

The Need for Network-Centric Programming Models

Konstantin Popov, Per Brand
(SICS)

Vlad Vlassov, Seif Haridi
(KTH/IMIT)

Important Grid Characteristics

- Scale
 - wide availability of computers and other resources
 - resources are networked
- Dynamism
 - resources and networking facilities come, change and go away
- Distribution
 - maintaining consistency becomes harder due to latency

Grid Services Today

- Inspired by the object-oriented metaphor
- Client-server model for service access

Grid Services, Middleware and OS

- Traditionally, middleware provides high-level abstractions for certain aspects of distributed programming
 - communication, synchronization, reliability, scalability and heterogeneity
- OS provide functionally vital for the middleware
- Performance-crucial features are implemented directly by OS

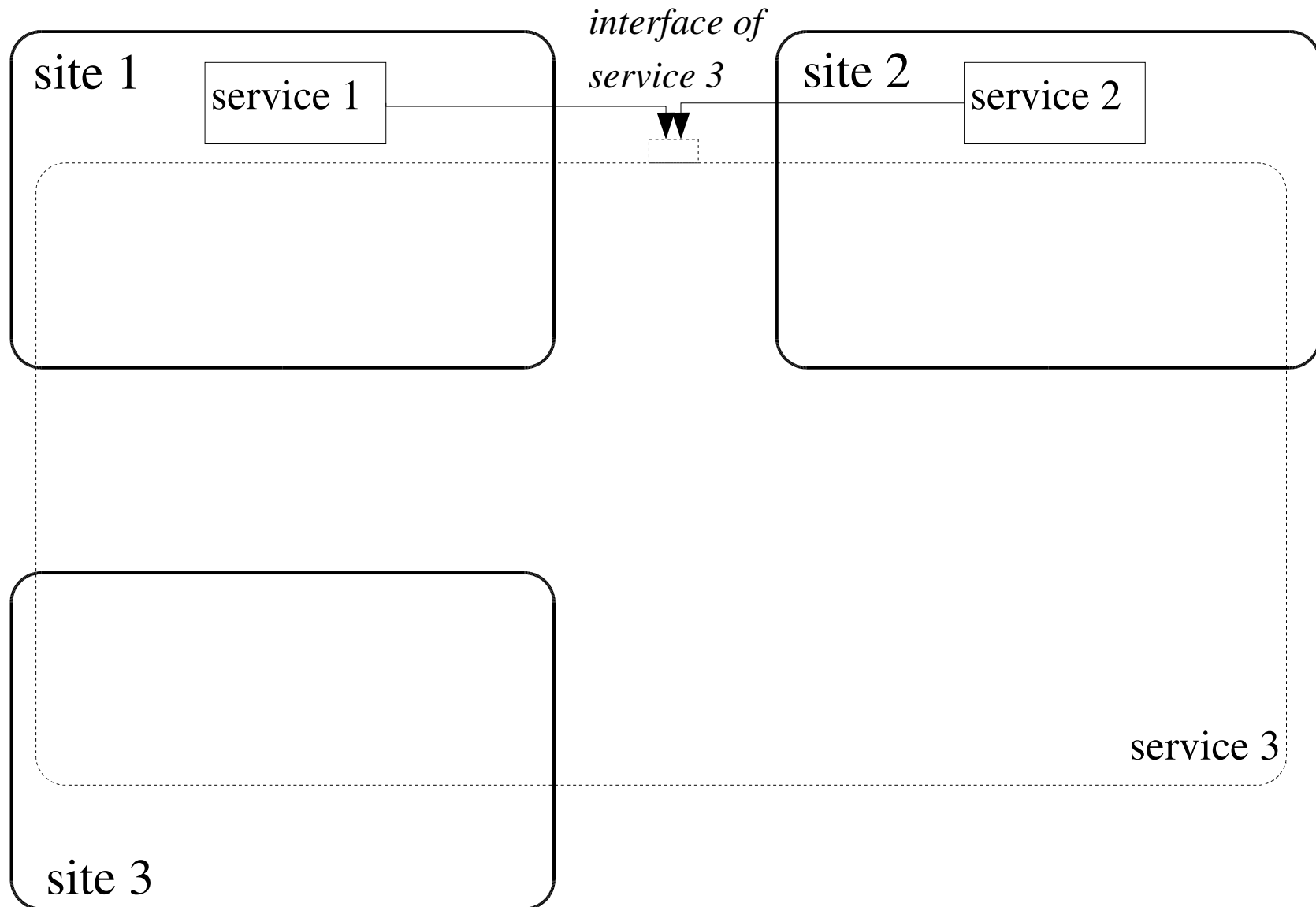
Programming Models for Designing Grid Services

- Client-server for interaction between services
 - OGSI/WSRF, GridRPC
- Shared memory, message-passing, and recently - P2P abstractions are studied for Grid
 - considered when a compute- or storage- intensive, but not a “client-intensive” service is to be implemented
 - shared memory and message-passing don't really deal with node and network failures
- Component-based frameworks for handling of dynamic resources (e.g. derived from CCA)

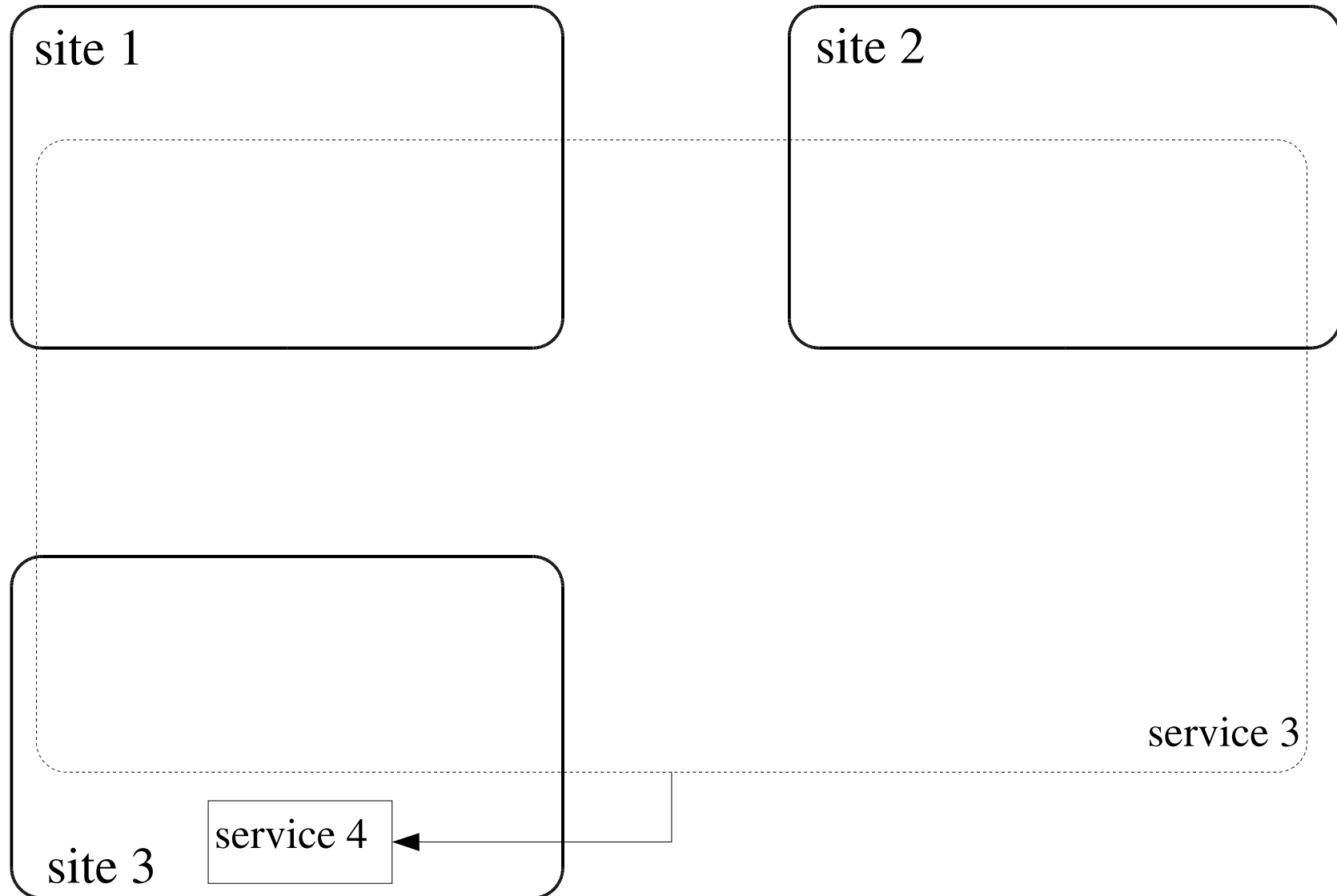
Large-Scale Grid Services?

- Concurrently service many clients which are physically distributed and/or mobile
- Necessarily utilize many auxiliary services and are distributed over Grid

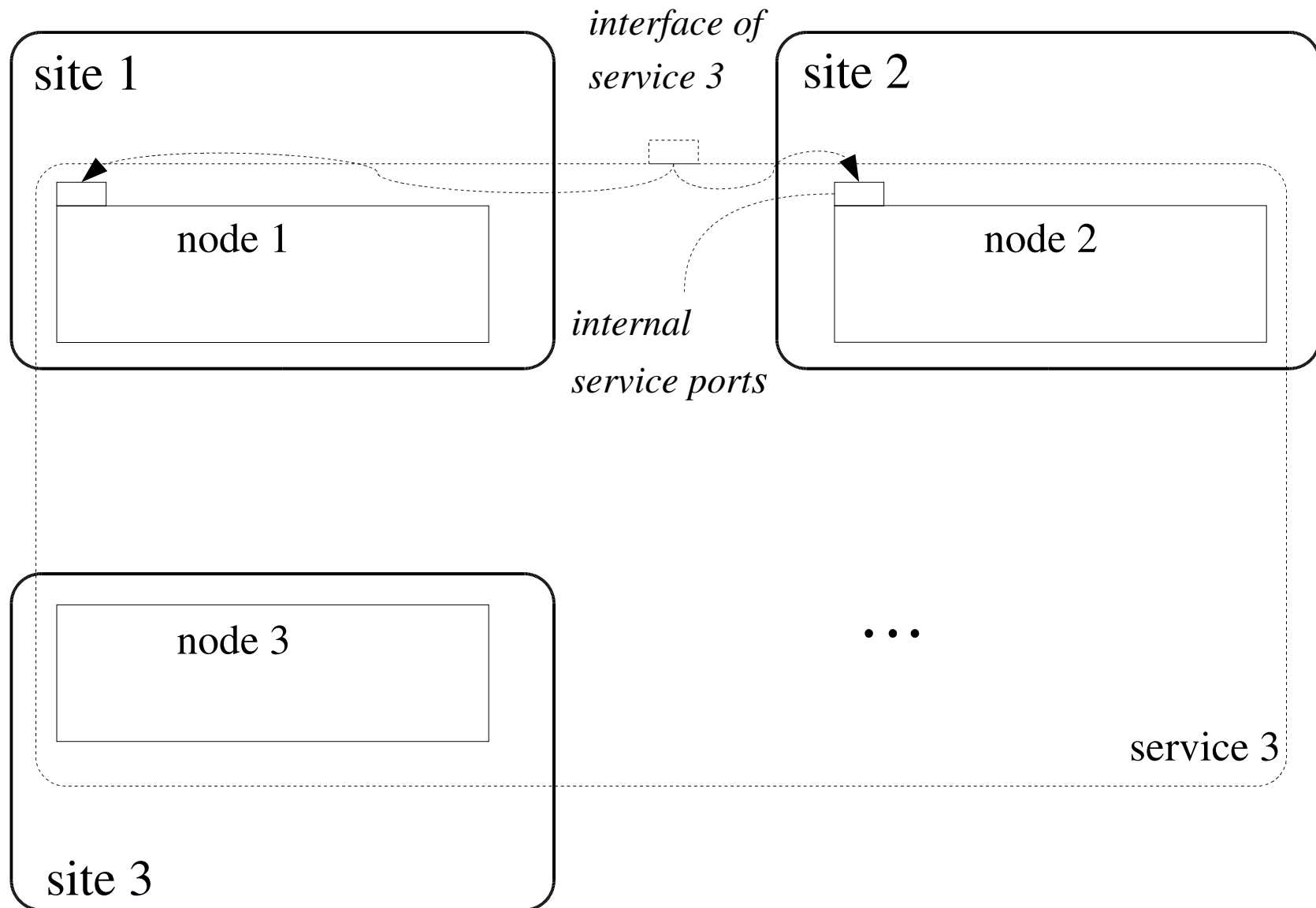
Distributed Grid Services: Interface



... uses other services



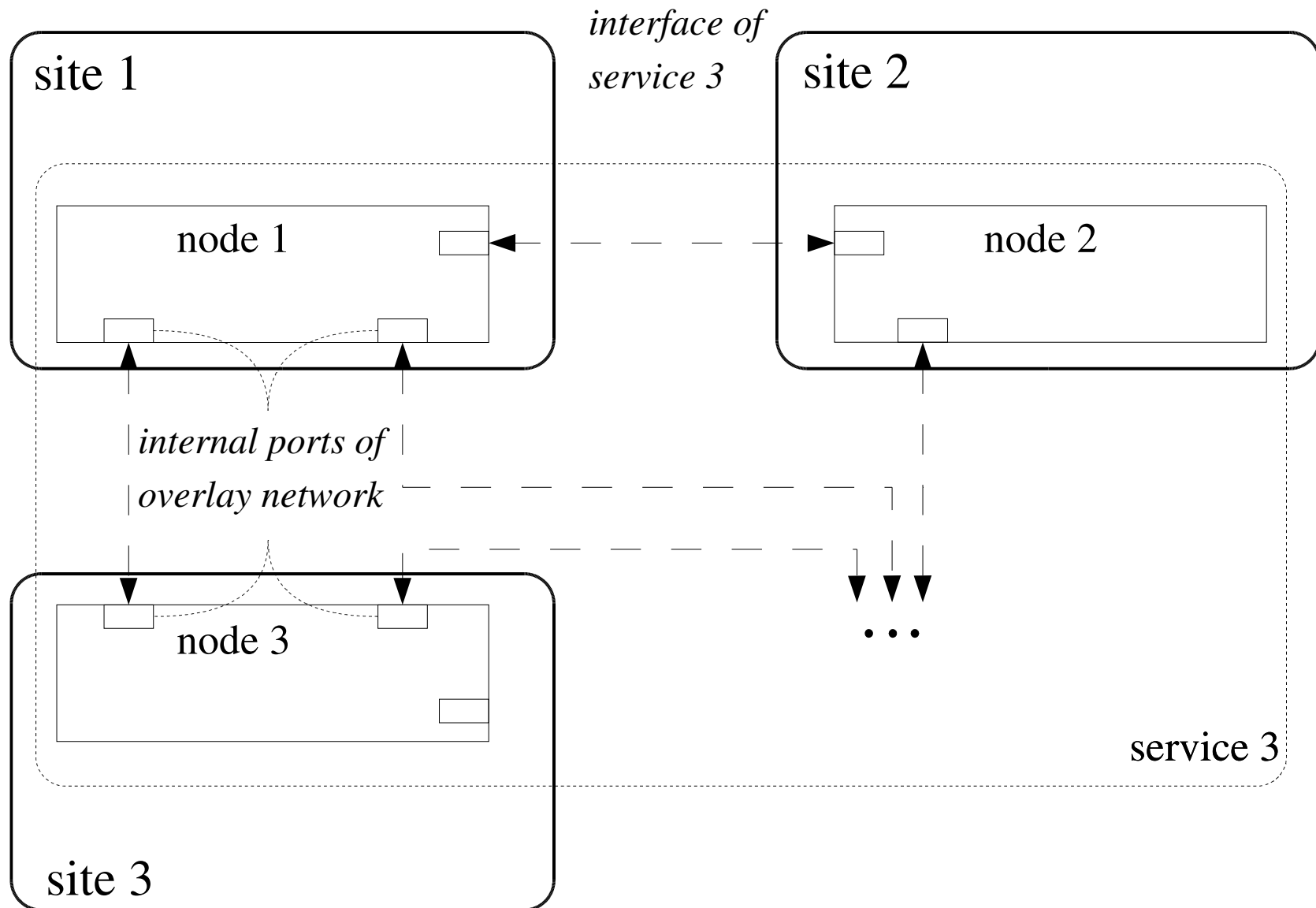
... is composed from *nodes*



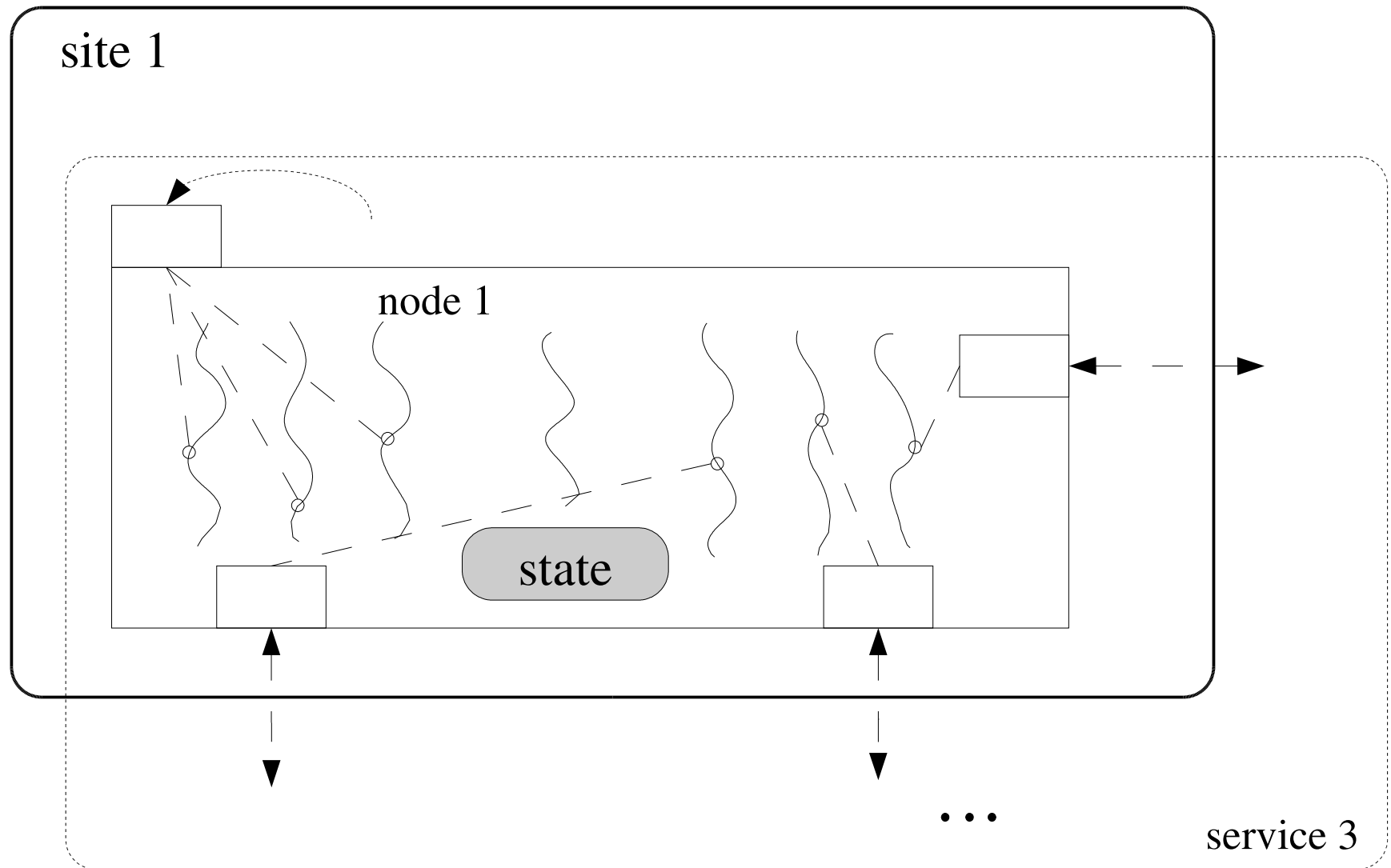
Important Catalyst for Scalable Grid Services: Overlay Networks and DHTs

- Abstracts physical network addresses (overlays) and data locations (DHTs) from the programmer
 - Theoretically, separates the scalability issue from the application's functionality
- Already existing implementations show good scalability and self-management properties

... nodes are interconnected by an overlay



... concurrent threads handling communication with clients & peer nodes



A Programming Model for Distributed Grid Services ?

- Service definition
- Programming a service
- Using a service

Service Definition

- Pragmatically, today's standard solutions (such as OGSI) should be conservatively extended
- Awareness- and control- facilities of a service as a *distributed* one could be added as additional ports
 - in style of e.g. the Fractal framework, or the CCA

Programming a Service

- Service composition
 - communication, synchronization, exception handling
 - non-functional aspects too!
- Service state
- Connection between external interface(s) and internal service ports

Using a Service

- Service consistency across internal ports
 - whatever the internal service port is chosen, the distributed services should be consistent according to a well-specified consistency model
- Composition of distributed services
 - what happens when individual nodes of a distributed service use another *distributed* service??

Support by Network-Centric OS

- Light-weight concurrency
 - nodes in a large-scale service contain a lot of concurrency used to model service's clients and peers
 - would eliminate the need for two-level scheduling (by the OS and within processes for light-weight threads)
- Overlay networking
 - bypassing user-space processes when routing messages
 - might demand kernel support for naming and addressing of services and service nodes
 - consider also kernel-space serialization of data

Conclusions

- Grid's scope and diversity of resources will encourage development of large-scale services
- Such services will necessarily be distributed, servicing client requests at different computers
 - Present day Grid software tools are insufficient
- A programming model has to define service composition and distributed state, and decouple the service interface from its implementation
- Network-centric OS should provide light-weight concurrency and low-overhead overlay networking