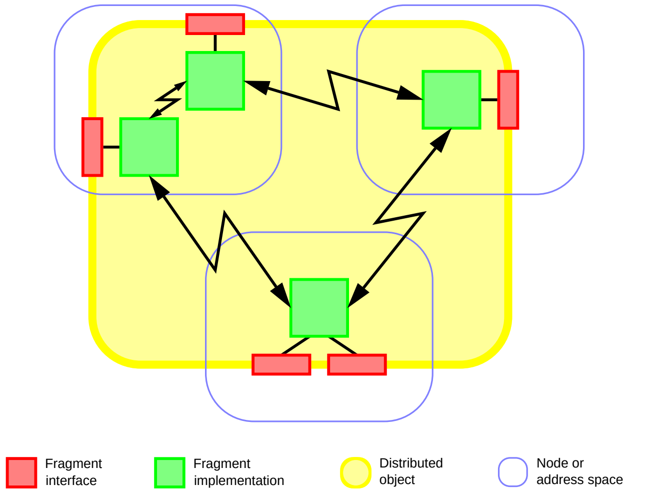
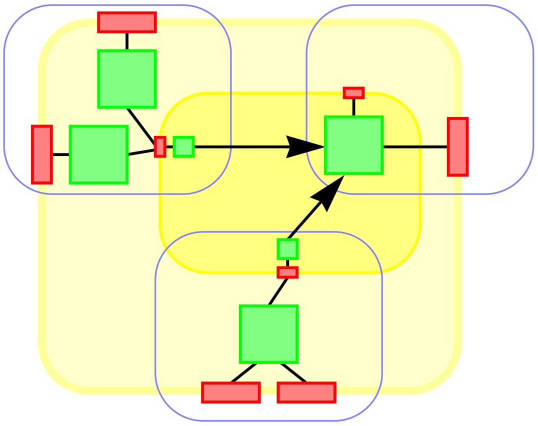
AspectIX – A Middleware for Aspect-Oriented Programming

Distributed Object Model



AspectIX distributed objects are fragmented over participating nodes or address spaces. The semantics of a distributed object are implemented by the entirety of its fragments. Therefore fragments communicate with one another. To interact with a distributed object, a client has a local fragment in its address space.
A fragment is separated into fragment interface and fragment implementation. Fragment interfaces hide the exchange of fragment implementations at run-time.

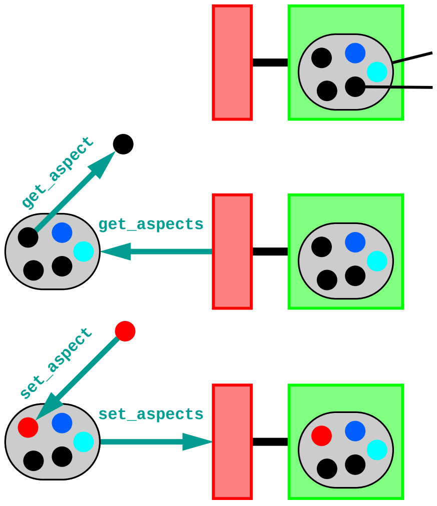
Fragment Cooperation



Fragments can recursively use fragmented objects to implement communication abstractions.
AspectIX distributed objects can be CORBA compliant, as the CORBA client/server communication pattern can be modelled by one fragment implementing server-object semantics and other fragments acting as stubs, forwarding method calls via IIOP to the server fragment.

Aspect Configuration

AspectIX supports non-functional aspects such as reliability, realtime requirements and consistency.
Each fragment implementation contains a set of aspect configurations.
Aspects can be active or inactive. Every aspect can be individually configured via its specific configuration object.



Set of aspect configurations

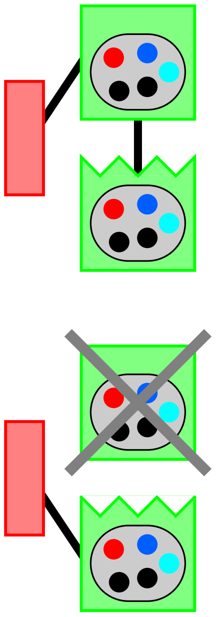
Specific configuration object

A client can obtain the set of aspect configurations of the fragment via `get_aspects`. A specific configuration object can be retrieved via `get_aspect`.

After activating or deactivating an aspect or after changing its configuration, a client can return this configuration to the set of aspect configurations via `set_aspect`.
A whole set of aspect configurations can be enabled atomically via the method `set_aspects`.

Fragment Exchange

When a new aspect configuration is enabled, the fragment implementation checks whether it can fulfill the new requirements.
A fragment implementation that is unable to adapt to the new conditions can replace itself with a new implementation.



A fragment implementation replaces itself by querying a fragment repository for a new implementation that fits the needs.
When an adequate fragment implementation is found and loaded, the current state of the fragment implementation and the set of aspect configurations are transferred to it.

When the state transfer is completed, the final action of the old fragment implementation is to instruct the fragment interface to refer to the new fragment implementation.

University of Erlangen-Nürnberg
Department of Computer Science

Bayerische Forschungsstiftung

3SOFT

SIEMENS
ATD TD-IT

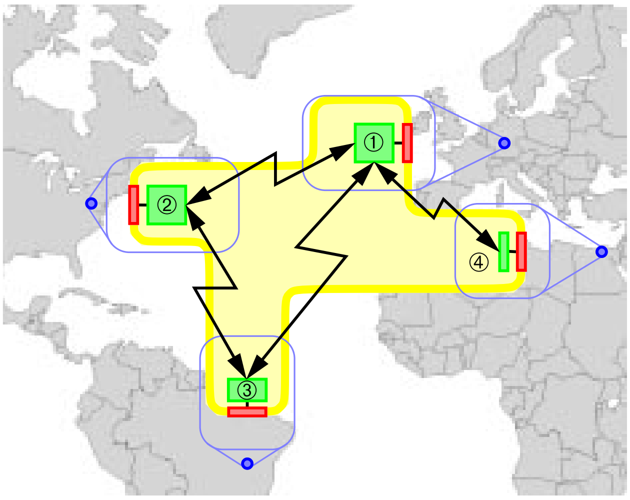
Sun

AspectIX-Team, 98-ECOOP-Poster.fm, 04.09.98

AspectIX – Solutions for Selected Application Classes

Wide-Area Distributed Systems

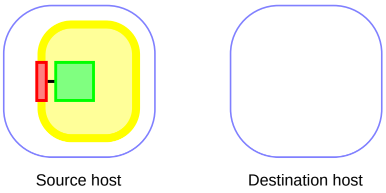
AspectIX can improve performance and availability of wide-area distributed systems by using replication combined with different consistency models.



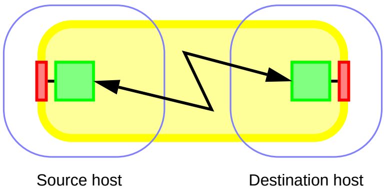
In a single distributed object, local fragments implementing strictly consistent replicas, ① and ②, can coexist with replicas having a looser consistency model like caches ③ or simple stubs ④. Fragments and consistency properties can change dynamically and transparently at run-time.

Mobile Agents

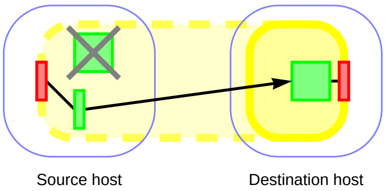
One requirement for the support of mobile agents is the implementation of mobility. The *AspectIX* architecture supports mobility by extending and shrinking the distributed object with fragments of identical semantics.



A mobile agent is modelled as a distributed object, where one fragment implements the whole agent semantics.



The distributed object is extended by a fragment with identical semantics on the destination host. Then the relevant state is transferred from the source to the destination host, so that the new fragment can act as the old one.

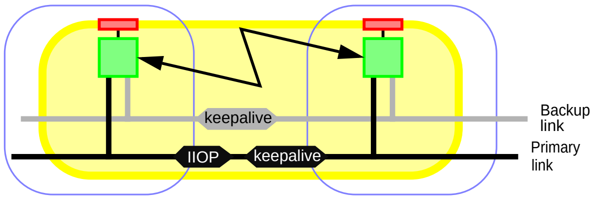


If communication with the mobile agent is still required on the source host, a stub fragment is left behind that just acts as a forwarding entity. Otherwise the distributed object is shrunk by deleting the fragments on the source host.

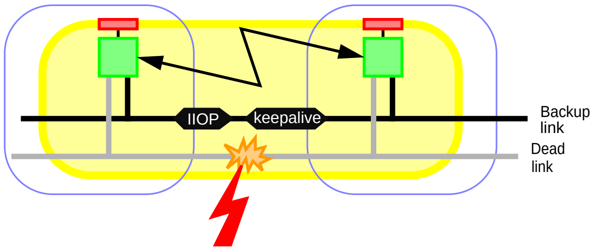
Process Control Systems

Process control systems are subject to rigid requirements in respect of reliability and availability. Often they also have to fulfill realtime constraints. The *AspectIX* architecture supports solutions for all these demands.

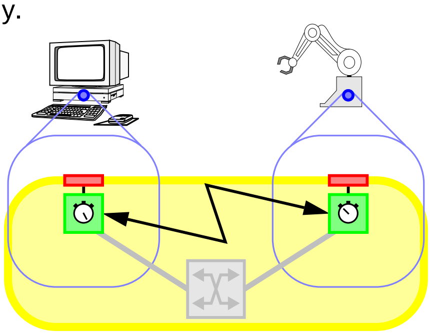
Fragment implementations can use redundant network links for their communication. In normal operation, all communication occurs over the primary link. Keepalives are used to monitor the integrity of all links.



If the absence of keepalives indicates that the primary link is dead, the fragment implementations transparently switch to the backup link.



Realtime requirements are defined as a set of parameters such as maximum execution time, period and jitter. These parameters are stored in an *AspectIX* specific configuration object and can be changed dynamically.



The fragment implementation deals with specific mechanisms like resource reservation and watchdog timing. Inter-fragment communication can use realtime capable network technologies. When a new realtime configuration is enabled and the current fragment is not able to fulfill the requirements, a suitable fragment implementation is loaded.

University of Erlangen-Nürnberg
Department of Computer Science

Bayerische Forschungsstiftung

3SOFT

SIEMENS
ATD TD-IT

Sun