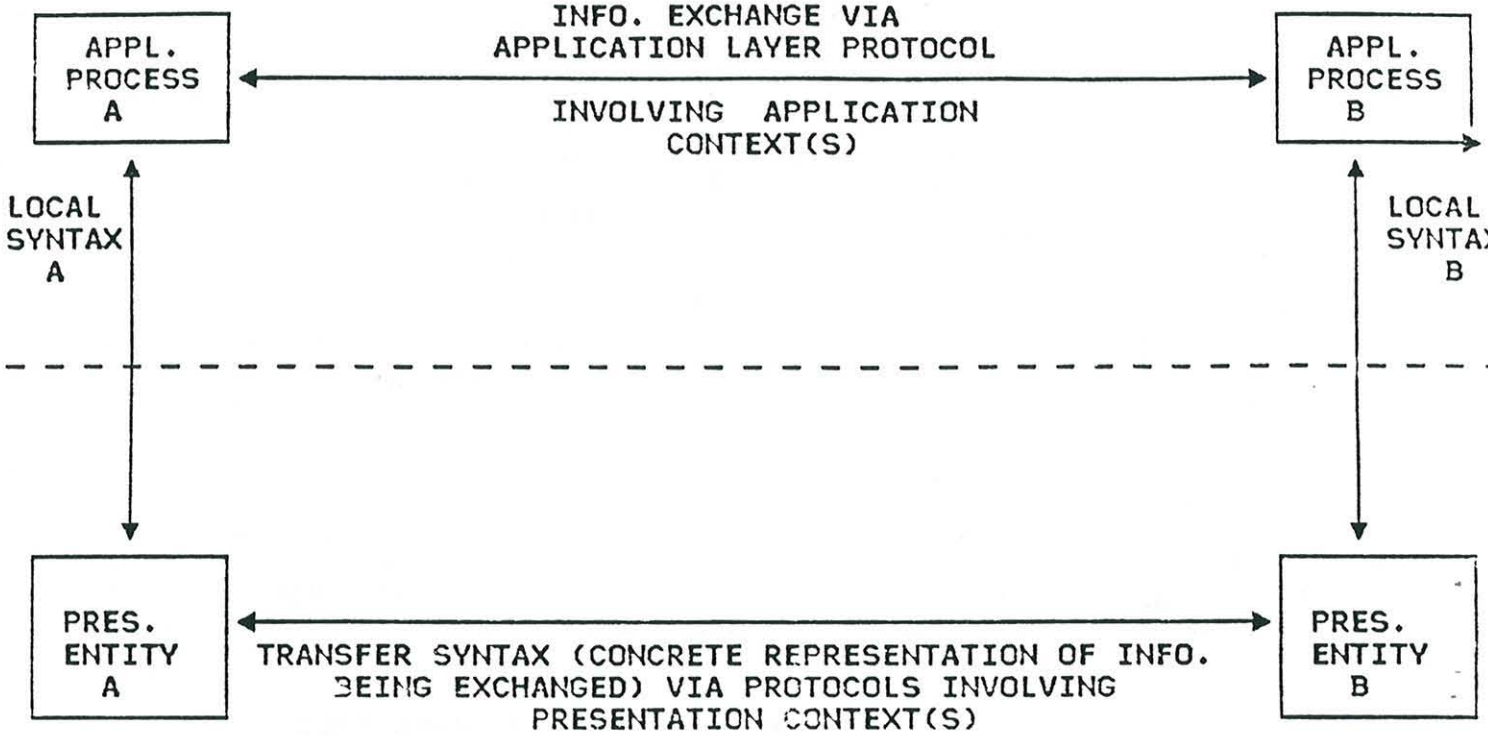



COUPLED APPLICATION PROCESSES

- THE UNIVERSE OF DISCOURSE IS THAT PART OF THE REAL OR HYPOTHETICAL WORLD WHICH IS UNDERSTOOD IN THE SAME WAY BY THE COMMUNICATING PARTNERS AND WHICH THEY AGREE TO COMMUNICATE ABOUT.

- THE CONCEPTUAL SCHEMA IS A FORMAL DESCRIPTION OF THE UNIVERSE OF DISCOURSE (ABSTRACT SYNTAX). IT DEFINES THE DATA ELEMENTS THAT CAN BE REFERRED TO IN THE COMMUNICATION AND THE ALLOWABLE OPERATIONS THAT CAN BE CARRIED OUT ON THESE DATA ELEMENTS

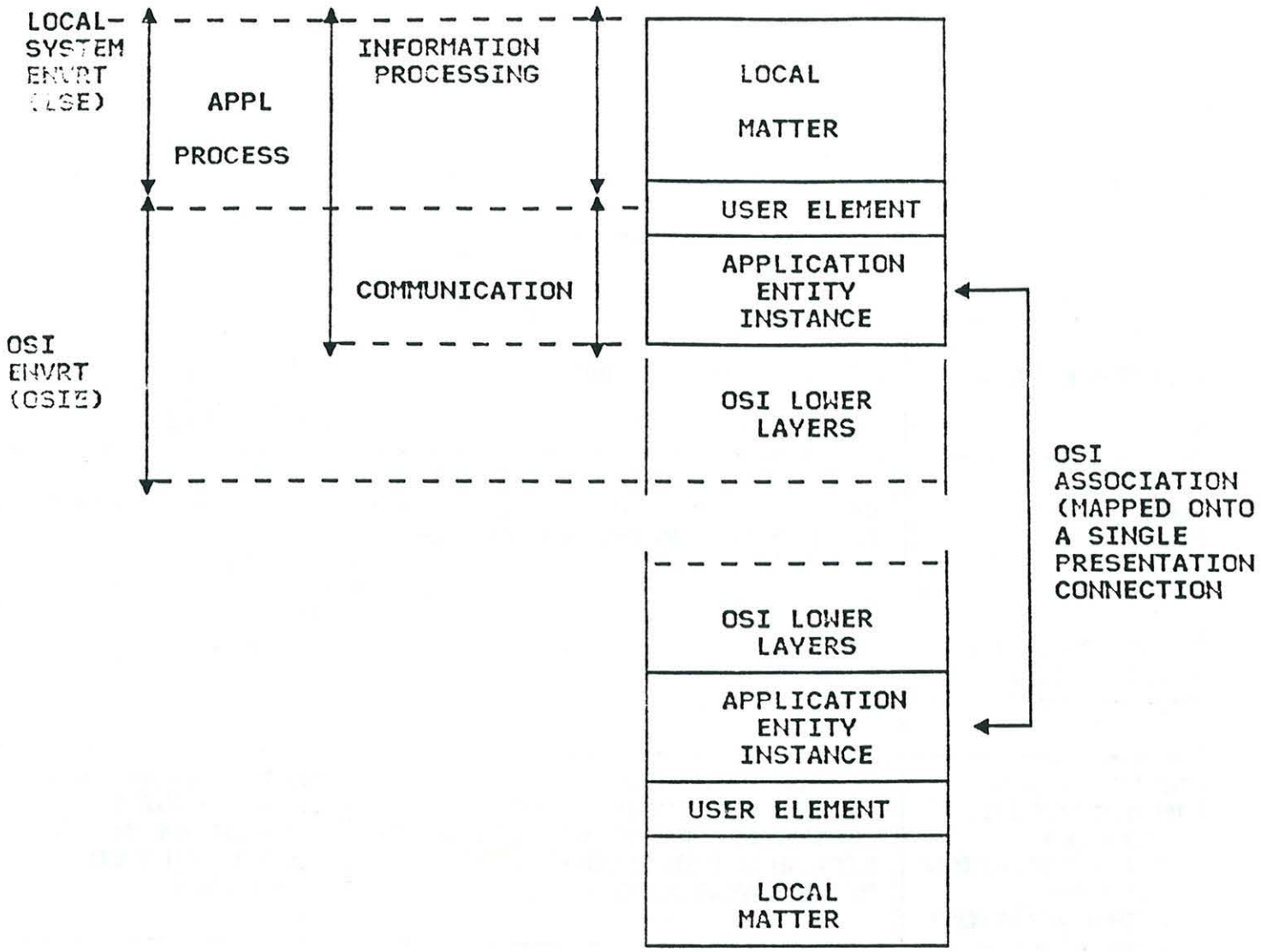
# OSI INTERWORKING: SOME CONCEPTS (2)




- THE APPLICATION CONTEXT IS THE PARTICULAR UNIVERSE OF DISCOURSE (WITH ITS ASSOCIATED CONCEPTUAL SCHEMA, ABSTRACT SYNTAX...) CHOSEN FOR USE IN COMMUNICATION BETWEEN APPLICATION PROCESSES.

- THE PRESENTATION CONTEXT IS AN ASSOCIATION BETWEEN THE SET OF ABSTRAC REQUIREMENTS OF AN APPLICATION LAYER STANDARD AND A TRANSFER SYNTAX CAPABLE OF SATISFYING THESE.

# APPLICATION PROCESS COMMUNICATION




- THE APPLICATION ASSOCIATION INVOLVES THE COUPLING OF TWO APPLICATION ENTITY INSTANCES. IT CONSISTS OF THE FOLLOWING FACILITIES
  - ASSOCIATION ESTABLISHMENT
  - ASSOCIATION RELEASE
  - CONTEXT MANIPULATION
  - INFORMATION TRANSFER

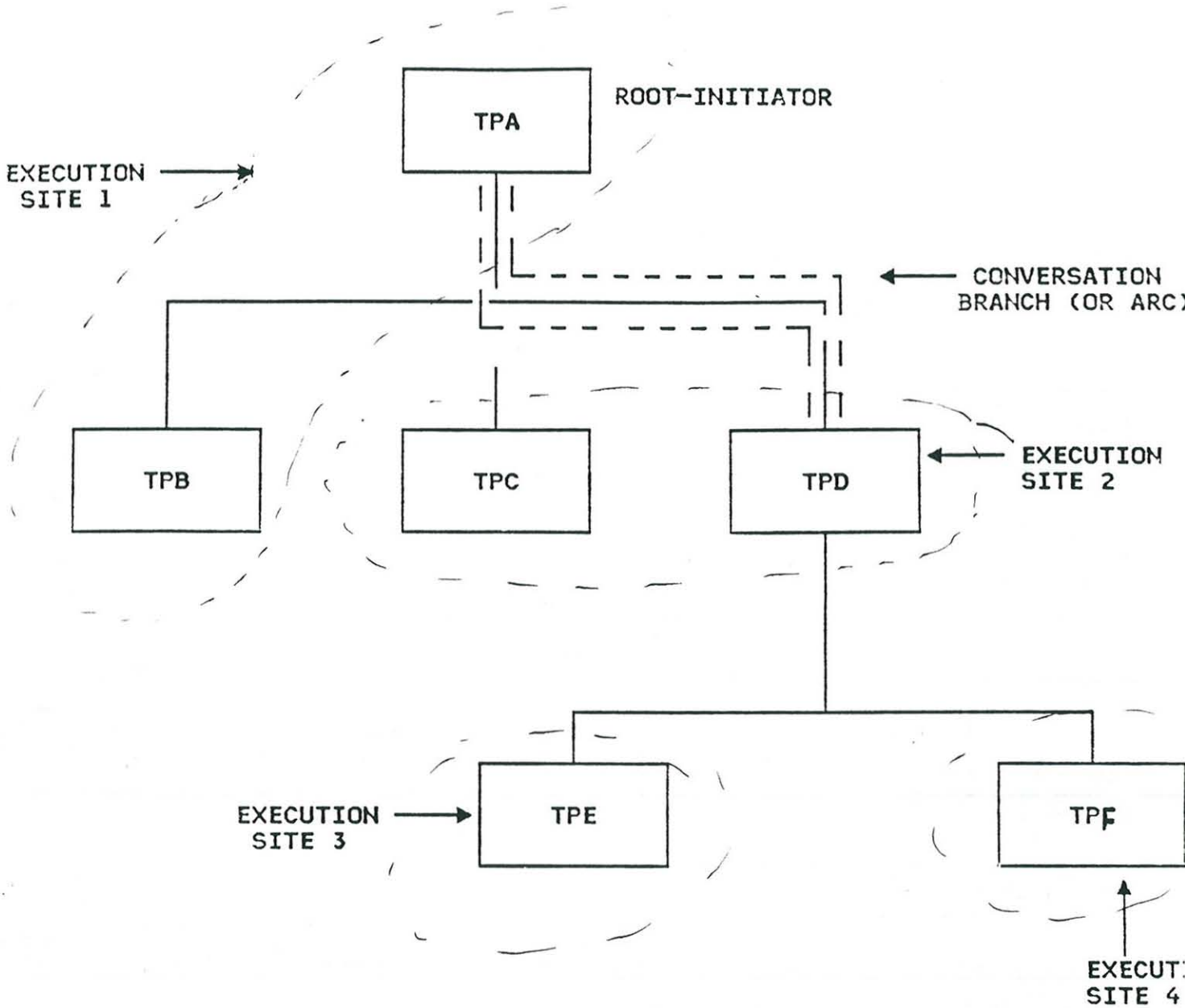
- THE APPLICAITON ENTITY IS A UNIQUE COLLECTION OF SERVICE ELEMENTS THAT PROVIDE PARTICULAR TYPES OF APPLICATION PROCESSES WITH THE COMMUNICATION SERVICES THEY REQUIRE

- SERVICE ELEMENTS ARE THE THINGS THAT GENERATE THE INDIVIDUAL PROTOCOLS EXCHANGES OR REQUEST SERVICES OF LOWER LAYERS. THEY CAN MAKE USE OF OTHER SERVICE ELEMENTS.

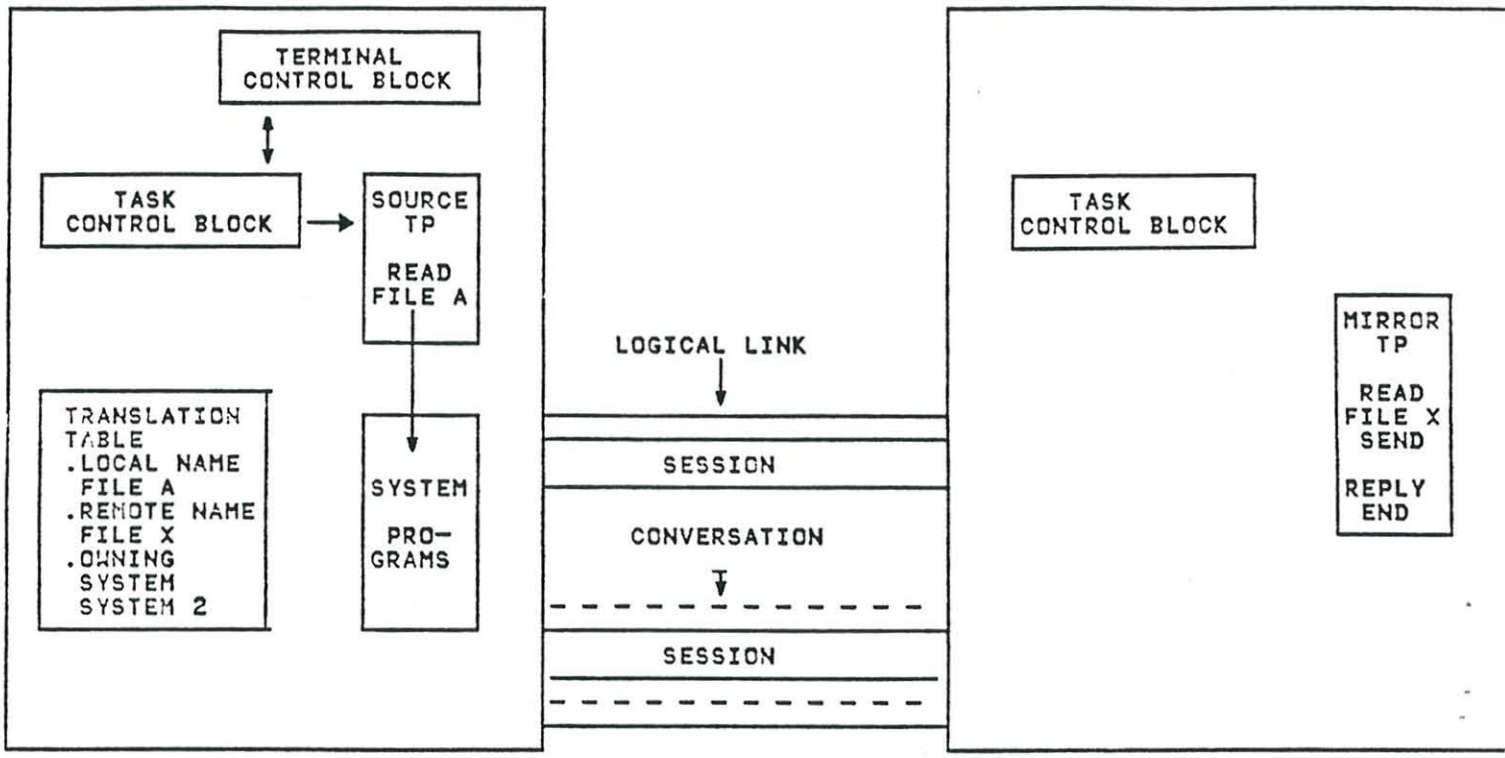
# DISTRIBUTED PROCESSING REQUIREMENTS

| D.P TYPE / MAJOR REQUIREMENTS | ASYNCHRONOUS (DELAYED DELIVERY) | SYNCHRONOUS |
|---|---|---|
| RESPONSE TIME | NOT A MAJOR CONCERN | VERY SENSITIVE. RESPONS TIME ALSO IS TO BE CONSISTENT |
| THROUGHPUT | VERY SENSITIVE MAINLY FOR BULK DATA TRANSFER (FILE TRANSFER JOB NETWORKING...) | NOT A MAJOR CONCERN |
| LOCAL RESOURCE SHARING AND CONCURRENCY CONTROL | NOT A MAJOR CONCERN SIMPLE MECHANISMS | MAJOR CONCERN COMPLEX MECHANISMS |
| COORDINATION OF DISTRIBUTED RESOURCES (SYNCHRONIZATION AND RE-SYNCHRONIZATION) | SIMPLE MECHANISMS BASED ON RESPONSES AND PRIVATE PROTOCOLS LIKE NOTI-FICATION (NOT ARCHITECTED RETRY CAPABILITY) | VERY COMPLEX AND COMPREHENSIVE MECHANISMS BASED ON ARCHITECTED PROTOCOLS |
| OVERALL RELIABILITY | NOT A MAJOR CONCERN TO THE EXTENT THERE ARE SIMPLE RETRY CAPABILITIES | MAJOR CONCERN TO THE EXTENT THERE ARE NO SIMPLE RETRY CAPABILITY |

# STRUCTURE OF THE PROCESSING TREE



## SIMPLE INQUIRY




SYSTEM 1

.TRANSACTION PROGRAM
READ FILE A
.SYSTEM ACTION MODULES
- TRANSLATE THE REQUEST
- IDENTIFY THE OWNING SYSTEM
- ACQUIRE AN APPROPRIATE SESSION
- SET UP A CONVERSATION
- SEND THE REQUEST

- TERMINATE THE CONVERSATION AND FREE THE UNDERLYING SESSION
- PASS BACK THE REPLY TO REQUESTING TP WHICH CONTINUES PROCESSING

TRANSMITTED INFORMATION

ATTACH TP NAME (MIRROR TPN)
READ FILE X

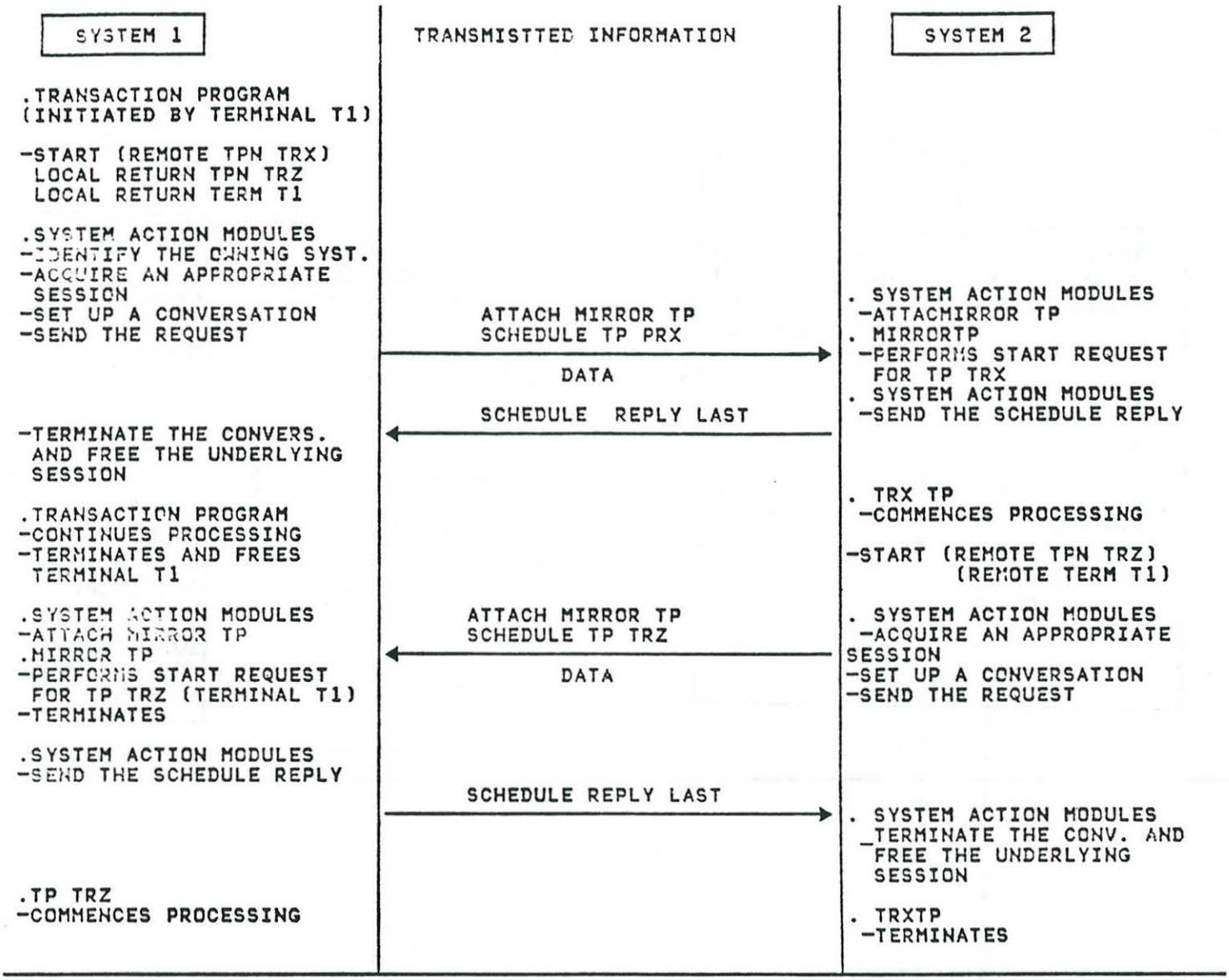
READ REPLY, LAST

SYSTEM 2

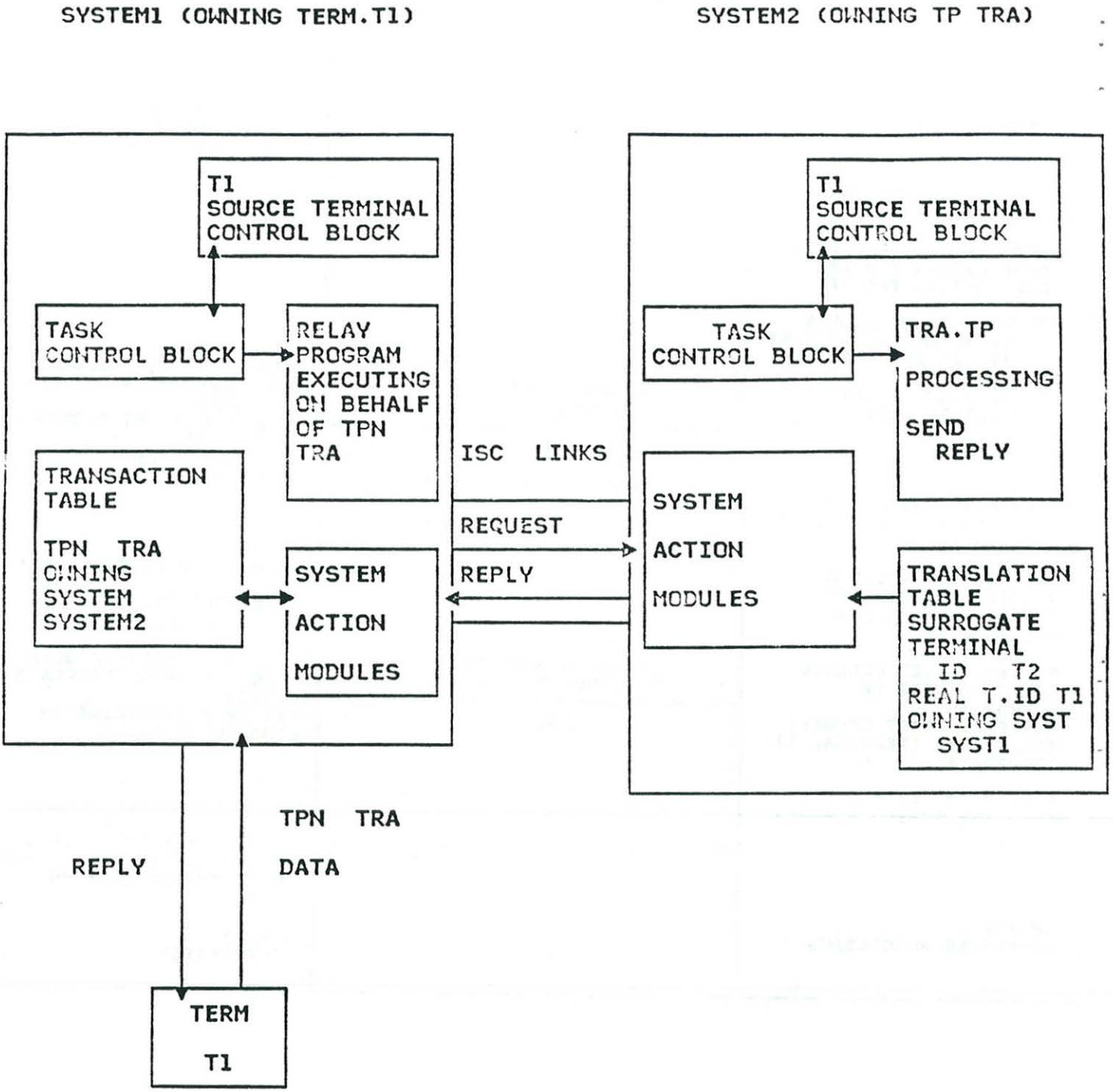
. SYSTEM ACTION MODULES
- ATTACH MIRROR TP
. MIRROR TRANS. PROG.
PERFORMS READ REQUEST
SENDS REPLY TERMINATE

FIGURE 8

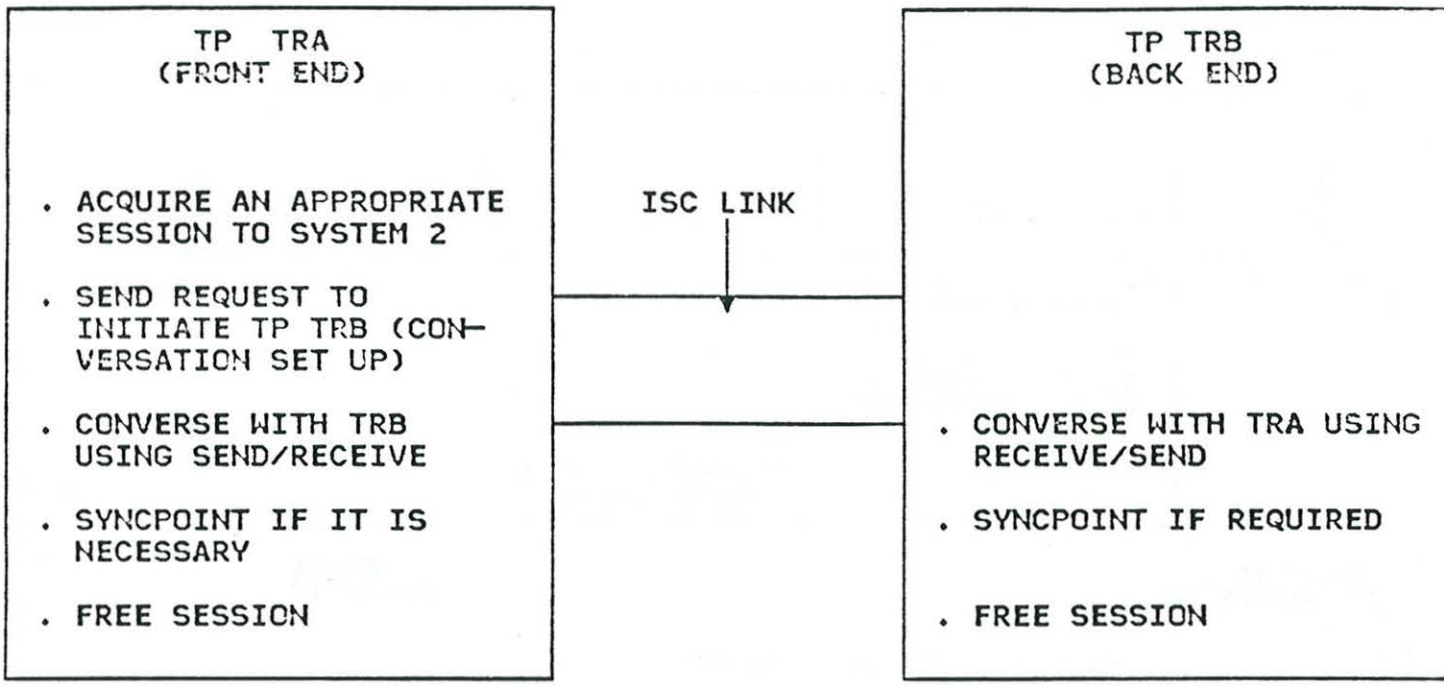
# ASYNCHRONOUS PROCESSING



# COMMUNICATION: TRANSACTION ROUTING

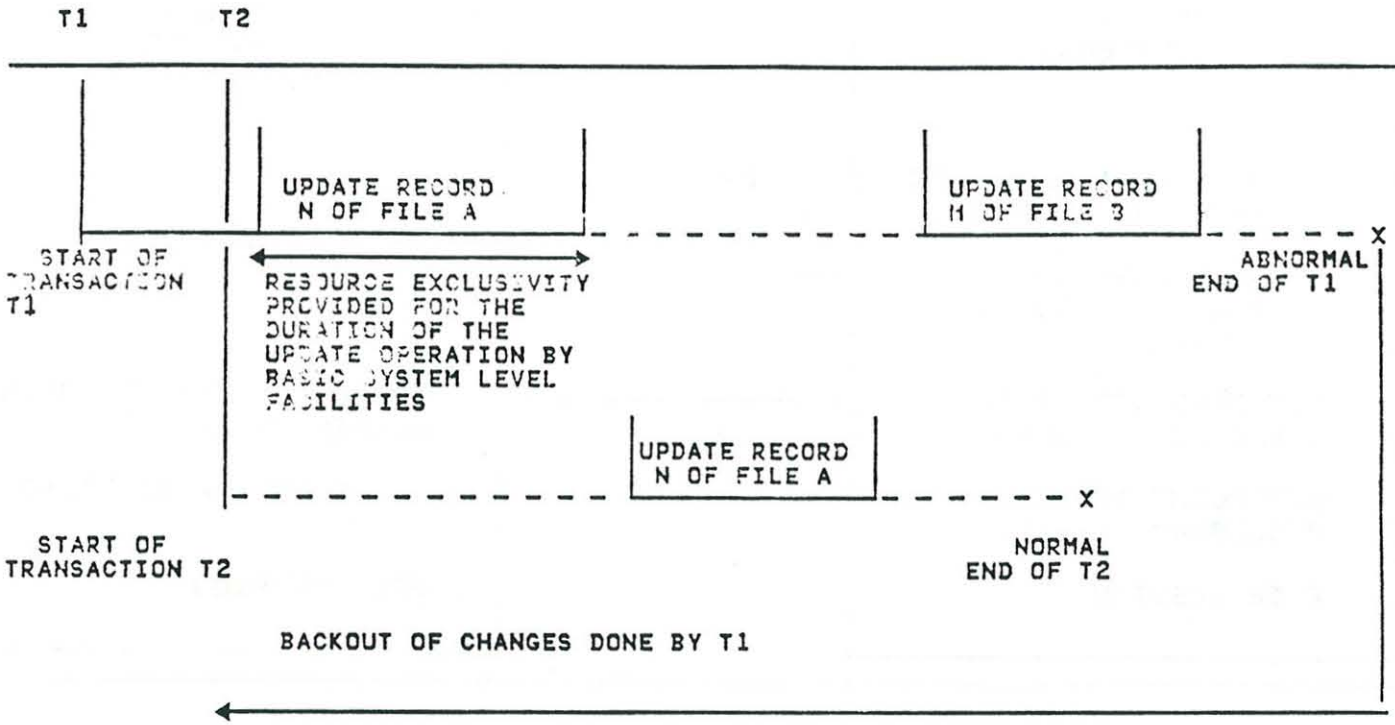


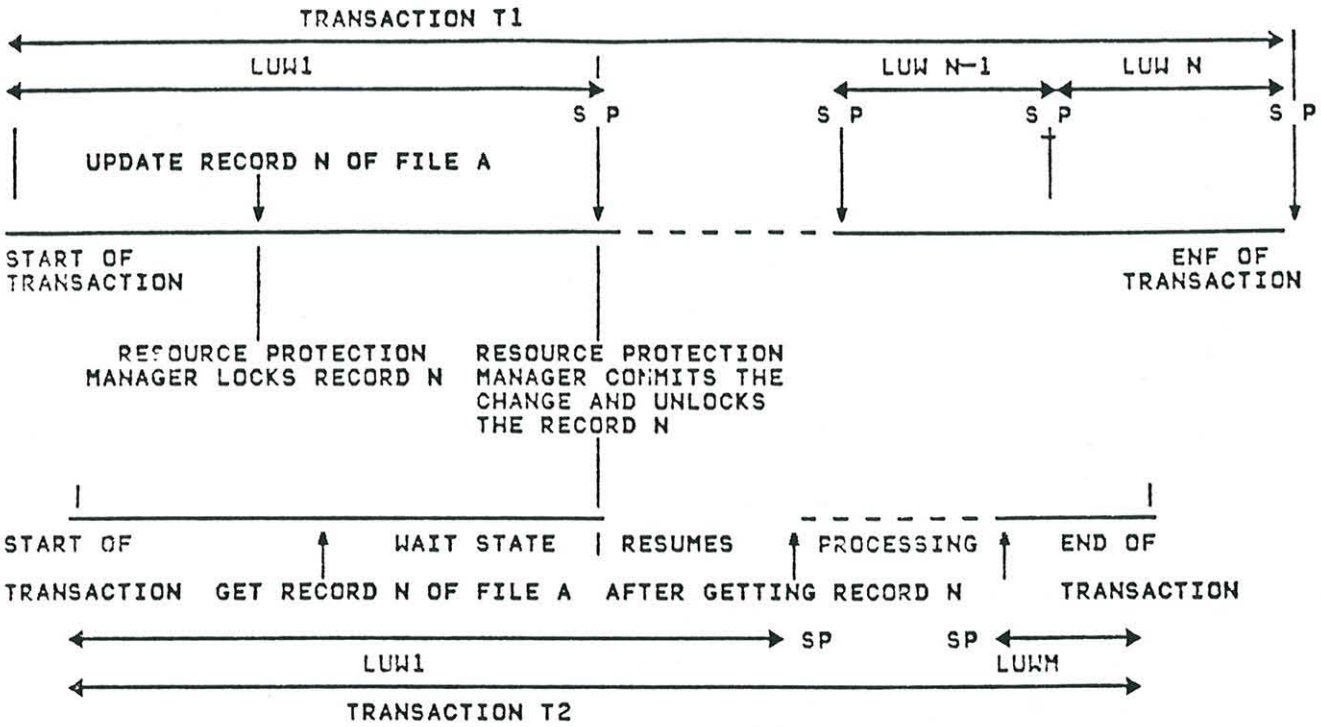
# DISTRIBUTED TRANSACTION PROCESSING



PROBLEM STATEMENT

TIME AXIS






.A TRANSACTION CONSISTS OF ANY NUMBER OF LOGICAL UNIT OF WORK

.A LOGICAL UNIT OF WORK IS AN ATOMIC SEQUENCE OF OPERATIONS DELIMITED BY TWO SYNCHRONIZATION POINTS

.A SYNCHRONIZATION POINT IS SCHEDULED EITHER EXPLICITELY BY THE TRANSACTION PROGRAM OR IMPLICITLY BY THE SYSTEM. AT SP TIME CHANGES OF RESOURCES ARE COMMITTED AND LOCKS ARE FREED.

.THE LOGICAL CONTENTION LEVEL IN A PROGRAM ISOLATION CONTEXT DEPENDS ON

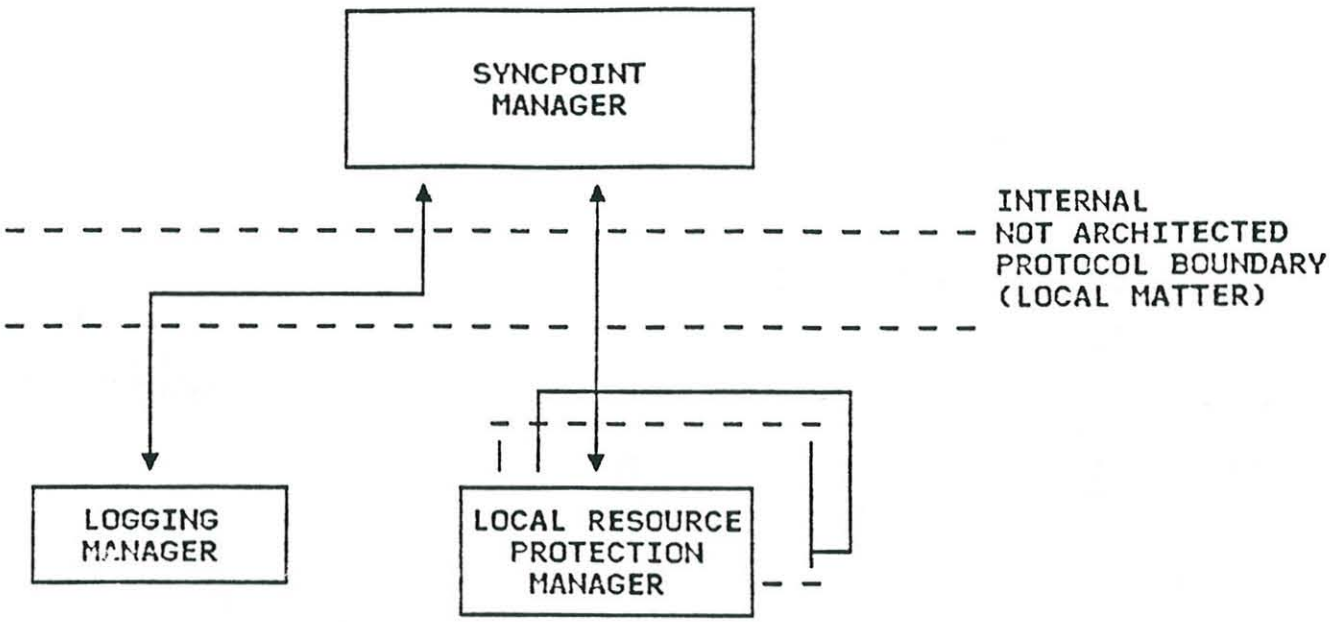
-THE INSTANTANEOUS TRANSACTION RATE (PARALLELISM LEVEL)

-THE GRANULARITY OF THE INVOLVED RESOURCES

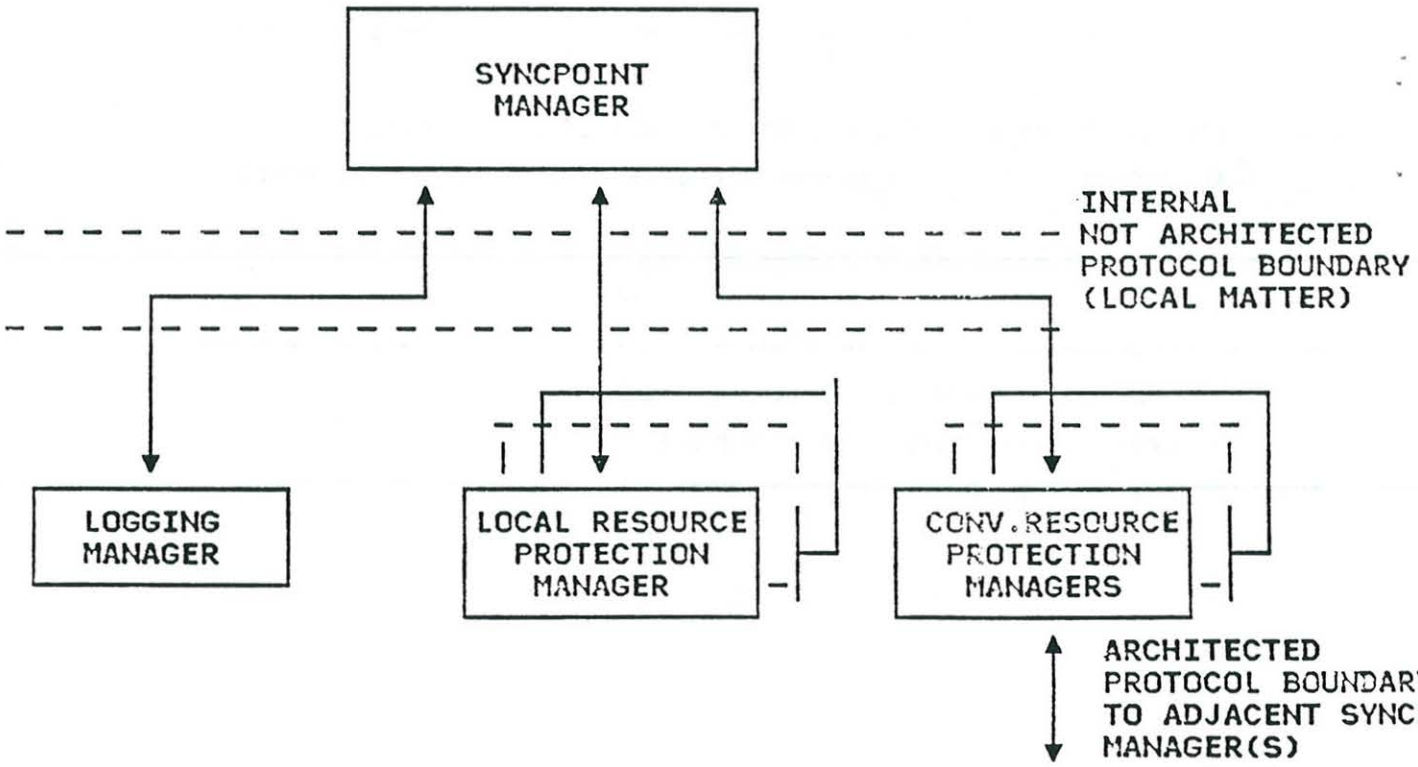
-THE LIFETIME OF THE LOGICAL UNIT OF WORK

# SYNCPOINT MANAGEMENT

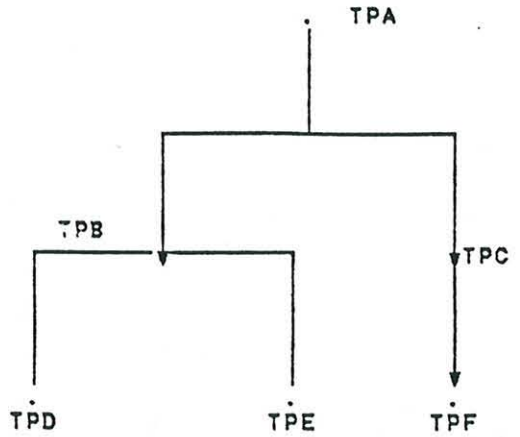
- SINGLE SYSTEM ENVIRONMENT




- DISTRIBUTED PROCESSING ENVIRONMENT



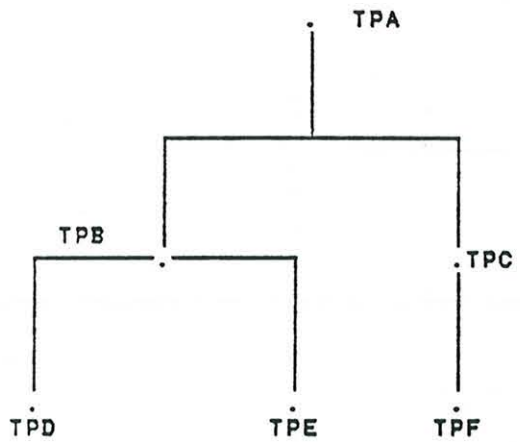
PROCESSING TREE



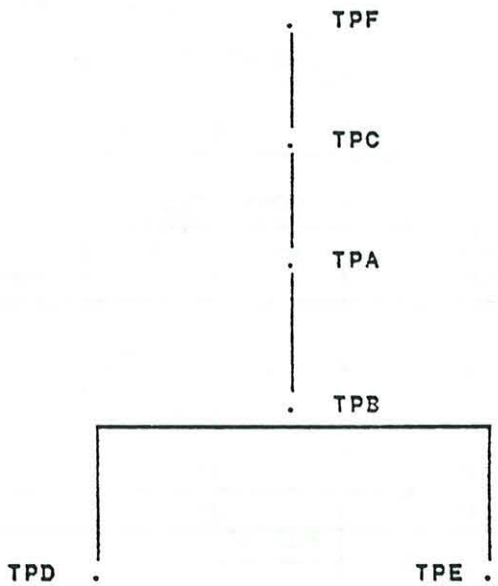

CORRESPONDING SYNCHRONIZATION TREES

USUAL STRUCTURE

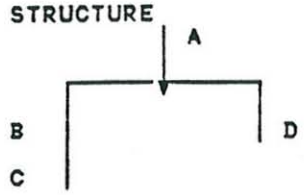
(SNA OR OSI C.C.R)



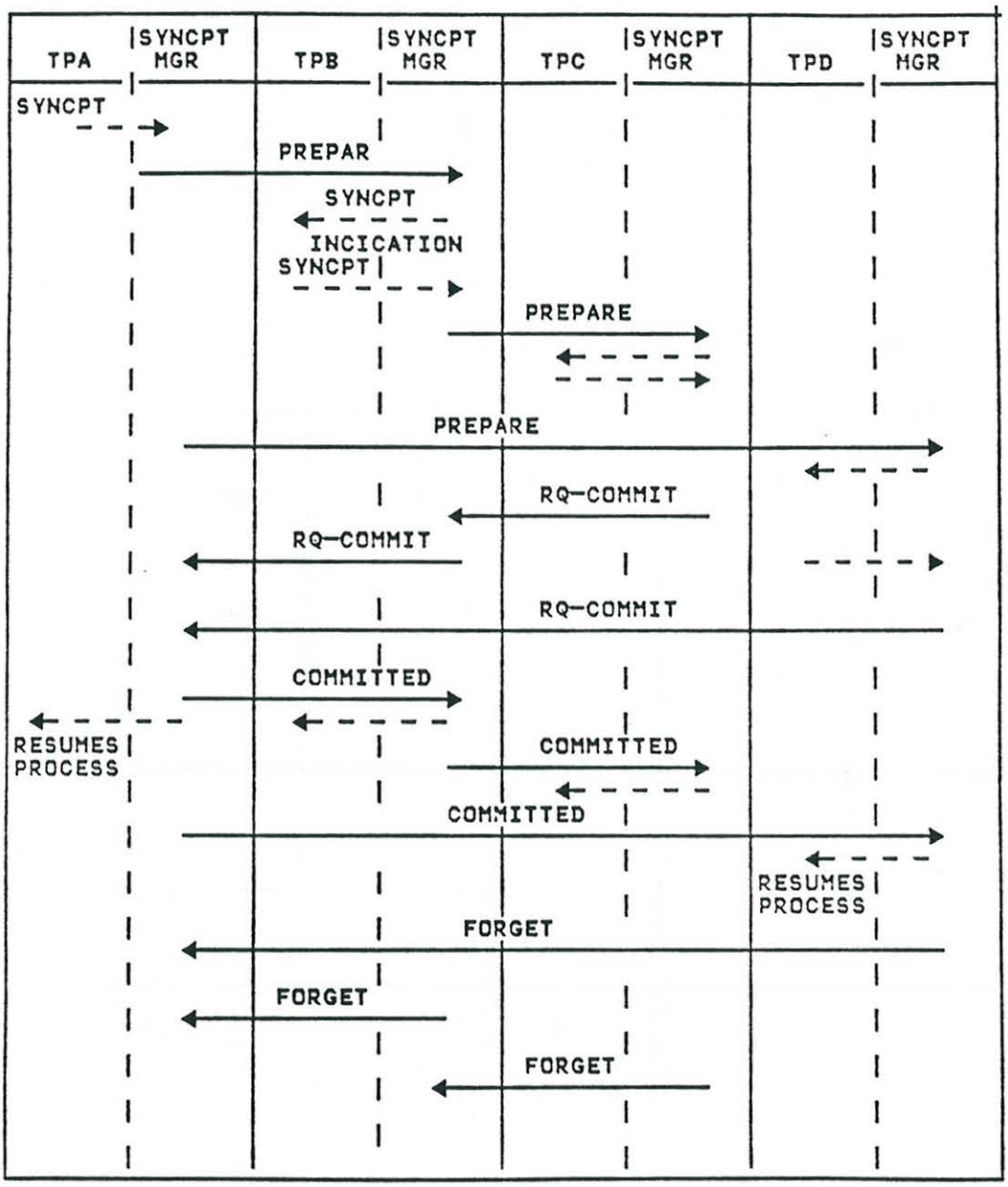

A POTENTIAL STRUCTURE (EXAMPLE)
(SNA)



- SYNCHRONIZATION TREE STRUCTURE




- DATA FLOWS



- PROBLEM STATEMENT