# Component Based Operating System

Patty Kostkova, Kevin Murray and Tim Wilkinson
*System Architecture Research Center*
*City University*
*London, UK*
*{patty,kam,tim}@sarc.city.ac.uk*

The microkernel concept - the leading operating systems design approach of recent years - brought both applications and kernels a level of flexibility. However, the pure micro-kernel approach has two main drawbacks: poor performance, due to extra kernel-server overheads, and the inability to tailor servers to applications demands and needs. To overcome these problems the new research is investigating better structuring abstractions which enable finer granularity in OS components. This research is primarily based around the new "extensible" operating system approach (e.g. Exokernel, SPIN, etc.) which tackles the problem of efficiency, flexibility and application support with reduction of resource abstractions and specific kernel customization via application provided kernel code.

We are designing a radically different operating system approach to cope with the requirements of efficiency, flexibility and ease of reconfiguration: it consists of independent "components" representing hardware devices or offering services. Hardware components provide a direct interface to real devices above which there is a "virtual machine" layer implementing necessary protection schemes. However, there is no enforced hardware abstraction which ensures the desired efficiency. The next layer consists of running applications and the common servers, such as file systems, distributed shared memory, etc. Applications then have the choice of accessing a raw device directly or using an abstraction provided by these servers.

The information about all components currently available in the system is provided by a special Manager component. It administrates a shared tuplespace and is responsible for resolving and negotiating application requests (which come in the form of tuples placed into the tuplespace) so as to find the most convenient component matches. Once all desired components for an application are selected and plugged together by the Manager, the application can communicate with these components using indirect calls. At this stage, trusted applications can access services through indirect procedure calls giving them shared library performance rates. Untrusted applications can still use these abstractions, but will pay a performance penalty because of protection checks. This gives the applications high speed, secure, safe access to abstractions and hardware resources in a flexible and manageable fashion.

Here we present a component based OS which consists of independent hardware components and abstraction services which are plugged together in response to application demands. The system supports flexibility through late binding, dynamic modification of its parts during execution, efficiency through reduction of unnecessary abstractions and by binding trusted applications to servers by indirect calls. In addition to its structure and flexibility, the system can support 24x7 operation by utilizing its ability to exchange components dynamically and through the potential for the Manager to replace failed components with alternatives providing an equivalent service level.