GRIDKit: A COMPONENT-BASED MIDDLEWARE FRAMEWORK FOR CONFIGURABLE AND RECONFIGURABLE GRID COMPUTING

Overview of GRIDKit

• Our approach to addressing the limitations of Grid Middleware is subsumed under the umbrella of the GRIDKit Architecture.
• The aim of GRIDKit is to provide underlying support in four domains for the provision of Grid Services.

Open Overlays

• Application Layer Overlay Networks allow new network services to be deployed without requiring changes to network level infrastructure.
• Members of an overlay network (hosts, servers, routers, or applications) organise themselves to form a logical network topology.
• Communication is between neighbours in the topology.
• A member of an overlay network sends and receives application data, forwarding messages intended for other members.
• Examples of overlays are peer-to-peer distributed hash tables, multicast trees, network flooding etc.
• Sophisticated communication services e.g. data streaming, P2P resource discovery and application-level multicast are supported by appropriate overlay configurations. Multiple service bindings can then operate over their selected overlay.

The GRIDKit Architecture

• GRIDKit is an instantiation of the generic OpenORB middleware platform and follows the philosophy of building systems using components (with the OpenCOM component model), component frameworks and reflection.

The diagram above demonstrates how we envisage “layers” of overlay networks. That is, overlay networks are layered on top of one another to achieve the required binding semantics. Hence, the same node participates in multiple overlays, and new bindings can be layered over existing overlays (supporting re-use).
• For example, in one set up a group binding can be underpinned by a multicast tree over a key based overlay. For example, SCRIBE layered over PASTRY.
• The same multicast tree can then be layered over IP at a different time.
• Our architecture intends the overlay to be controlled by a component framework which manages the composition of multiple overlays.
• The diagram below gives an overview of the components plugged into the Overlay framework, which provides open access to configure and dynamically reconfigure the node’s participation in multiple overlay networks.
• Each overlay is composed of three components: State (network), Control (algorithm to maintain topology), and Forwarding (routing algorithm).
• The framework then maintains rules for valid configurations.

The diagram above demonstrates how we envisage “layers” of overlay networks. That is, overlay networks are layered on top of one another to achieve the required binding semantics. Hence, the same node participates in multiple overlays, and new bindings can be layered over existing overlays (supporting re-use).
• For example, in one set up a group binding can be underpinned by a multicast tree over a key based overlay. For example, SCRIBE layered over PASTRY.
• The same multicast tree can then be layered over IP at a different time.
• Our architecture intends the overlay to be controlled by a component framework which manages the composition of multiple overlays.
• The diagram below gives an overview of the components plugged into the Overlay framework, which provides open access to configure and dynamically reconfigure the node’s participation in multiple overlay networks.
• Each overlay is composed of three components: State (network), Control (algorithm to maintain topology), and Forwarding (routing algorithm).
• The framework then maintains rules for valid configurations.

Service Binding

This area provides sophisticated communication services beyond SOAP: i.e., support for QoS management, and for different interaction types e.g. (publish-subscribe, group communication, peer-to-peer, data sharing and others). Hence, a range of interaction types are abstracted using a web services API, thus providing the basis of an extensibility framework in which to accommodate the more complex range of interaction types.

Resource Discovery

This provides service, and more generally, resource discovery services, allowing for the use of multiple discovery technologies so as to maximise the flexibility available to applications. Examples of alternative technologies are SLP or UPnP for more traditional service discovery, GRAM for CPU discovery in a Grid context, and P2P protocols for more general resource discovery.

Resource Management

This comprises both the coarse-grained distributed resource management that is currently provided by services such as GRAM, and the fine-grained local resource management (e.g. of channels, threads, buffers etc) that is required to build end-to-end QoS.

Grid Security

This supports secure communication (authentication and confidentiality) between participating nodes orthogonally to the interaction types in use.

Overlay Networks

An extensible family of open and programmable overlay networks underpins the overall GRIDKit architecture.

The role of the Overlay Network is to route packets through virtual networks that are tailored to support the various service interaction types, thus providing an approach that is network-centric, offers a strong architecture for the system infrastructure, and facilitates self-management through the inherent openness of the network.