The interworking of traders with implicit linking

Finland

Abstract

The ODP trading function enables clients to find suitable servers in a distributed system. Servers advertise their services with service offers to trader and clients request the trader to find suitable server to their needs. Thus, trading function provides location transparency. The global trading system may be very large and it needs to be partioned. In the global system there may be several traders, which are linked together: traders are said to interwork when they are linked to share their service offers spaces. The links between traders can be seen either implicit or explicit. Explicit links are static, they are created separately by some authority. Implicit links are created automatically when traders export and import their trading service offers, following the policies of the system. When the links are seen implicit, there is no need for separate consepts or operations for the linking. The interworking of traders can cross type domain boundaries. Different domain boundaries are transparent to the traders because of the interceptors placed in the channels between interacting traders. A trader is a normal object in a distributed system, being able to rely on the infrastructure hiding for example the differences between different type domains

Computing Reviews Categories and Subject Descriptors:

C.2.4 [Computing Communication Networks]: Distributed Systems

General Terms:

Distributed systems, heterogenious systems

Adittional Key Words and Phrases:

Trading function, interworking of traders

1. Introduction

The ODP *trading function*, defined in [ISO94] enables the clients to find suitable servers in a distributed system. The objects can advertise their capabilities and match their needs against advertised capabilities with the trading function.

Trading function facilitates late bindings of objects. The consept of late bindings means, that an object may invoke an operation at another object, which is created later than the initiating object itself: the binding is not created statically at the time of object creation. This gives the benefit of flexible system evolution. With the help of the trading function, objects in the system are able to use new services provided by new servers without knowing their location beforehand - thus, trading function provides the location transparency.

The global trading system can be very, very large and it needs to be partioned, since no trading service operation can succesfully range over the global system. A trader may - anyway - wish to interact with other traders to expand the set of service offers it provides. Traders are said to *interwork* when they are linked to share their service offer spaces without loss of autonomy.

A *trader link* describes the knowledge one trader has of another trader. In this paper the nature of the links connecting interworking traders is seen either explicit or implicit or: explicit links are more static, set separately by some authority - implicit links are dynamic, set automatically during the system runtime, following the policies of the system. In [VBB95] the linking of traders is explicit, the links are created by the administrator of the system. In this paper the links are seen more dynamic - implicit.

The benefit of implicit links is that there is no need for any separate consepts concerning with the links. The purpose of the following presentation is to show, that the interworking of traders can be established with or without the help of explicit linking - there is no difference between these two types of links. Neither does the differences of different type domains have any affect to the interworking, the differences are transparent to the traders.

2. Trading service

An object in server role offers its services by advertising a service offer to the trading function with service export, which is an interaction with trading function [ISO95]. A *service offer* is a

description about an interface, including both the identifier and the computational interface signature type of the interface. A service offer may lead to a contractual context concerning the service provider and the service user.

Each interaction of an object belongs to a unigue interface. Also the interactions with trading function belongs to an interface; the interface in which the export (advertisement of service offer) and import (search of a service offers) interactions happens is called the *trading service interface*. This interface is provided by the *trader* object. An object initiating interactions with trading function must know the location of the trading service interface. The information about trading service interface is called the *trading service offer*. The trading service offer can lead to a contractual context concerning an object and a trader.

When an object (of any type) in a system is instantiated, it may be initially bound to some other objects, i.e. it may initially know some service offers. For example the trading service offer should be initially known by all such objects in the system, that want to use late bindings during their existence. The *initial trading service offer* is given to the object without the help of trader - the trading interface can not be found without this information. The initial trading service offer can be (implicitely) imported to the object in the instantiation of that object - it is (implicitely) imported to the object by some authority.

Also a trader might know an initial trading service offer of another trader. Thus, it is able to initiate export or import interactions at the other trader, for example to export or import trading service offers. When a trader *exports* its own trading service offer (or a trading service offer of some other trader) to another trader, the other trader is able to use and give further the service offer exported - if the (exporter) policy of the trading service offer allows it. When a trader *imports* a trading service offer from another trader, the trader is able to use the trading service offered but it not allowed to give the service offer further. This is because of the imported service offer is not put to the set of service offers of that trader - the trader importing a service offer plays the importer role, not traders

role. It is important to notice the differences in the roles a trader can have: an interworking trader can be in exporter role, in importer role or in traders role. These roles restricts the behaviour of interworking traders: no trader can import anything just to be able to export those imported service offers - the initiative for exporting is always in the exporters side. Figure 1 represents the difference between an exported and an imported service offer.

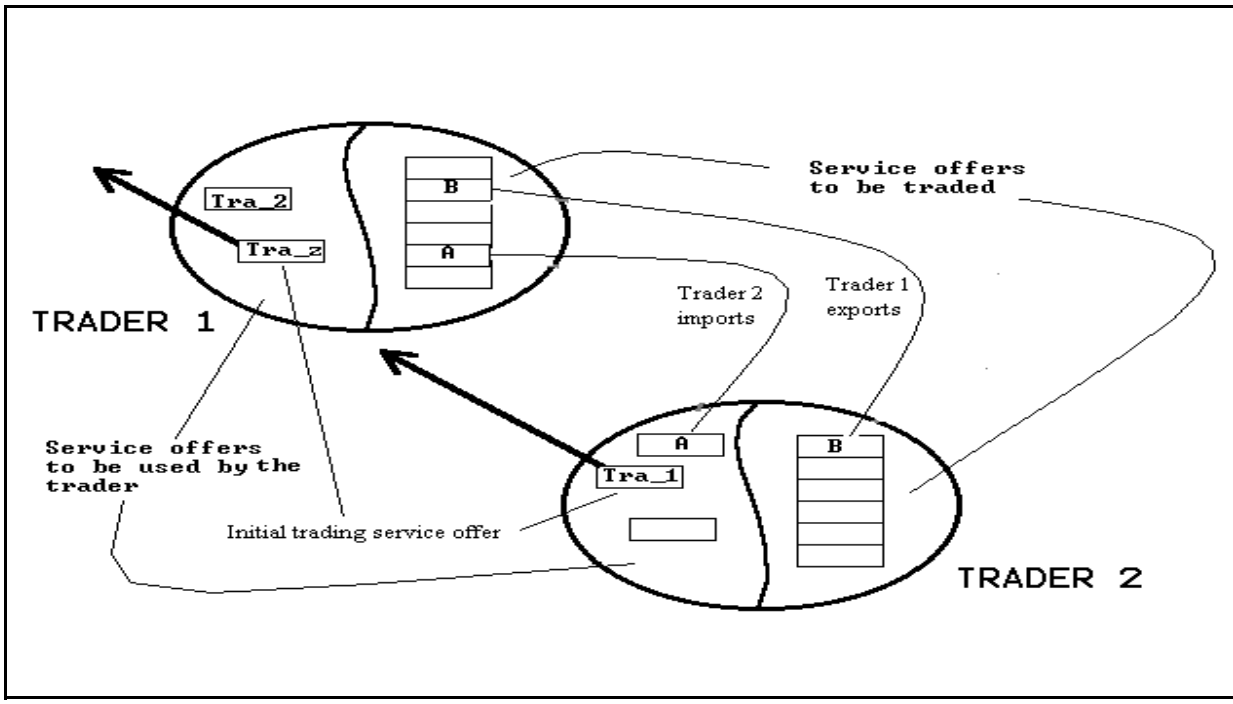

Figure 1: Trader 2 has an initial trading service offer of Trader 1. Thus, Trader 2 can import service offers from Trader 1 or export service offers to Trader 1. The imported offers are not stored to the set of service offers from which the trading is done. Trader 1, that has the service offer of Trader 2, exports service offer to Trader 2's set of service offers. This exported service offer is put to the set of service offers and can be further traded by Trader 2.

3. The initial trading graph

The *trading graph* represents the knowledge of trader information objects of other trader information objects in a group of interworking traders. The basic configuration of the interworking traders - *the initial trading graph* - can be represented with the initial trading service offers.

The administrator (or some other authority) defines the initial trading service offers to a trader which is ment to interwork. An initial trading service offer (implicitely) links a trader to be a part of the trading graph. The initial trading graph - drawn dynamically according to the initial service offers

placed in the trading system - developes when initial service offers are placed to the traders. The initial trading graph defines the area of interworking, the *service offer space* of a trading federation; a trading federation is a community of trading domains [ISO95]. In the initial trading graph there exists at least one path from all of the traders in the federation to all of the traders in the federation; anyway, the path may go through several traders. The traders connected together with the initial trading graph are able to learn about each others if the policies allow it; the whole trading graph developes evolutionary when traders export and import trading service offers, but the size of the trading graph is restricted by the initial trading graph.

In the global trading system there may exist several separate trading federations. The only way to enlarge a trading federation from the size of initial trading graph is to (implicitely) import an initial trading service offer from a trader in another trading federation. The size of the trading federation is dependent on the initial trading service offers - the initial trading graph - and this way controlled by the administrators of the trading federation. Two trading federations which are to be connected with an initial trading service offer are represented in figure 2.

4. The control of the interworking

In ODP systems the interactions between objects are controlled by a set of rules, by the *policy*, forced by the administrator. A rule for a particular purpose can be expressed as an obligation, a permission or a prohibition. The administrator can give these rules to make the system work as s/he wants.

The exporter and importer policies are defined among others in the trading function standard [ISO94]. The *exporter policy* is a policy of an exporter which restricts the set of importers to which the trader may conway the service offer. The *importer policy* is a policy of importer restricting the set of service offers considered by a trader in a service import. With these policies the exporter and the importer (basicly the administrator of exporter and importer) can control the service provided by the trading function.

The *trader security policy* restricts the set of importers and exporters that can interact with the trader. The *service offer acceptance policy* together with the *import request acceptance policy*

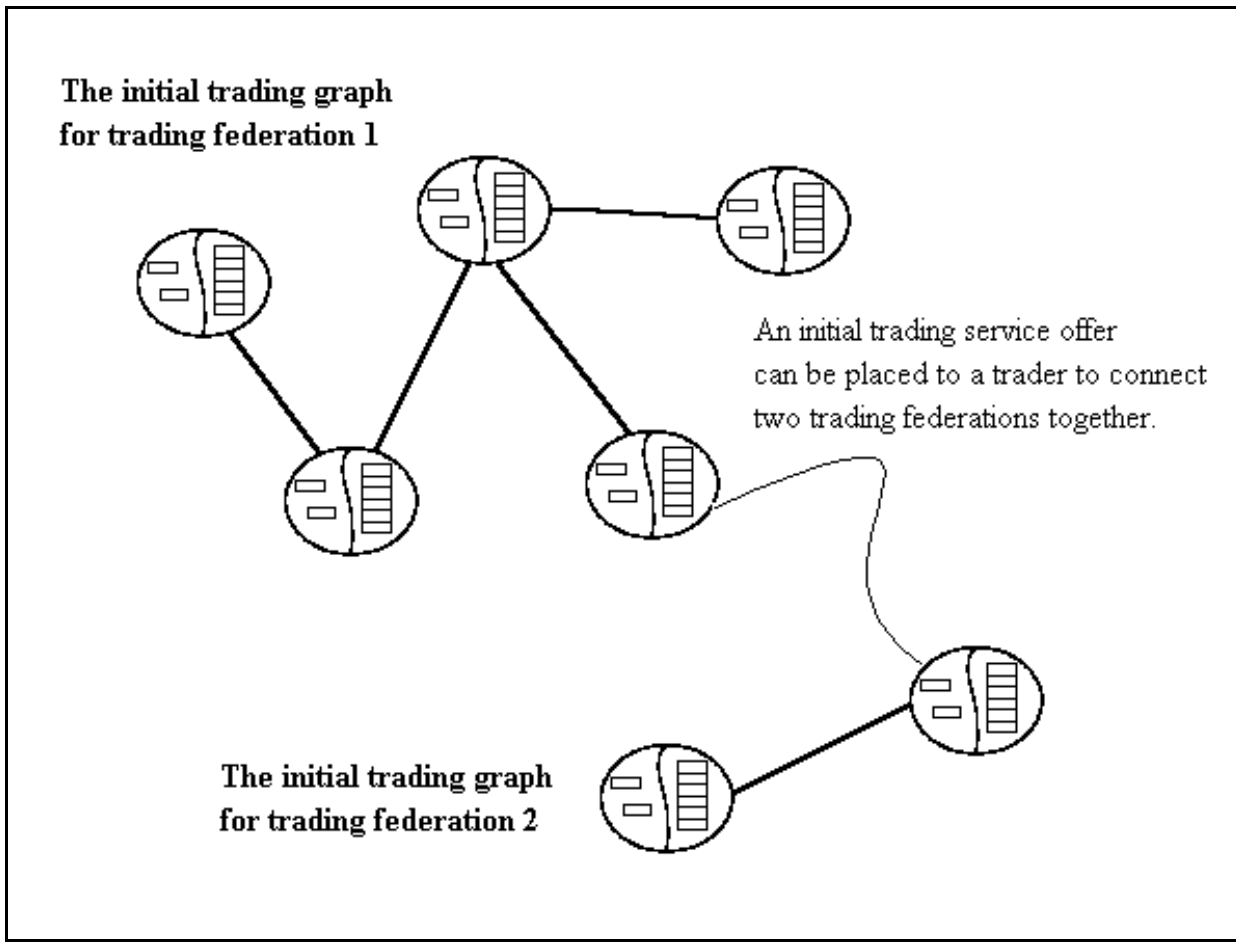

Figure 2: The size of a trading federation can be controlled with the initial trading service offers. Two trading federations can be connected together by placing an initial service offer of one trader from the other trading federation to a trader on the other.

restricts the types of service offers which are accepted to be exported or imported. With these policies the (administrator of) trading service can control the use of the trading service.

The consept of policy is the usual way to set administrative rules for any systems behaviour. No separate administrative actions on runtime is needed if the policies are defined with care. To control a system, the administrators create the *initial configuration* of objects, with trading system the *initial trading graph*, and also give the rules according to which the system can develope - the policy. *The initial trading graph with the policy statements controlling the evolution of the trader graph can be seen as a set of (implicit) trader links.* This set of (implicit) links grows evolutionary when the traders export and import trading service offers.

5. Is there any need for trader links?

The activities concerning the trading of trading service offers can be described with the help of a temporary consept of *traders' trader*. A traders' trader is a trader for trading services: when a trader wants to export a trading service offer, it consults the traders' trader and when a trader wants to get to know another trader, it also consults the traders' trader.

The communication between a trader and a traders' trader is like any communication: a trader in client role (acting as exporter or importer) to the traders' trader has somehow got the trading service offer of the traders' trading service, and thus is able to use it. The traders' trader meditiates the advertisement and discovery of trading interfaces, the service is controlled by the policies forced by an administrator - just like any other service. In a distributed system the traders' trader can be distributed - it can even be seen as a union of those parts of the functionality of the separate, interworking traders, which are dealing with the trading of the trading service offers. The centralised and distributed traders' traders are represented in figure 3.

The difference between a normal trader and the traders' trader is that the only accepted service offer type for the traders' trader is the trading service offer and the only accepted client type is the trader. This minor difference can be handeled with the policies: service offer types can be controlled with the service offer acceptance policy and the client type can be controlled with the trader security policy. The trader exporting or importing can control the traders' trading service with exporter and importer policies. As a matter a fact, the functionality of traders' trader is similar with the normal functionality of the normal trader (with minor restrictions, which can be handeld with the policies). Thus, the traders' trader can be seen eqvivalent to a normal trader.

Explicit trader links as a separate consept leads however to separate consepts for a system in which the trading service offers are treated separately. If we define separate consepts for the trading of trading services, we also define a separate consept for the trading trading service. But, because the traders' trader is equivalent to a normal trader, *no separate consepts for traders' trader is needed, neither any separate consepts for (explicit) trader links.*

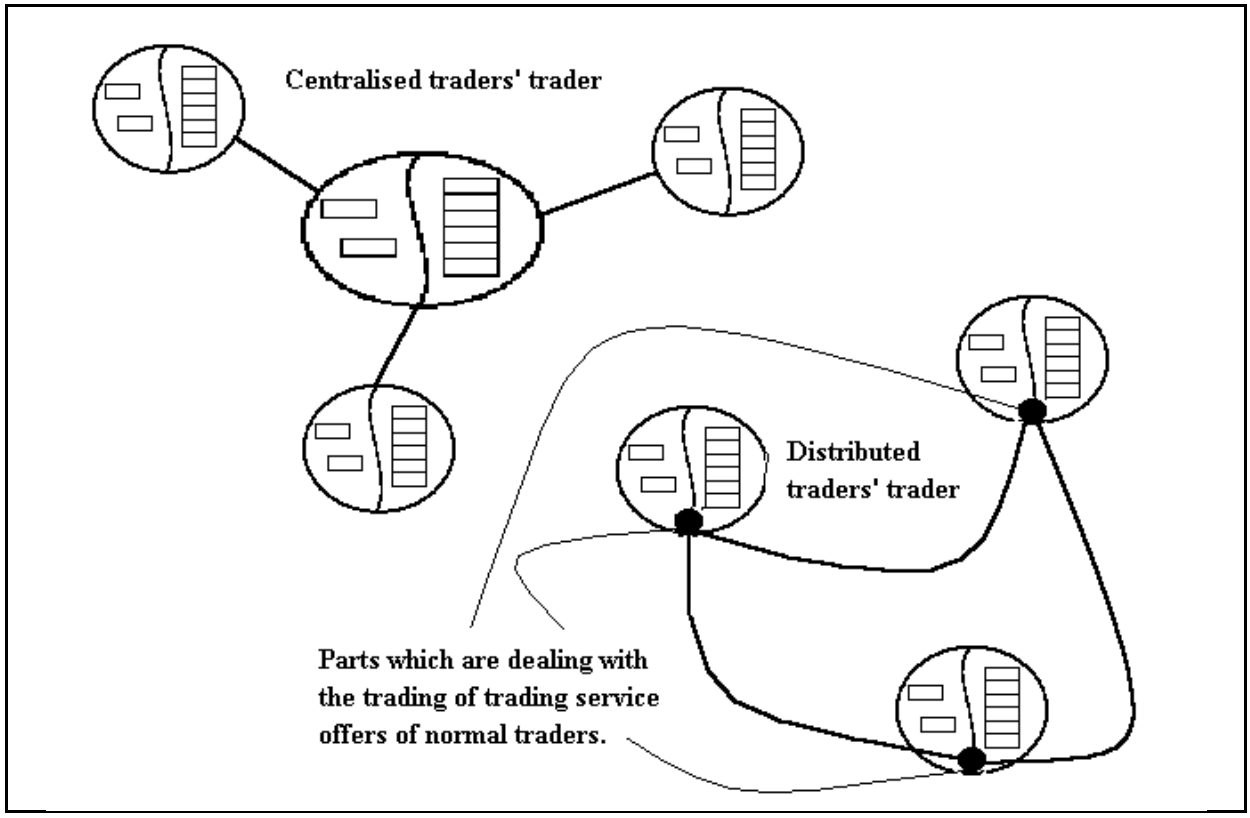

Figure 3: The trading of trading services can be done with the traders' trader. The traders' trader take care of the trading of trading service offers. The traders' trader can be either centralised or distributed.

6. Crossing the domains

Two interworking traders can be located in different type domains. The type domain boundary is normally transparent to the computational level specifications because of the ODP infrastructure or because of the consept of channel in ODP infrastructure.

A channel is a configuration of stubs, binders, protocol objects and interceptors. An *<x> interceptor* is an object in a channel, placed at a boundary between <x> domains. An interceptor performs checks to enforce or monitor policies on permitted interactions, and performs transformations to mask differences in interpretation of data by basic engineering objects (i.e. computational objects) in different domains.

A *type interceptor* masks the differences in interpretation of types in different type domains; a *type repository* manages a repository of type specifications and type relationships in a domain. The type interceptor consults the type repository of both of the type domains, and gets the knowledge how to transform the data from one type domains representation to other type domains representation, if possible. If the transformation is not possible, the interceptor can for example give an infrastructure failure to the initial party of the communication.

All the objects communicating over a type domain boundary can rely on the interceptors performing the transformations for different representations of data, also the trader. The computational object sees only a channel to another computational object and it can imagine that the other object understands the same language as the computational object itself. All the data in a type domain is represented to the computational objects in the types of the domain. Thus, it is possible to create systems, which do not need to care of the differences between different domains - the computational objects can rely on the infrastructure, here the interceptors, being able to transform data between different domains: the data representation differences are transparent.

Also the trader can rely in its communication on the transparency of different representations of types - *a trader is just an object in a distributed system, and it can communicate just like any other object - relying in an infrastructure* (the infrastructure for a trader does not necesserily have the functionalities of the trading function however).

7. An example about interworking

As an example a trader, here trader A, in type domain A wants to get to know another trader in another domain and export its trading service offer to that trader. The configuration of this example is represented in figure 4.

Trader A knows the service offer of trader B in type domain B by some means (it might be imported from another trader, exported by an other trader or got as an initial trading service offer). Trader A imports a trading service offer (type *trading_t_A* in domain A) from trader B.

The communication between type domains A and B is done by creating a channel from trader A to trader B - if the *trader security policy* allows it. This channel has an interceptor which transforms

the form of the request from type *trading_t_A* to type *trading_t_B*. The transformation is done using the type repositorys help.

The trader B searches for service offers of type *trading_t_B* from its repository - if the *import request acceptance policy* allows the trader A to request for trading service offers in trader B. Trader B finds a service offer of trader C, which is located in type domain C. Trader B gives the service offer to trader A - if the *exporter policy* of the found service offer and the *importer policy* of the service import request allows it. The *domain boundary crossing policy* is also checked.

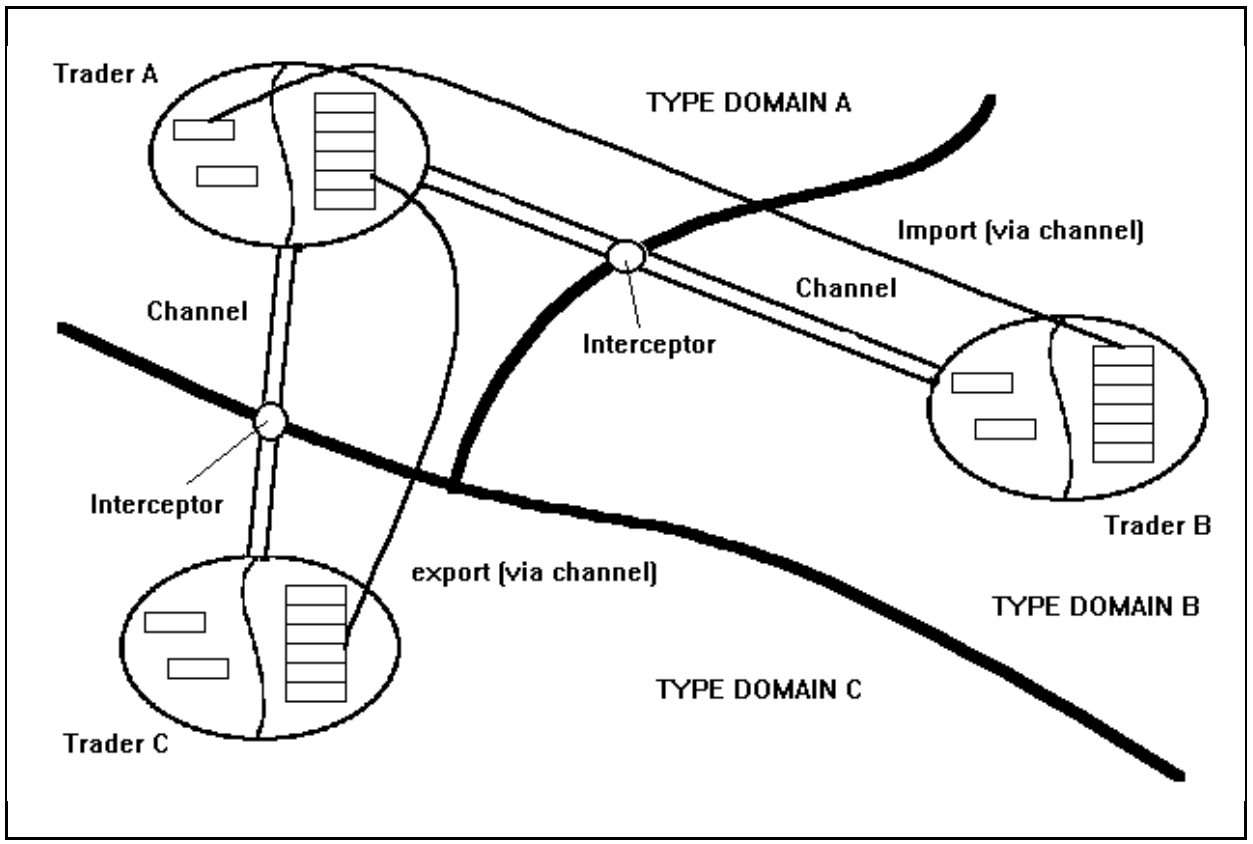

Figure 4: The configuration of the example. Trader A imports trading service offer of Trader C from Trader B. After that Trader A exports its own trading service offer to the Trader C.

The interceptor between domains A and B transforms the offer C, which is initially of type *trading_t_B* to type *trading_t_A*. The service offer of trading service C is now known, at the local types of type domain A (here *trading_t_A*) by the trader A. Trader A can not, however, further export the service offer of trader C, becouse the service offer was reached with an import initiated by trader A acting as importer, not as a trader. Trader A (acting *as importer*) can only use the trading service offered by the trader C.

Now trader A wants export its own trading service offer to the trader C. It creates a channel to the trader C with an interceptor being able to transform type descriptions between domains A and C. If the *trader security policy* and the *service offer acceptance policy* of trader C allows, trader A exports its trading service offer to trader C. Trader C (which is acting *as trader* when trader A is acting as importer) can now further trade the trading service offer of trader A - if trader A allows it with the *exporter policy* of the trading service offer.

With the help of the policies the administrators (of traders/exporters/importers) can control the interworking of traders. With the help of the interceptors a trader does not need to concider the differences on the different representation of data in different domains. The differences between type domains should be concidered in the infrastructure on which the trader can rely, not in the trading function: *the separation of different functionalities, f.ex. trading function and type transformations, must be clear when standardizating the functionalities.*

8. Conclusions

There is no need for a separate consepts or separate operations for trader links. The information stored to a trader link is, as a matter a fact, the information stored to a trading service offer added with the rules which control the use of trading function - the policy.

All the objects, also traders, in a domain might have initial trading service offers to be able to use trading service. The initial trading service offer makes it possible to connect to the trading interface and initiate interactions with trader when the object is initiated.

A trader can be considered as a normal object in a distributed system with no specialities - it can have different roles; as exporter it can export service offers to other traders, as importer it can import service offers, and as trader it can serve different clients - even traders acting in exporters or importers role. A trader must know the service offer of another trader to be able to initiate interactions with that other trader. The trading service offer known by a trader can be either an initial trading service offer or got with a trading interaction from another trader.

The size of the trading federation depends on the initial trading service offers known by the traders. The initial trading service offers defines the initial trading graph. The trading graph developes

evolutionary when traders export and import trading service offers - according to the policies of the system. Also the initial trading graph developes evolutionary when new initial service offers are introduced to traders.

The interactions between different type domains should be type representation transparent to trading function. This transparency can be reached with the help of interceptors and type repository function. A trader can rely on the distributed infrastructure, as any object, but without the services provided by trading function.

Acknowledgments

This paper has been written partly based on the discussions with the personnel in the *DRYAD project* and partly with the personnel in ODP standardization group at the University of Helsinki. DRYAD project has produced an experimental implementation of trading function including federation features.

References:

[ISO95] ISO/IEC JTC1/SC21 10746-2, International Standard for Open Distributed Processing Reference Model, Part 3: Architecture. Draft of 8 Feb 95.

[ISO94] ISO/IEC JTC1/SC21 N-9122, Open Distributed Porcessing - ODP Trading Function, WG7 Committee Draft, Southampton, July 1994.

[VBB95] Vogel, Bearman, Beitz, Enabling Interworking of Traders, Proceedings of ICODB 95. Brisbane Feb 1995.