# Data Interfaces as Support for Module Migration

Kenneth J. Goldman*
Washington University in St. Louis

*Module migration is the relocation of a running process from one host to another such that the move is transparent to the modules that interact with it. This position paper suggests that data interfaces can simplify run-time support for module migration and facilitate writing migratable modules.*

**The Interface Question:** *Procedural interfaces*, supported by remote procedure call mechanisms, provide a convenient mechanism for abstraction and encapsulation. However, *data interfaces* are a viable alternative for modular distributed systems.

We have recently developed a programming model called I/O abstraction in which encapsulated modules act upon local data structures, some of which may be published for external use. Communication among modules is declared in the form of logical connections among their published data structures. Whenever a module updates published data, data transmission occurs implicitly according to the logical connections [1].

We claim that this kind of data interface offers helpful properties for the support of module migration.

**Requirements:** Module migration must not affect the observable behavior of the system. Hofmeister and Purtilo list the following requirements for the support of module migration in heterogeneous distributed systems (conference presentation of [2]):

| | |
|---|---|
| (R1) | communication across heterogeneous hosts |
| (R2) | current configuration is accessible |
| (R3) | interconnections are not compiled into modules |
| (R4) | no covert communication among modules |
| (R5) | ability to add/remove modules and bindings |
| (R6) | access to messages in transit |
| (R7) | mechanism for synchronizing activities |
| (R8) | access to module's state information |

**Data Interface Properties:** Data interfaces provide a clean a separation of computation from communication. Each module is concerned only with local computation, while communication is declared separately in terms of logical connections. The logical configuration is available to the run-time system, and can be modified independent of the software modules (satisfying R2–R5). The run-time system can use the configuration information to support communication across heterogeneous hosts (R1) and in order to control the ordering and delivery of data transmission (R6–R7).

Data interfaces expose state information and make it available to the run-time system (R8). This simplifies the module migration problem, since state extraction mechanisms are not required. The trick is to provide a clean data interface that captures only the essential I/O behavior of a module and at the same time exposes enough state information so that the module can be relocated without state extraction.

**Writing Modules that Migrate:** Once the state information is exposed, the next problem is determining when it is safe to move the module. The programmer might specify this explicitly by identifying safe points in the code, but we prefer confining the process migration code (cleanup and restart) to one section of the program and relying on existing mechanisms for specifying atomic steps to prevent untimely migration. If a module is moved between atomic steps, then its behavior will be unaffected. In [3], we offer three approaches to writing migratable modules that exploit data interface properties, localize process migration code, and use existing atomicity mechanisms.

## References

1. Kenneth J. Goldman, Michael D. Anderson, and Bala Swaminathan. The Programmers' Playground: I/O abstraction for heterogeneous distributed systems. In *Proceedings of the 27th Hawaii International Conference on System Sciences*, January 1994.
2. Christine R. Hofmeister and James M Purtilo. Dynamic reconfiguration in distributed systems: Adapting software modules for replacement. In *Proceedings of the 13th International Conference on Distributed Computing Systems, Pittsburgh, Pennsylvania*, pages 101–110, May 1993.
3. Bala Swaminathan and Kenneth J. Goldman. Dynamic reconfiguration with I/O abstraction. Technical Report WUCS-93-21, Washington University in St. Louis, August 1993.

*Supported in part by National Science Foundation grant CCR-91-10029. Washington University Department of Computer Science, St. Louis, MO 63130, kjg@cs.wustl.edu

0-8186-5390-6/94 $3.00 © 1994 IEEE