Poseidon House
Castle Park
CAMBRIDGE CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (0223) 323010**
INTERNATIONAL: **+44 223 323010**
FAX: **+44 223 359779**
UUCP: **...!uknet!ansa!apm**
ARPA Internet: **apm@ansa.co.uk**

ISA PROJECT

An Abstract Model For Groups

Abstract

The purpose of this document is to explore from a modelling point of view, the requirements for which the engineering model of groups, as described in [AR.002], provides one possible solution. It aims to identify the implicit assumptions and hence the options available to the designer.

It is found that a high degree of symmetry can be obtained in the structuring of groups. Whilst not an end in itself, this permits the model to be constructed from fewer distinct conceptual concepts.

Number:	APM/TR.020.00	
Task:	T.06	
Date:	January 13, 1992 9:00 am	
Type:	TR	(Technical Report)
Classification:	U	(Unrestricted)
Distribution:	G	(General)
Author:	AJT	(Alastair J. Tocher)

Contents

1 INTRODUCTION

1.1 Purpose

The purpose of this document is to explore from a modelling point of view, the requirements for which the engineering model of groups, as described in [OSKIEWICZ 90], provides one possible solution.

It aims to identify the implicit assumptions and hence the options available to the designer.

It is found that a high degree of symmetry can be obtained in the structuring of groups. Whilst not an end in itself, this permits the model to be constructed from fewer distinct conceptual concepts.

1.2 Overview

The document is presented in nine chapters.

Chapter 1 is this introduction.

Chapter 2 addresses the purpose of groups: why groups are needed and which facilities they provide.

Chapter 3 introduces a model for the conceptual structure of groups.

Chapter 4 contains notes on design choices which are available to implementers of groups.

Chapter 5 gives more details of the particular case where the mode of interaction is restricted to that of remote procedure call (RPC).

Chapter 6 outlines some areas for further investigation, whilst Chapter 7 draws some brief conclusions from the work.

Acknowledgements are made in Chapter 8 and the list of references is in Chapter 9.

Appendix A contains some introductory notes on the modelling technique and notations used in this paper for the benefit of those not yet familiar with the modelling style.

1.3 Background

The main inputs to this work have been the earlier work on groups which is presented in [OSKIEWICZ 90] and [OLSEN 91].

The mathematical notation used in this document is based on that of the specification notation Z [HAYES 87, SPIVEY 89] and the reader is assumed to be familiar with that notation. Some modifications of the notation have been made and the full list of basic mathematical notations used is to be found in [TOCHER 90a].

The graphical notation used is that of ANSA Graphical Object Notation [TOCHER 90c] which is based on the Object Theory described in [TOCHER

90b]. The modelling terms (*e.g.* "interface", "interaction", "connection", etc.) are used in the sense of [Ibid.]

2 PURPOSE OF GROUPS

A group is a structure (or mechanism) which provides a solution to a number of nonfunctional system design requirements.

A group is a structure which, from the point of view of a group user, supports abstraction from one or more design details of its construction. In particular it abstracts from the multiplicity of the group members, giving a user the impression of interacting with a single server.

2.1 What do groups provide?

A group is a structure which may provide the following features relative to a singleton entity providing the a given functional service:

- changed performance, and
- changed dependability

One might have expected at this point to see "improved" rather than merely "changed" applied to the listed features, but this weakened constraint on groups is indeed the strongest that can be formally applied. Notwithstanding that, it is often the intention that the group provide improved dependability or improved local performance. The kinds of improved dependability which might be obtained are listed in more detail in section 4.7.

2.2 Kinds of conformance

It has been found useful to distinguish different kinds of conformance in describing the requirements on groups[1]. We introduce a selection of distinct notions of conformance in this section and show how they are related. It will be shown later how they apply to groups.

2.2.1 Informational conformance

A given system is said to be **informationally conformant** to another system if and only if the given system embodies, and makes available to its users, at least the same information as the other system.

For example, two systems (or objects) might each provide a representation of propositional calculus. However one might provide operations **true, and**, and **not**, whilst the other provides operations **false**, and **implication**. Whilst it would not be possible to compare them in any operational sense, they do clearly embody the same informational content and would qualify for being considered informationally conformant, in this case in both directions: i.e. they are **informationally equivalent**.

---

1. More precise definitions of and distinctions between these kinds of conformance should be the topic of further study and a separate document. However, the potential distinctions are as yet unrecorded elsewhere and are included here so that reference can be made to some of them within this document. Informational conformance in particular relates to work on the information Projection which has not yet been done.

2.2.2 Signature conformance

A given interface is said to be **signature conformant** to another interface if and only if the interactions at the given interface include all the interactions of the other interface and the given interface could be safely substituted for the other interface without introducing any unexpected messages to the an object connected to that interface.

Signature conformance corresponds to the notion of type conformance in the ANSA Computational Model, but is intended to be more general in that it should also encompass signature conformance of non-message-based or RPC interfaces.

2.2.3 Temporal order conformance

A given interface is said to be **temporal order conformant** to another interface if and only if the interactions at the given interface may occur in the same temporal order as at the other interface.

This addresses *when* parts of an given interaction at the given interface must (or must not) occur, in *relative* terms. For example, an interaction is to occur *after* some other specific interaction has completed.

2.2.4 Operational conformance

A given system is said to be **operationally conformant** to another system if and only if the interfaces of the given system are both signature conformant and temporal order conformant to those of the other system.

2.2.5 Functional conformance

A given system is said to be **functionally conformant** to another system if and only if the given system is both informationally conformant and operationally conformant to the other system.

2.2.6 Absolute temporal conformance

A given interface is said to be **absolute temporally conformant** to another interface if and only if the given interface is temporal order conformant to the other interface and the interactions at the given interface satisfy the absolute (*i.e.* measurable) temporal requirements of the other.

This addresses when parts of an given interaction at the given interface must (or must not) occur, in measurable terms. For example, an interaction is to occur *within 100ms* after some other specific interaction has completed.

2.2.7 Absolute spatial conformance

A given interface is said to be **absolute spatially conformant** to another interface if and only if the interactions at the given interface satisfy the absolute spatial requirements of the other.

This addresses *where* parts of an given interaction at the given interface must (or must not) occur. For example, an interaction is to occur on a particular geographical site, or at one of a particular set of computer terminals.

2.2.8 Performance conformance

A given system is said to be **performance conformant** to another system if and only if the interfaces of the given system are both absolute temporally conformant and absolute spatially conformant to those of the other system.

This addresses the ability of a system to meet performance criteria relating to performing tasks within measurable time and location constraints.

2.2.9 Behavioural conformance

A given system is said to be **behaviourally conformant** to another system if and only if the given system is both functionally conformant and performance conformant to the other system.

2.3 Group creation and population control

This paper is not directly concerned with how groups are created or how groups population changes. The ability to discuss the creation and change in population of groups does, however, require that a definition of what does and does not constitute a group does exist.

This document specifically addresses the problem of defining what is and is not a group, and as such can act as the basis for further work on relations between groups which are at the heart of group creation and population changes.

3 GROUP CONCEPTUAL STRUCTURE

This chapter investigates the details of the structure of a group and how that structure is related to a single entity to which it conforms functionally.

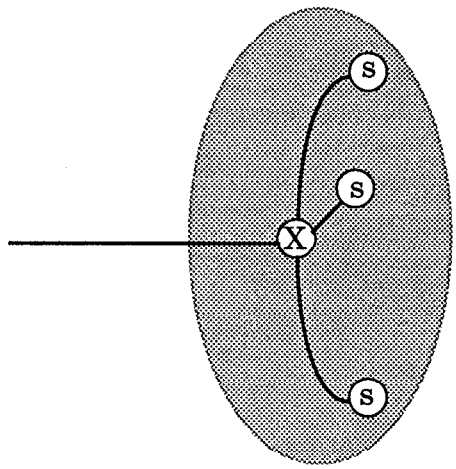
3.1 Definition

Conceptually, a **group** is a structure comprising

- a set of **members** each of which comprises an interface together with its associated entity, and
- a **group infrastructure** entity which is connected to the members and presents a single group interface.

A simplified example of this structure is illustrated in Figure 1: the infrastructure entity is labelled X and the members are labelled S.

Figure 1: Simplified group conceptual structure



The general case of a group both

- permits multiple member interfaces to be supported by the same entity and
- permits the same interface to act as multiple members.

In the latter case the sharing of the member interface may be modelled by the introduction of a multiplexer entity between the group infrastructure and the shared member interface.

It would be tedious to include these cases explicitly in every discussion, so in what follows these possibilities should continue to be borne in mind[1].

1. The provision of the same actual interface to act as distinct group members introduces a number of complications, the implications of which are not yet fully understood. This should be a topic of further study.

Whilst a group may, in general, have any number of members in particular cases membership may be restricted by the implementer for local design reasons[1].

3.2 Group basic structure derivation

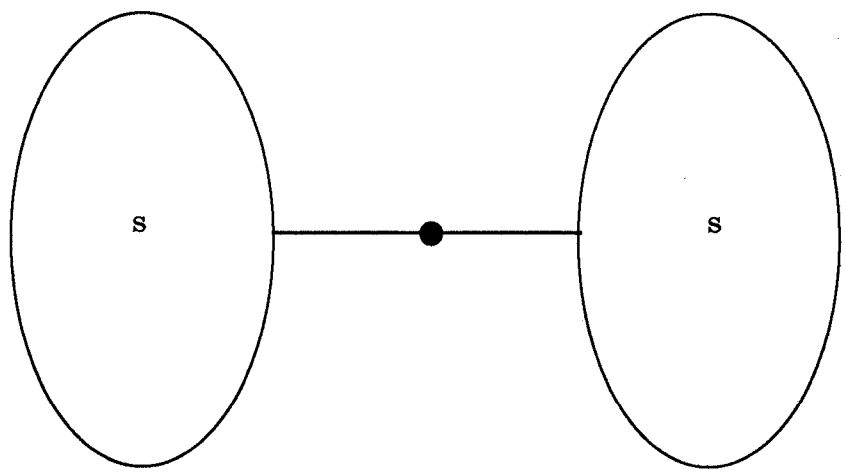
This section shows how a group structure may be derived from any given server. This exposition proceeds based on a model where directionality of interaction is not explicitly addressed. In this respect it differs from (insofar as it abstracts from) the ANSA computational Model. Directedness of interaction is dealt with in Chapter 5.

3.2.1 Simple remote servers

The computational model view of interaction in ANSA does not assume any necessary collocation of interacting entities within a system. Instead the case of remote interaction, via some medium of variable quality, is catered for in every case: the case of local interaction between entities is treated as a special case of this.

This simple view of potentially remote interaction is illustrated in Figure 2 in which three entities are shown: two servers, each named S, connected via a third entity, denoted by the anonymous black circle, acting as a medium. The symmetry of this model (and consequent conceptual simplification of the model) is a direct result of abstraction from the directedness of interaction.

Figure 2: Potentially Remote interaction model



1. Diagrams will, for traditional reasons of habit, show groups having three members. This is not meant to suggest any implicit restriction in number of members: groups may have any number of members subject to having at least one member.

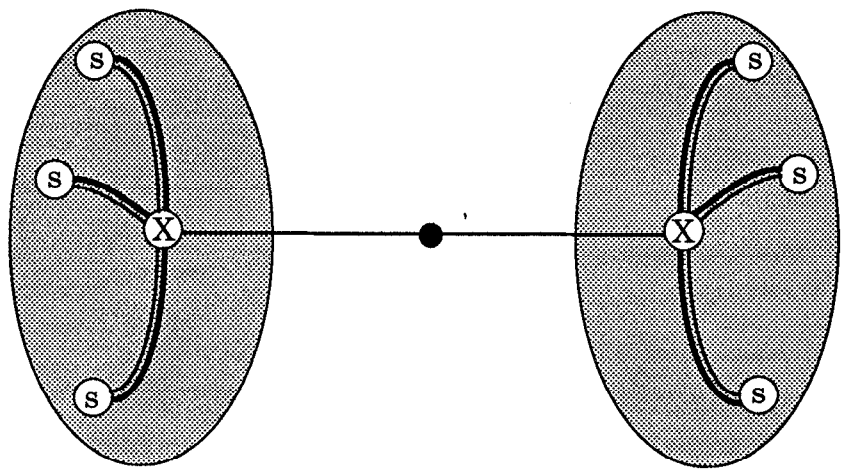
3.2.2 Basic group structure

Each server may be decomposed into a set of interacting components. In particular one may decompose each server into

- a set of **members**, each of which offers an interface which is functionally conformant[1] to that of the original server and each of which also potentially interacts with the other members via another interface, and
- a **group infrastructure** entity which provides the means of communication amongst the group members and between the group members and the group interface.

The resulting structure is shown in Figure 3. The group members are labelled s, whilst the infrastructure entities are labelled X. For convenience in the later decomposition of the infrastructure, the interactions between the infrastructure and each member are shown decomposed into two interfaces: that indicated by the thicker line represents interaction relating to the functionality of the server, whilst the thinner lines represent interaction relating to supporting the group structure.

Figure 3: Basic group structure



3.3 Distributed group structure derivation

This chapter shows how the basic group structure can be further decomposed in order to discuss the possibilities of greater distribution.

3.3.1 Decomposed basic group structure

Editorial: A name is needed, both here and elsewhere, for the entity which combines the functions of arbitration and distribution/collation.

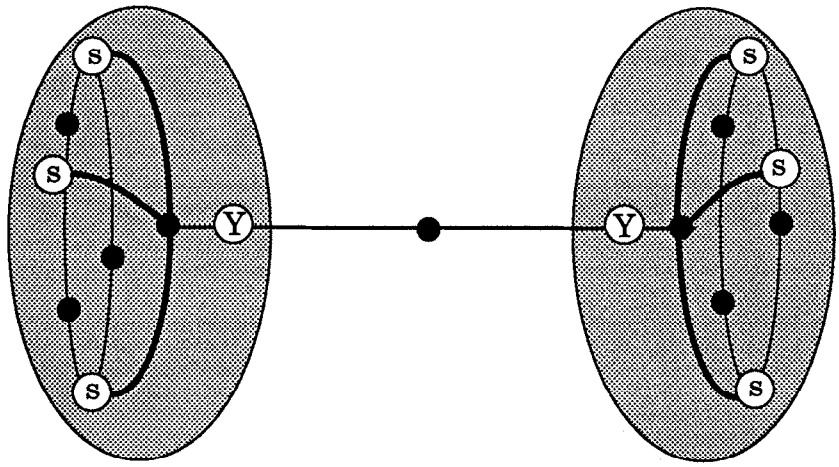
The group infrastructure may itself be decomposed into

1. That is, each member must provide all the functionality of the original server. This implicitly excludes the so called 'functionality distributed groups' of [OSKIE-WICZ 90] from the set of structures defined here as groups.

- a **subinfrastructure entity** which determines which parts of the externally visible service interaction are communicated to each member, and which parts of each interaction with the members are communicated externally, and
- medium entities interposed between the potentially interacting components

The resulting structure is shown in Figure 4. The subinfrastructure entities are labelled Y and, once again, entities acting as media are denoted by anonymous black circles.

Figure 4: Basic group structure



3.3.2 Distributing the medium

In Figure 3.4 a single medium entity was shown between the subinfrastructure and the distributed server. However the medium itself may be distributed (e.g. in order that it is not itself a single point of failure), and this is illustrated explicitly in Figure 5. The parts of the resulting structure corresponding to the previously undecomposed medium are shown within the shaded areas.

3.3.3 Decomposed subinfrastructure

Each subinfrastructure entity may, in turn, be decomposed into a set of interacting components. In particular one may decompose each subinfrastructure entity into

- an **arbiter**, A,
- a **distributor/collator**, D, and
- a **medium** interposed between the arbiter and distributor/collator

The resulting structure is shown in Figure 6. Once again, entities acting as media are denoted by anonymous black circles.

Figure 5: Distributed subinfrastructure-to-member medium group structure

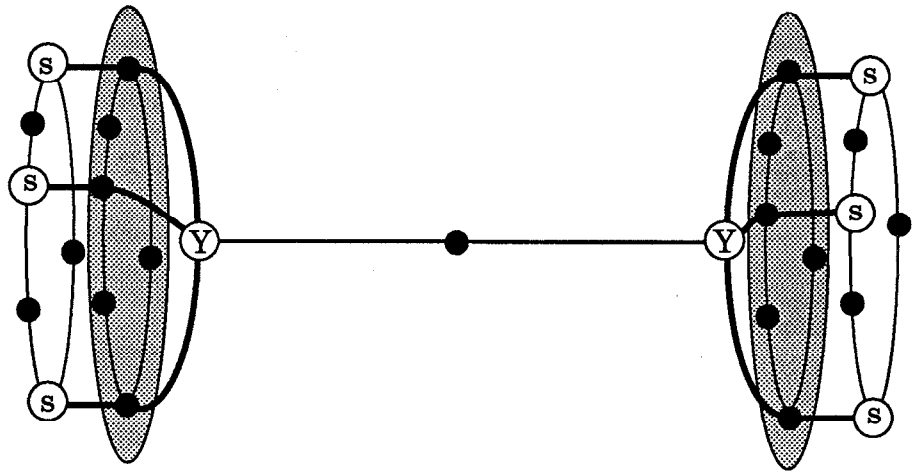
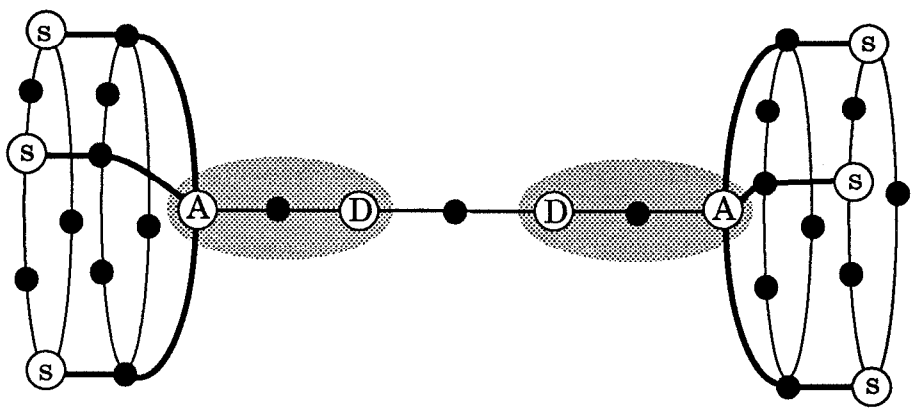


Figure 6: Decomposed subinfrastructure structure



3.3.3.1 Distributor / collator

The distributor/collator of a given group, as its name suggests, embodies the functionality associated with

- **collating** outgoing messages (*i.e.* messages from the group members) and
- **distributing** incoming messages (*i.e.* messages to the group members).

The **collation function,** as applied to outgoing messages, determines which messages are forwarded from the group, based on some policy applied to messages received from the group members via the arbiter. The collation function requires a means of identifying which incoming messages it is appropriate to collate (*i.e.* which incoming messages constitute the 'same' logical message for forwarding).

Conversely, the **distribution function,** as applied to incoming messages, determines which messages are forwarded to the group members via the arbiters, based on some policy applied to messages received from the group

user. The distribution function requires a means of marking each outgoing message in such a way as to permit recipients to decide whether they have received the 'same' message.

Distribution and collation functions are considered together since each effectively forms a converse of the other and, furthermore, the choices of policy for distribution and collation in any given group are closely linked. An overview of different policies which might be applied to collation and distribution is given in the next chapter.

3.3.3.2 Arbiter

The arbiter embodies functionalities relating to

- **ordering** of messages as seen by each member,
- **agreement** of order between members,
- **delay**

The **ordering function,** as applied to outgoing messages, determines a temporal partial order in which the messages are sent by each member; as applied to incoming messages, it determines the temporal partial order in which messages are forwarded to each member.

The **agreement function** determines whether the order of messages, whether to or from each member, is the same (or at least in some sense consistent[1]) at each member.

The **delay function,** as applied to outgoing messages, may introduce a (potentially unbounded) time delay to the forwarding of those messages; as applied to incoming messages it may introduce a similar (potentially unbounded) time delay to the forwarding of those messages.

This use of delays relates to the use of members as active or passive members (*vide* section 4.2). In the case of lazy members, outstanding messages may be delayed by the arbiter until eventually required.

3.3.4 Distributed arbiter

In much the same way as the members were distributed, so too can the arbiter be distributed, with one component per group member. The resulting structure is illustrated in Figure 7.

3.3.4.1 Distributed distributor/collator-arbiter medium

The medium between the distributor/collator and the distributed arbiter may also be distributed. The resulting structure is shown explicitly in Figure 8.

3.3.5 Distributed distributor/collator

The next step, of distributing the distributor/collators, is a little more complex than earlier steps.

Distributing the distributor/collator changes the interface through which the underlying medium interacts with the group: *i.e.* the interface is changed both

1. The meaning of "consistent" is application dependant and potentially a very complex issue.

Figure 7: Decomposed subinfrastructure structure

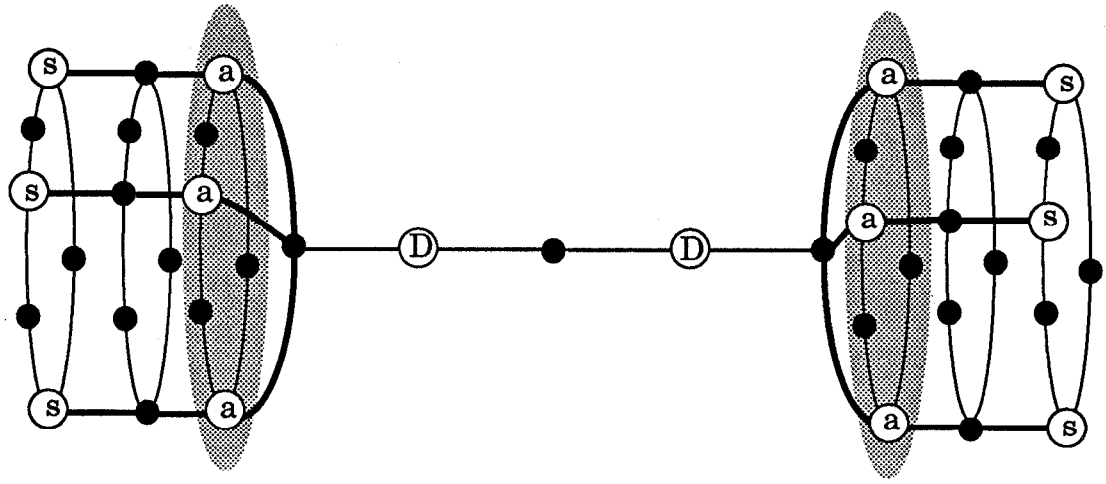
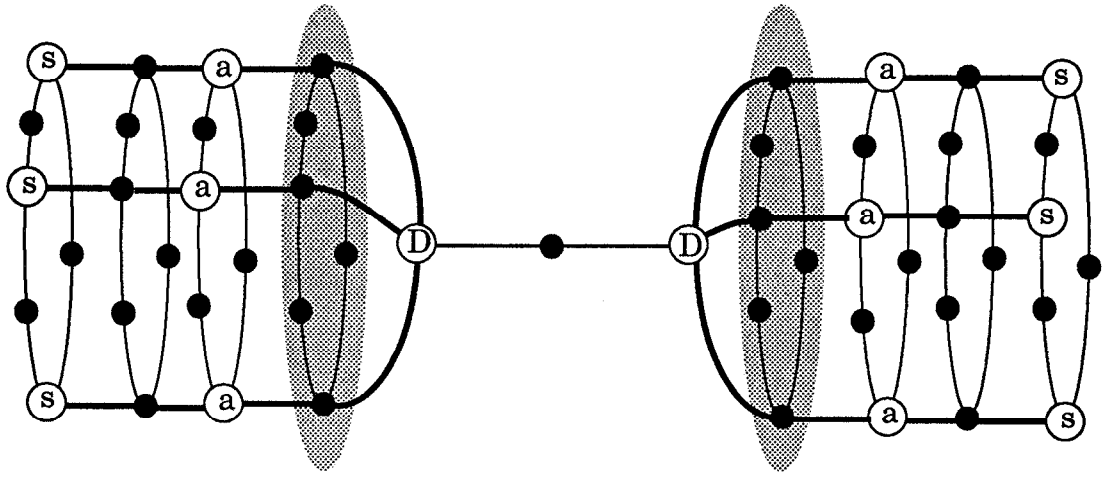


Figure 8: Distributed distributor/collator-to-arbiter medium group structure

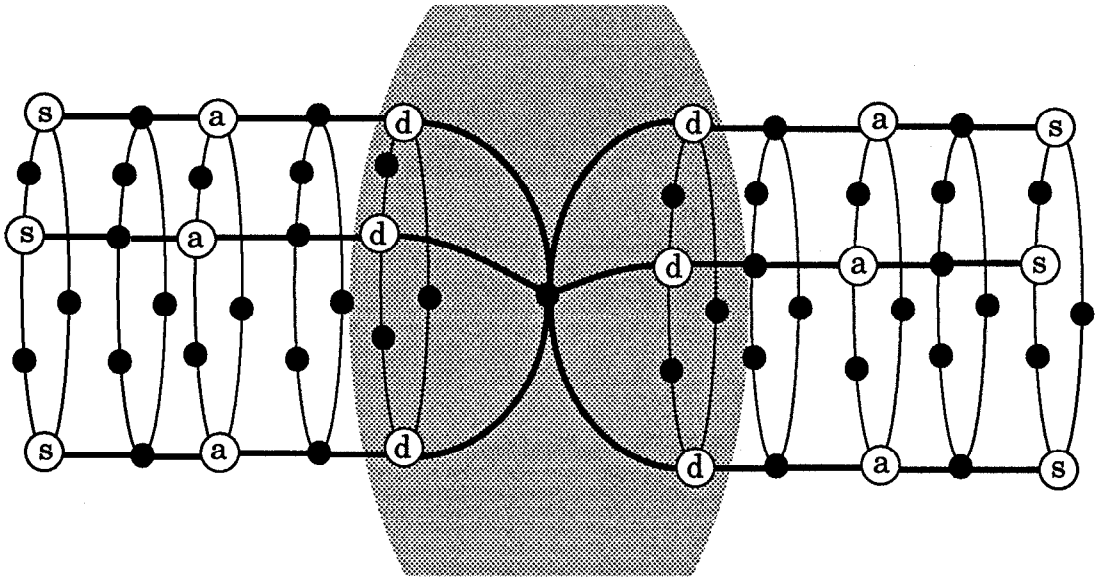


for the distributor/collators and the medium. In making that change the requirement that the distributed servers' interfaces conform functionally to the interface presented by the group as a whole cease to apply, since the groups interface per se ceases to exist as a single identifiable entity.

In distributing the distributor/collators it is necessary to consider the transformation of the combination of the two distributor/collators and the interposed medium rather than the transformation of the distributor/ collators in isolation. In particular, the substructure comprising the distributor/collators and the interposed medium is transformed into a new structure comprising distributed distributor/collators and a new medium having one interface per component of the distributed distributor/collators.

The resulting structure, with the shaded area indicating the extent of the transformation, is shown in Figure 9.

Figure 9: Distributed distributor/collator group structure



4 DESIGN CHOICES

This chapter discusses a number of possible design choices which are to be made in implementing any specific group structure. Whilst the categorisations and examples are intended to provide some insight into the range of possibilities, and to describe particularly what are thought to be commonly occurring cases, they are by no means intended to be exhaustive.

The design aspects are considered under six headings:

- distribution and collation policies,
- active and passive membership policies,
- physical collocation policies,
- distributed server simplification policy, and
- distributed distributor/collator simplification policy
- fault tolerance

4.1 Distribution and collation policies

The description of distribution and collation policies given here applies regardless of whether the messages in any specific case are viewed as requests or results.

4.1.1 Distribution policies

Whilst in principal there is a choice amongst policies for broadcasting to multiple members of a group, in practice the most important policy is that of broadcasting to the entire membership of the group.

4.1.1.1 Group broadcast

The **group broadcast** policy requires that every message received is forwarded to every known group member.

This is likely to be the most common distribution policy.

4.1.1.2 Group multicast

- The **group multicast** policy requires that every message received be forwarded to some non-empty subset of the group members.

This policy is appropriate in cases where either

- the messages being distributed are in response to messages received from a (non-empty) subset of the members of the calling group or
- the collation policy for replies arising from the messages being distributed requires messages only from a subset of the group members.

In the latter case the determination of how many messages to distribute may be made dynamically: receipt of sufficient replies may curtail distribution of further outgoing messages.

4.1.1.3 *Group unicast*

The **group unicast** policy requires that every message received be forwarded to precisely one of the group members.

Whilst this could be used, for example, in groups intended to provide improved local performance, with the single message sent to the nearest server group member, this policy does **not** permit any improvement in service reliability relative to a single undistributed server. This policy therefore appears to have fewer practical applications than the group broadcast and group multicast policies.

4.1.2 Collation policies

All collation policies are based on forwarding a message following receipt of some number of messages from the (distributed) server from which some known maximum number messages can be forthcoming. A classification of some possible policies is proposed below. It is recognised that some cited policies might have little application in practice but they are nevertheless included here for completeness.

Where the number of messages expected is known[1], the policies can be categorised according to the ratio of the number of messages awaited to the maximum number of messages expected. In particular

- 1:n - the forwarded message is dependent on receipt of a message from just a single member of the server group,
- m:n ($1 \leq m \leq n$) - the forwarded message is dependent on receipt of messages from more than one, but not all, the members of the server group,
- n:n - the forwarded message is dependent on receipt of messages from all members of the server group

In addition categorisation may be classified according to whether forwarding of a message depends on prior receipt of all expected messages.

If the number of messages expected is not known or if the collator has neither have knowledge of nor access to the operations on the messages to be collated[2], then the range of possible collation policies is reduced to the single "any message" policy (including first past the post).

---

1. In the case where population size is permitted to vary over time, ascertaining, in general, the expected number of messages is a nontrivial problem.

2. There is an unresolved problem relating to the computational of collation strategies, which shows up when trying to implement certain collation strategies. Policies which depend on the values of more than one of the received responses require that the collator can tell whether the values in the repsonses it receives from multiple invocations are 'the same'. The problem is that, in the computational model, these values are actually service reference and hence, in order to determine the spread of values in the responses from group members, another round of invocations will be required (to ask each response value wheter it thinks it is the 'same' as each of the others). Although these could possible be optimised away for the simpler types such as integers or strings, implementing collation policies, other than 'any message', for a general purpose group structure remains a potential problem.

4.1.2.1 Any message

The **any message** policy is perhaps the weakest of the policies in that any single message received may be forwarded.

A subsidiary policy decision may determine whether the forwarding of a message must await receipt of all expected messages (*i.e.* strict versus nonstrict evaluation) (*vide* temporally based policies below).

4.1.2.2 Proportion

The **proportion** policy permits forwarding of a message which occurs as some given proportion of all messages received.

A subsidiary policy decision may determine whether the forwarding of a message must await receipt of all expected messages (*i.e.* strict versus nonstrict evaluation) (*vide* temporally based policies below.).

4.1.2.3 Majority

The **majority** policy is a special case of the proportion policy where the given proportion is 50%.

4.1.2.4 Averages: mean, median, mode

An **average** policy permits forwarding of a message which is an average of all messages received. The choice of average could be mean, median, or mode, but note that the use of the mean requires some knowledge of the application since it could forward a message which was not received by the collator from any server member[1].

In the case of the mode a subsidiary policy decision may determine whether the forwarding of a message must await receipt of all expected messages (i.e. strict versus nonstrict evaluation) (vide temporally based policies below.).

4.1.2.5 Temporally based policies

In some cases the policy may impose a time limit upon the choice of messages used to determine the forwarded message.

For example, in the case of messages acting as replies to previously sent requests, a collation policy might require that only answers arriving within some time interval since one of the requests was sent be considered for application of one of the policies described above.

Alternatively, the time limit might be applied from receipt of the first message by the collator.

4.1.2.6 First past the post

The **first past the post** policy is a special case of the any message policy combined with the temporal restriction that all messages within zero time of the first to be received by the collator be considered in determining the forwarded message.

---

1. This might in fact prelude the use of the mean as a pratical collation policy in general.

4.1.3 Combining distribution and collation policies

Care is required in combining distribution and collation policies within a given group structure. Some combinations are inappropriate for any application: e.g. a distribution to a subset of the server group members and a collation policy that indefinitely awaits responses from all group members.

It is also interesting to note that it is not necessary that all messages, even within the same group, need follow the same distribution and collation policies. However the ability to make such choices of policy appears to depend on the semantics of the server interactions, and so would therefore not be appropriate in the case of a generic group structure capable of use in conjunction with arbitrary member entities.

4.2 Active and passive membership policies

A member is **active** with respect to a given message if and only if it receives that message as soon as possible and acts upon it as soon as possible.

A member is **passive** with respect to a given message if and only if the group can proceed without that member receiving the given message.

A member is said to be **strictly passive** if and only if it is passive and if prior to being activated, it need not first receive previously unreceived messages. (included in this category is that class of members of which the group infrastructure periodically directly updates the internal state of the passive member).

A member is **lazy** with respect to a given message if and only if it is passive and if prior to being activated, it cannot proceed without first receiving outstanding messages.

Within a given group it is the group infrastructure which determines which members are active or passive, not the members themselves.

4.2.1 Active replica groups

An **active replica group** is a group in which all members are active.

4.2.2 Passive replica groups

A **passive replica group** is a group in which one or more members, but not all members, are passive.

4.2.2.1 Strictly passive replica group

A **strictly passive replica group** is a passive replica group in which all passive members are strictly passive.

4.2.2.2 Lazy replica group

A lazy replica group is a passive replica group in which one or more of the passive members is lazy.

4.2.3 Active versus passive replica groups

More members may be made active where either there is an abundance of otherwise underutilised processing resource or there is a need to provide fast recovery from failure of another active member (though in the latter case strictly passive replicas might also be adequate).

Passive members, and lazy members in particular, whilst 'normally' low in processor usage, might require additional delay in recovering from failure in an active member owing to the subsequent lazy evaluation of the outstanding messages.

The rÔles of any member, as being active or passive, may change over time (*cf.* Birman's generalised coordinator/cohort model [BIRMAN 87]). There are no constraints on the minimum unit of time between which a group member may, in general, change rÔle[1].

4.3 Message ordering policies

Within the general group structure of Figure 9, it is possible to identify many possible partial orders over the messages between the distributed servers of the two groups.

In particular one can identify potentially distinct temporal orders at each interface between a conceptual component and its adjacent medium on the path between the distributed servers[2]:

- the temporal order of messages at the distributed server - medium boundary (1),
- the temporal order of messages at the medium - distributed arbiter boundary (2),
- the temporal order of messages at the distributed arbiter - medium boundary (3),
- the temporal order of messages at the medium - distributed distributor/ collator boundary (4), and
- the temporal order of messages at the distributed distributor/collator - medium boundary (5)

These orders apply both to outgoing and incoming messages on each of these interfaces. The boundaries at which these orders apply are indicated in Figure 10.

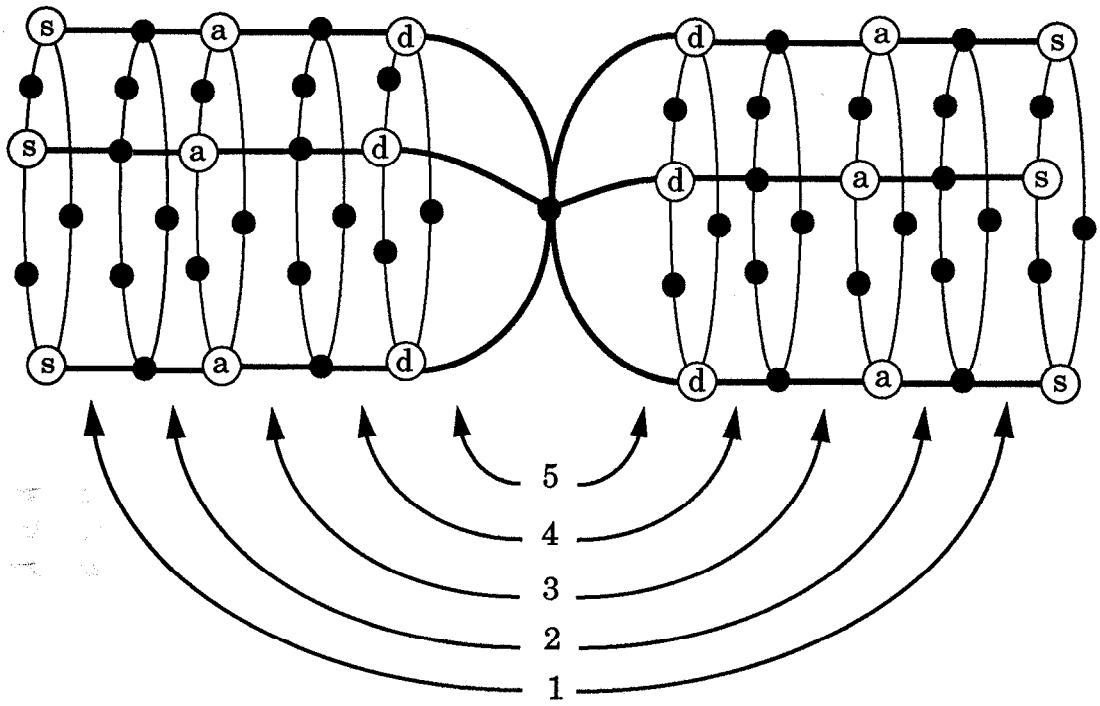
The relation between these orders at distinct interfaces may vary, although the choices of distribution and collation policy might restrict the possible

---

1. In the case of members providing a RPC-like interface, pratical considerations suggest that each member maintain a single role relative to any given individual invocation. Distinct invocations, even when overlapping in time, to the same member might however be dealt with differently (*i.e.* actively or passively).

2. The casual order of sending by a distributed server and the corresponding execution order on receipt, whilst important to an application, are not directly relevant to a discussion about groups.

Figure 10: Message temporal ordering points



combinations. As will be seen in the following sections, the choice of physical collocation policy and other possible simplifications also have an effect on the relation between the possible message orders.

4.4 Physical collocation policies

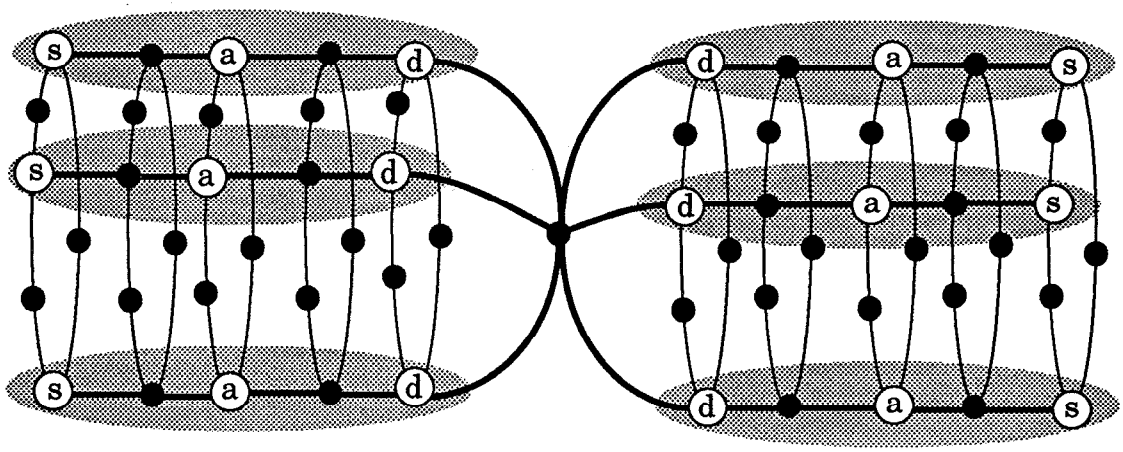
The choice of the physical location of the components of a group has an impact on the reliability, the performance, and the complexity of the construction of the group. In particular the physical collocation of components means in some cases that failure of one of the components, as a result of hardware error, can be assumed to imply failure of the other components collocated with the failed component: *i.e.* either all the components are free from the effects of hardware failure, or none are.

Some possible design choices relating to physical location are considered in this section.

4.4.1 Server-arbiter-distributor collocation

One might wish to physically collocate each server-arbiter-distributor triple. Such a grouping is illustrated in Figure 11, in which the shaded ovals indicate physical collocation of the contained logical entities.

Figure 11: Physical collocation group structure



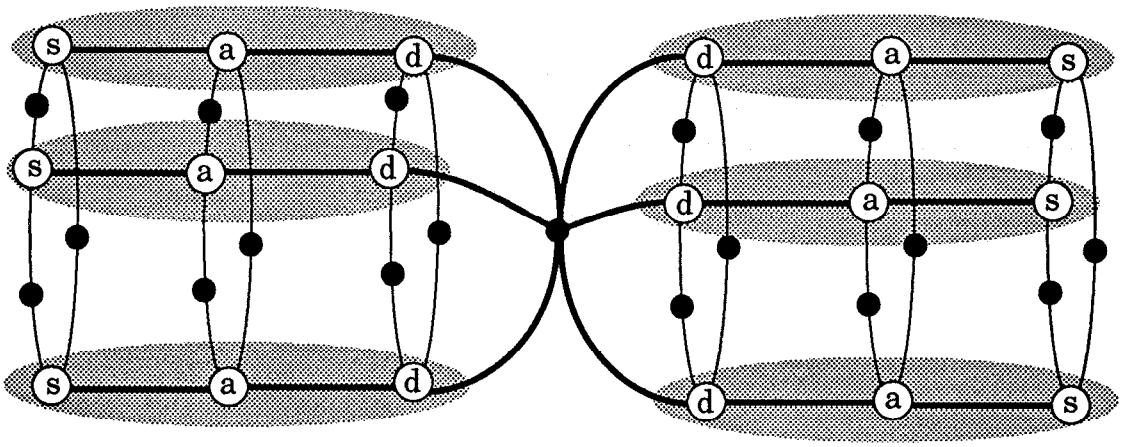
4.4.2 Simplified physical collocation

Having chosen to physically collocate each server-arbiter-distributor triple, further modelling simplifications can be applied. In particular, in the presence of highly reliable intralocation communications between entities one might make the following simplifying assumption:

- communication between the entities within a single physical location is 100% reliable.

Under this assumption, the resulting simplified design is illustrated in Figure 12.

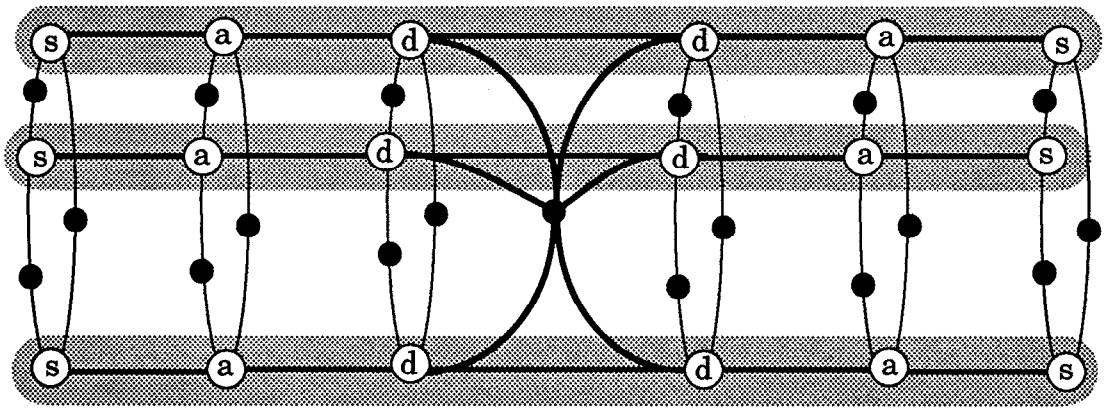
Figure 12: Simplified physical collocation group structure



4.4.3 Extended physical collocation

For improved local performance it can be useful to collocate members, arbiters, and distributor/collators from each of the distributed servers. In this case the communication between collocated distributors might also be assumed to be reliable. Such an arrangement is shown in Figure 13.

Figure 13: Simplified physical collocation group structure



In this case communication delay between the members of the groups can be reduced, but this improvement is dependent on the collation policy chosen. Whilst the distribution policy may remain one of communicating with all group members, the collation policy best suited to performance improvement is that of forwarding a result on receipt of the first reply: the *first-past-the-post* policy.

Unfortunately this choice of policy might not be acceptable in all cases. For example, in the case of a server in which some interactions cause state changes but others do not, whilst it might be possible to apply the first-past-the-post collation policy for interactions causing no state change, those interactions leading to state change in the server might require a collation policy based on results from more than one server group member for consistency reasons.

4.5 Distributed server simplification policy

For added reliability, one might choose to impose the following **simplifying requirement**:

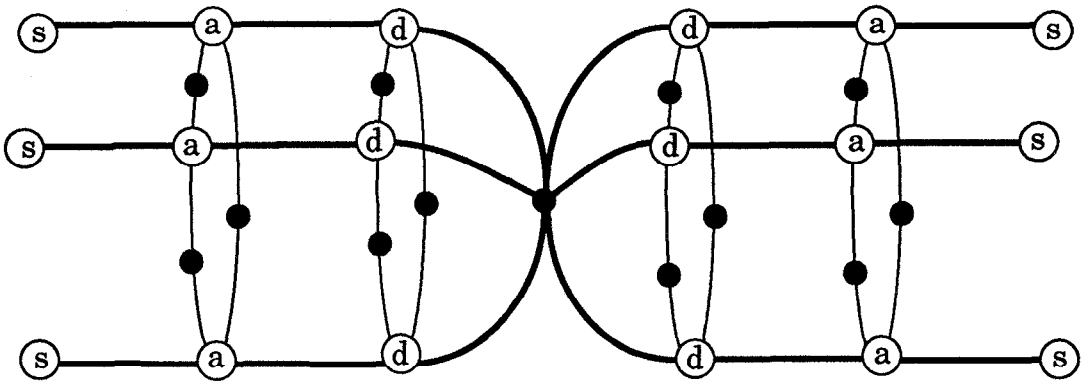
- direct communication between the members of the group is not permitted.

This simplification is important in the case where a group is constructed from preexisting members, each of which has only the group service interface, and is therefore incapable of interacting directly with other group members. (This also precludes the case of the infrastructure directly updating passive member state from that of active members.) Under this assumption, the resulting simplified design is illustrated in Figure 14.

4.6 Distributed distributor/collator simplification policy

In a similar way, one might choose to impose the following simplifying requirement:

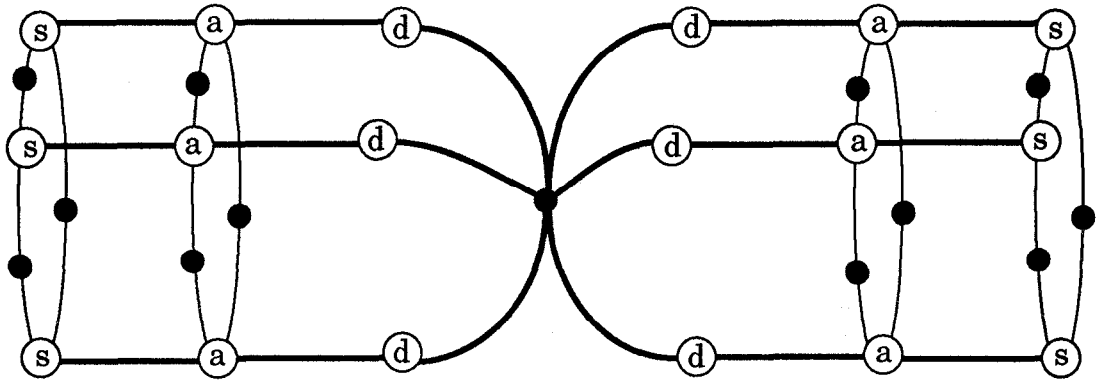
Figure 14: Distributed simplified server group structure



- direct communication between the distributor/collators of the group is not permitted.

Under this assumption, the resulting simplified design is illustrated in Figure 15.

Figure 15: Simplified physical collocation group structure



4.7 Fault tolerance

The ability of a group to withstand the occurrence of certain kinds of fault lies at the heart of their usefulness in achieving dependability. However it is important to identify precisely which kinds of faults a given group can withstand.

Three commonly catered for faults are those of

- hardware failure,
- software failure, and
- corruption and loss of messages

Resilience against such errors increases with the number of members in the group.

4.7.1 Hardware failure

The desire to tolerate hardware failure leads to physically distributed and replicated designs. In particular it promotes designs where distinct members are located on physically distinct hardware.

In an extreme case this would suggest the physical distribution of even the arbiter and distributor/collator associated with each member. However, since these conceptual elements exist primarily to support the member, the failure of the member would make the remaining arbiter and distributor/collator redundant. This leads to a physical distribution design as described in section 4.4.1.

Physical distribution must also be applied however to the media being used to communicate between conceptual components of the group: all the protection applied to other aspects will be wasted if all communication between components is dependent on a single medium which subsequently fails.

Within the broad category of hardware failure, it is necessary to distinguish the manner in which the hardware fails: fail-silent or fail-insane. The former can be dealt with relatively straightforwardly, in much the same way as dealing with a very slowly responding member, but the latter require Byzantine solutions in order to reach agreement on collated results. .

4.7.2 Software failure

The possibility of programming errors occurring in certain situations leads to the use of n-version programming. In the context of groups this would give rise to groups in which the members were not all derived from the same source code.

Protection from this kind of error, by the use of n-version programming, would seem to preclude the use of the group infrastructure to copy state directly from active to passive members.

Note that protection from software failure does not necessarily require physical distribution.

4.7.3 Corruption and loss of messages

The corruption and loss of messages in transit by the various media are catered for by the replication of messages necessary to communicate with distinct members and by appropriate choices of distribution and collation policies.

5 GROUPS AND RPC

In the case where the groups mode of communication is by RPC, some specific limitations on design choices are introduced.

5.1 Distributor/collator

The key issue in the design of the distributor/collator is how one identifies which messages denote the same logical message.

5.1.1 Requests

The crux of identifying all messages denoting the same logical message at all stages in the communication cycle between groups lies in the ability to identify outgoing requests, from distinct members, which denote the same logical message. Identification of all other messages in the RPC request-response cycle can be achieved based on a solution to that specific identification problem.

5.1.1.1 Outgoing

Identifying which outgoing requests, sent by distinct members denote the same logical message is difficult in general.

It is trivially achievable in the case where a group has just one member: the outgoing request can be tagged with a mark which distinguishes it from other logical messages sent from that member.

It also appears achievable at least in the case where all members exhibit identical behaviour, and are therefore completely deterministic. However this would impose stringent determinacy constraints on all other entities with which the requesting members interact which are likely to prove impractical in many real applications.

5.1.1.2 Incoming

Identifying which incoming requests denote the same logical message can be achieved relatively easily by requiring that, on sending, the requests have been tagged with some identifying mark common only to those requests denoting the same logical message.

This solution is dependent on solving the problem of identifying outgoing requests denoting the same logical message.

5.1.2 Responses

Provided a solution can be found to the identification of requests, the solution to that of identifying responses is straightforward.

5.1.2.1 Outgoing

Identifying outgoing responses is relatively easy for an RPC-based group. Responses, since they arise from receipt of earlier requests can be tagged with a mark derived from the identifying mark of the received request.

5.1.2.2 *Incoming*

Incoming responses which denote the same logical message can be readily identified if they have been tagged on sending as described in the previous section.

This is dependent on solving the problem of identifying outgoing responses denoting the same logical message.

5.2 Distribution groups and collation groups

Returning to the model for simplified group structure (vide section 3.1), this can be viewed, for a single group communicating using RPC, as composed of two structures:

- one conveying messages from the group interface to the members, and
- one conveying messages from the members to the group interface.

The former is referred to as a **distribution group**; the latter will be referred to as a **collation group**.

5.2.1 Distribution groups

A distribution group is composed of

- a **distributor** and
- one or more **members**.

5.2.1.1 *Connectivity*

The distributor, **d**, has an unconnected interface (connected ultimately to the user of the distribution group) and is connected to each of the member objects, s_i (i:1..n), by a single connection.

The interaction on the unconnected interface is **not** determined by the distributor; the interaction on each connection between the distributor and a member is determined by the distributor, **not** the member.

The connectivity aspects of a distribution group with two member objects are illustrated in Figure 16[1].

5.2.1.2 *Interface types*

Let the unconnected interface of the distributor have signature T_D. Then for each member, s_i (i:1..n), the signature, T_{Di}, of the connection between that member and the distributor must conform to the signature of the distributor's unconnected interface:

$$\forall i: 1..n \bullet T_{Di} \textbf{ signature\_conforms } T_D$$

The signature conformance aspects of a distribution group with two member objects are illustrated in Figure 17[2].

---

1. The use of arrowheads on arcs denoting connections to denote directedness of connections is an, as yet, informal extension to the Graphical Object notation.

Figure 16: Connectivity of a two member distribution group

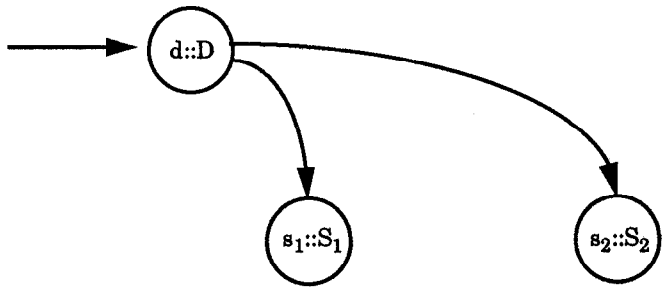
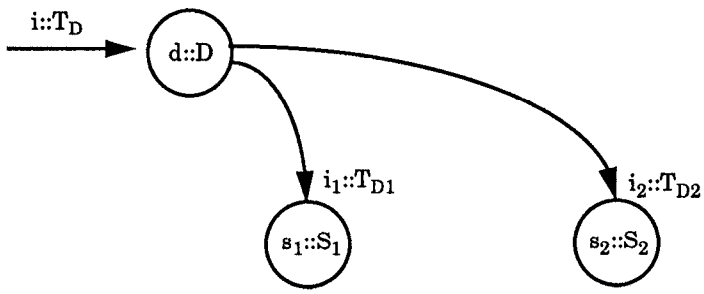


Figure 17: Connectivity and types of a two member distribution group



5.2.1.3 Distributor functional constraint

The product of the signatures of the interfaces of the distributor is of the form

T_D X T_{D1} X...X T_{Dn}

5.2.2 Collation groups

A collation group is composed of

- a collator and
- one or more members.

The collator has an unconnected interface with signature, T. The collator is connected to each of the members. The signature of the connection between

---

2. There remain some important questions regarding the conformance between the services provided by the members and that provided by the group: in particular, whilst signature conformance appears to be necessary, it is not clear that informational (or semantic) conformance is enforceable. Investigate the respective implications of (a) requiring semantic conformance and (b) not requiring it.

each member and the collator conforms to the signature of the group interface.

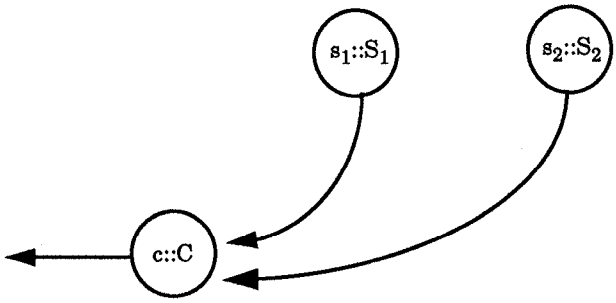
5.2.2.1 Connectivity

The collator, c, has an unconnected interface (connected ultimately to the user of the collation group) and is connected to each of the member objects, s_i (i:1..n), by a single connection.

The interaction on the unconnected interface is determined by the collator; the interaction on each connection between the collator and a member is determined by the member, **not** the collator.

The connectivity aspects of a collation group with two member objects are illustrated in Figure 18.

Figure 18: Connectivity of a two member collation group



5.2.2.2 Interface types

Let the unconnected interface of the collator have signature T_C. Then for each member, s_i (i:1..n), the signature of the group interface must conform to the signature, T_{Ci}, of the connection between that member and the collator:

$\forall i: 1..n \bullet T_C$ **signature_conforms** T_{Ci}

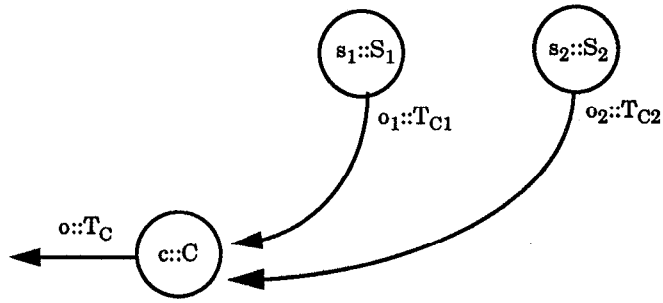
These type conformance aspects of a collation group with two member objects are illustrated in Figure 19.

5.2.2.3 Collator functional constraint

The product of the signatures of the interfaces of the collator is of the form

T_C **X** T_{C1} **X...X** T_{Cn}

Figure 19: Connectivity and types of a two member collation group



5.2.3 Duality of distribution and collation groups

Collation groups may be viewed as the dual of distribution groups[1], and vice versa, in that a collation group may have the same objects as a distribution group but with properties (*viz.* the directedness of interaction, the causal order of the distributor constraint, and the signature conformance of interfaces) reversed.

Consequently, properties of collation groups may be deduced as the duals of properties of distribution groups, and *vice versa.*

5.2.4 Combining distribution and collation groups

Distribution groups and collation groups may be combined by sharing members between groups. An informal distinction can be made between two kinds of such combination.

It is commonly the case that a user expects a response from a group of members based on a previously distributed request. Thus the user both wishes the members to be members both of a distribution group and a collation group and wishes to interact with both the distributor and collator of the respective groups.

The informal distinction to be made lies in whether the group members are also members of further collation and distribution groups: if the group members do not make shared use of further objects in providing the result, then the group is said to be **simple distribution and collation group**; if use is made of a further shared object then the group is said to be **complex distribution and collation group**.

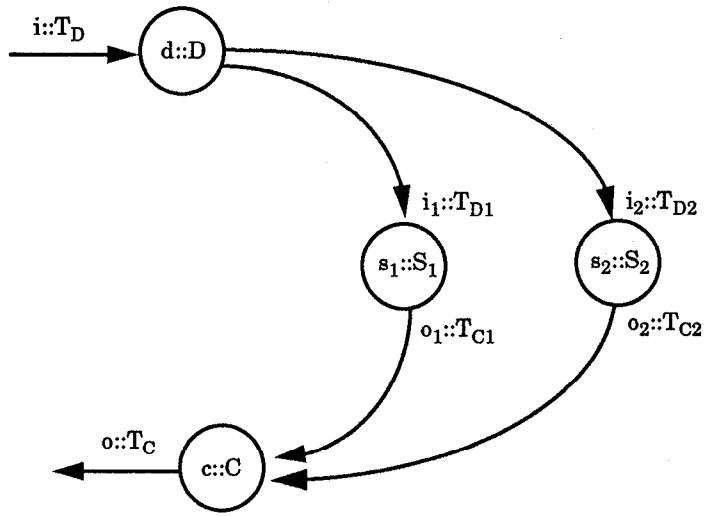
5.2.4.1 Simple distribution and collation groups

A simple distribution and collation group, based on the distribution group of Figure 17 and the collation group of Figure 19, is illustrated in Figure 20.

---

1. This is appropriate for modelling purposes, in order to simplify the conceptual model. There is no suggestion here that, in practice, one would deliberately implement a distributor which sends different messages to each member (being dual of a collator which can accept such disparate messages)!

Figure 20: A two member distribution and collation group

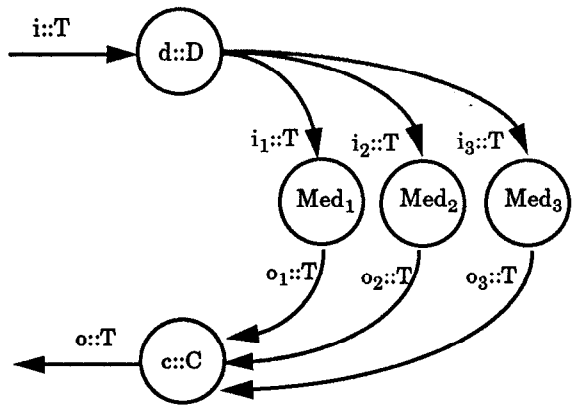


An example of a simple distribution and collation group is to be found in a simple "send three copies" protocol. The specific details of such a system are

- the constraint imposed by the distributor ensures that the messages to the members are identical to that received by the distributor,
- there are three members, each of which imposes a constraint which models the behaviour of a communications medium capable of some degree of corruption, and
- the constraint imposed by the collator ensures that the output is the majority (or other suitable collation constraint) of the messages received from the members.

For the case where all types are identical, instead of being simply conformant, the system is illustrated in Figure 21.

Figure 21: A communication medium distribution and collation group

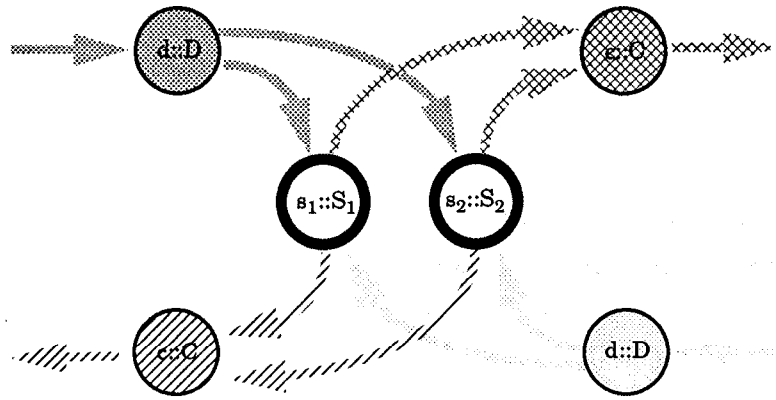


5.2.4.2 Complex distribution and collation groups

Whilst a complex distribution and collation group structurally more complex than the simple case, it is composed simply by further compounding of distribution and collation groups.

In particular, in order to make shared use of a further object, each member must communicate with that shared object. In order to achieve this, the outgoing messages (requests) must be collated before reaching the shared object; and the incoming messages (responses) must be distributed to each member. So each member is a member of two further groups: a further distribution group and a further collation group. The connectivity of this kind of system is illustrated in Figure 22. Each member forms a part of each of four groups; membership of each group by other components is illustrated by the use of filled circles with dark and light shades and also with the use of diagonal lines(Interface type detail has been deliberately omitted to emphasise the structural aspects).

Figure 22: A complex two member distribution and collation group



6 FUTURE DIRECTIONS

Given the criteria expressed above for what does constitute a group, future work can be progressed to address issues dependent on that definition, including

- group creation,
- group population change, and
- possible recovery procedures after partial failure.

A further topics for greater study include the analysis of the implications of permitting the 'same' interface to be provided as two members of a group.

7 CONCLUSIONS

This paper attempts to document the requirements which are satisfied by the engineering solutions described in [OSKIEWICZ 90]. in such a way as to permit the concept of group to extend to communication modes other than simple message passing and RPC.

In doing so it was found that the generally applicable requirements are few and provide only a weak constraint on designs. Many of the constraints imposed on designs in practice arise from the choice of fault-tolerances or local performance improvements which a specific group structure is required to satisfy.

One specific outcome of the work was the inability to derive criteria which would permit a so called 'functionally distributed group' [OSKIEWICZ 90] to be distinguished a distributed system which is not a 'functionally distributed group'. In that regard, without disputing the usefulness of such engineering solutions in some conceivable application, there appears as yet no basis for their distinction as a class of structures which meet specific testable criteria which can be guaranteed to users of such structures.

8 ACKNOWLEDGEMENTS

Thanks are due to the members of the ESPRIT ISA Project for the many useful discussion about groups, their importance, and their unimportance. Particular thanks are due to John Warne, Andrew Watson, Owen Rees, Ed Oskiewicz, Michael Olsen, and Yigal Hoffner.

9 REFERENCES

BIRMAN 87 Birman, K., "Exploiting Virtual Synchrony in Distributed Systems" in *ACM Operating System Review*, 21(5), pp. 123-138, November 1987.

HAYES 87 Hayes, Ian (ed.), *Specification Case Studies*, Prentice-Hall International, London 1987.

OLSEN 90 Olsen, M.H., Oskiewicz, & E., Warne, J.P., *A Model for Interface Groups,* APM/RC.267.00, Architecture Projects Management Ltd., Cambridge 1991.

OSKIEWICZ 90 Oskiewicz, E., Warne, J.P., & Olsen, M.H., *A Model for Interface Groups,* APM/TR.009.00, Architecture Projects Management Ltd., Cambridge 1990.

SPIVEY 89 Spivey, J.M., *The Z Notation*, Prentice-Hall International, London 1989.

TOCHER 90a Tocher, A.J., *A Mathematical Glossary*, APM/RC.068.03, Architecture Projects Management Ltd., Cambridge 1990.

TOCHER 90b Tocher, A.J., *Towards a Theory of Objects,* APM/TR.007.00, Architecture Projects Management Ltd., Cambridge 1990.

TOCHER 90c Tocher, A.J., *Graphical Object Notation,* APM/TR.008.00, Architecture Projects Management Ltd., Cambridge 1990.

A APPENDIX: Modelling entities

In interpreting the diagrams used to support the model description of the groups model, is it important to recognise that each circle (or oval) represents a potentially interacting entity within some system. A line joining two such entities represents the connection of the entities at the ends of the line, such that the possible interactions between the entities are limited to those which are possible for both entities. Furthermore, it is always the case that any interaction which does occur is known identically to both parties: communication on connections denoted by such lines is assumed to be *perfect.*

Consequently there is no notion of distance or intervening medium (both of which would introduce the possibility of corruption) associated with such connections. Instead, the model assumes that all distribution, physical or logical, is represented within the entities themselves: between distinct interfaces of the same entity; and between distinct parts of interactions within a given interface.

Therefore if one wishes to model two entities who interact with each other via an imperfect medium (which might, for example, corrupt or otherwise transform the interaction) then that medium must be included explicitly within the model as a third entity.