and locating service that efficiently maps high level names from a network-global structured name space to instances. Instances communicate by *remote procedure calls* in a client/agent relationship, using the results of the location service to achieve network transparency. At the application level, a network of BirliX hosts pretend as a single large system providing computing resources without regard to the multiplicity and distribution of the machines which actually provide the resources.

Fault tolerance is based on two principles, the *redundancy* provided by networks and a *checkpoint and recover* mechanism. Again, instances are the basic unit in both cases. Redundancy is supported by providing *replicated instances*, which can be checkpointed (the current state of the instance is frozen) and recovered (a frozen state is restored). These fault tolerance mechanisms are small and high-speed primitives, inherited from the primary type. These primitives provide a sound base for the implementation of high level fault tolerance policies, like e.g. a transaction management or application system checkpointing.

The BirliX security is based on the assumption, that the kernel has to be *trustworthy* to build a *securable system* on top of it. The primary type provides the necessary mechanism to build the various application security policies on top of the BirliX system. BirliX is designed in such a way that integrating new security policies is relatively easy. All policies base on the protection scheme for instances. This scheme guarantees the *encapsulation of instances* making them accessible only via their exported operations. Additionally, each type may individually protect each of its operations by a type-dependent protection strategy. The protection policy currently exported from the primary type is based on *access control lists*. Nevertheless, the entire kernel protection scheme can be changed by simply redefining the protection scheme within the primary type.

Technical reports and other documents describing BirliX can be obtained using anonymous FTP of host gmdzi.gmd.de [129.26.1.90]. The papers are located in the directoy /pub/gmd/BirliX.

References

[1] H. Härtig, W. E. Kühnhauser, O. C. Kowalski, W. Lux, W. Reck, H. Streich, and G. Goos, "The Architecture of the BirliX Operating System," in *11. ITG/GI Fachtagung, März 1990*, ITG/GI, Mar. 1990.

[2] O. C. Kowalski and H. Härtig, "Protection in the BirliX Operating System," in *Proceedings of the 10th International Conference on Distributed Computing Systems*, pp. 160–166, IEEE, May 1990.

[3] W. E. Kühnhauser, H. Härtig, O. C. Kowalski, and W. Lux, "Mechanisms for Persistence and Security in BirliX," in *Security and Persistence,Bremen 1990*, Springer-Verlag, May 1990.

[4] W. Lux, H. Härtig, and W. E. Kühnhauser, "On the Implementation of Abstract Data Types in BirliX," in *Progress in Distributed Operating Systems and Distributed Systems Management, Lecture Notes in Computer Science 433*, Springer-Verlag, Apr. 1989.

[5] W. E. Kühnhauser, "Reliable Naming Services in Distributed Environments." Microkernel Applications Workshop, OSF Research Institute, 27-29 November 1990, Workshop Verteilte Betriebssysteme, Schloss Gaussig, 15-17 Januar 1991, Nov. 1990.

[6] W. Reck and H. Härtig, "Software Configuration Management for Medium-Size Systems," in *Advanced Information Systems Engineering, Lecture Notes in Computer Science 436*, Springer-Verlag, May 1990.

BirliX
(Birlinghoven Operating System)

German National Research Center for Computer Science (GMD)

Personnel

Principal Investigators	Hermann Härtig	haertig@gmdzi.gmd.de
	Winfried Kühnhauser	kuehnhsr@gmdzi.gmd.de
Researchers	Oliver C. Kowalski	kow@gmdzi.gmd.de
	Hermann Streich	str@gmdzi.gmd.de
	Wolfgang Lux	lux@gmdzi.gmd.de
	Wolfgang Reck	reck@gmdzi.gmd.de
Students	Peter Schüller	phs@gmdzi.gmd.de
	Martin F. Gergeleit	mfg@gmdzi.gmd.de

Contact

Dr. Hermann Härtig
Gesellschaft für Mathematik und Datenverarbeitung (GMD)
Institut für Systemtechnik
Postfach 1240
5205 Sankt Augustin 1
Germany

Project Description

Rapidly growing customer requirements and technical progress in communications hardware result in a demand for open system architectures that are easily extensible and adaptable without interfering with a well designed system architecture. Therefore, the key objective of the BirliX project is to present an operating system design that integrates extensibility and flexibility as the basic part of its architecture.

Distributed environments and their impacts on fault tolerance, performance, and security build the second key objective of the BirliX project. While distributed systems give access to a large number of resources that can be used to promote fault tolerance and performance, security becomes a serious problem.

The BirliX kernel paradigm is the *data abstraction*. The system is basically an *abstract data type* (adt) *management system* with services to define and instantiate types, to identify instances and to support instance communication. The common behavior of all types — concerning distribution, security and fault tolerance — is defined by the so called *primary type* and is imported from it whenever a new type is defined. A very natural feature of an adt management system is the functional extensibility and adaptation to new requirements by simply defining new adts.

Concerning distribution, the adt paradigm provides us with the smallest unit of identification and communication, the *adt instance*. Application programs on BirliX are sets of interacting adt instances distributed among several nodes in a computer network. Instances are identified and located by a *distributed naming*