



Jini and the Grid

P. Kacsuk

Laboratory of Parallel and Distributed Systems

MTA SZTAKI Research Institute

kacsuk@sztaki.hu

www.lpds.sztaki.hu

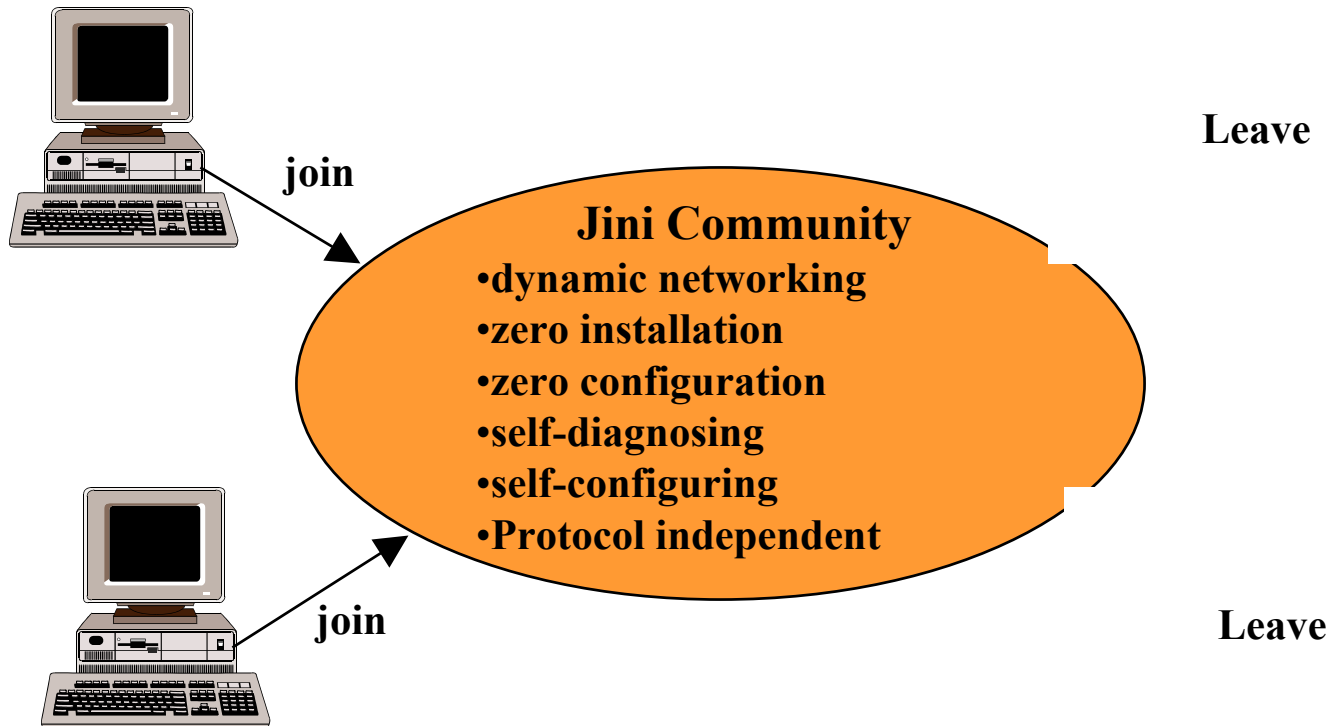


Jini: an overview

- Jini, in Arabic word meaning „magic”, is a **dynamic distributed network computing architecture** for providing networking of **services**
- Jini provides an infrastructure to enable this dynamic formation of services with
 - 0% installation
 - 0% configuration
 - 100% service interaction
- **Self-diagnosing, self-configurable** architecture
- **Protocol-independent** architecture, it can interact with any distr. Object using any protocol (RMI, CORBA, DCOM, etc.)



Jini: a dynamic network



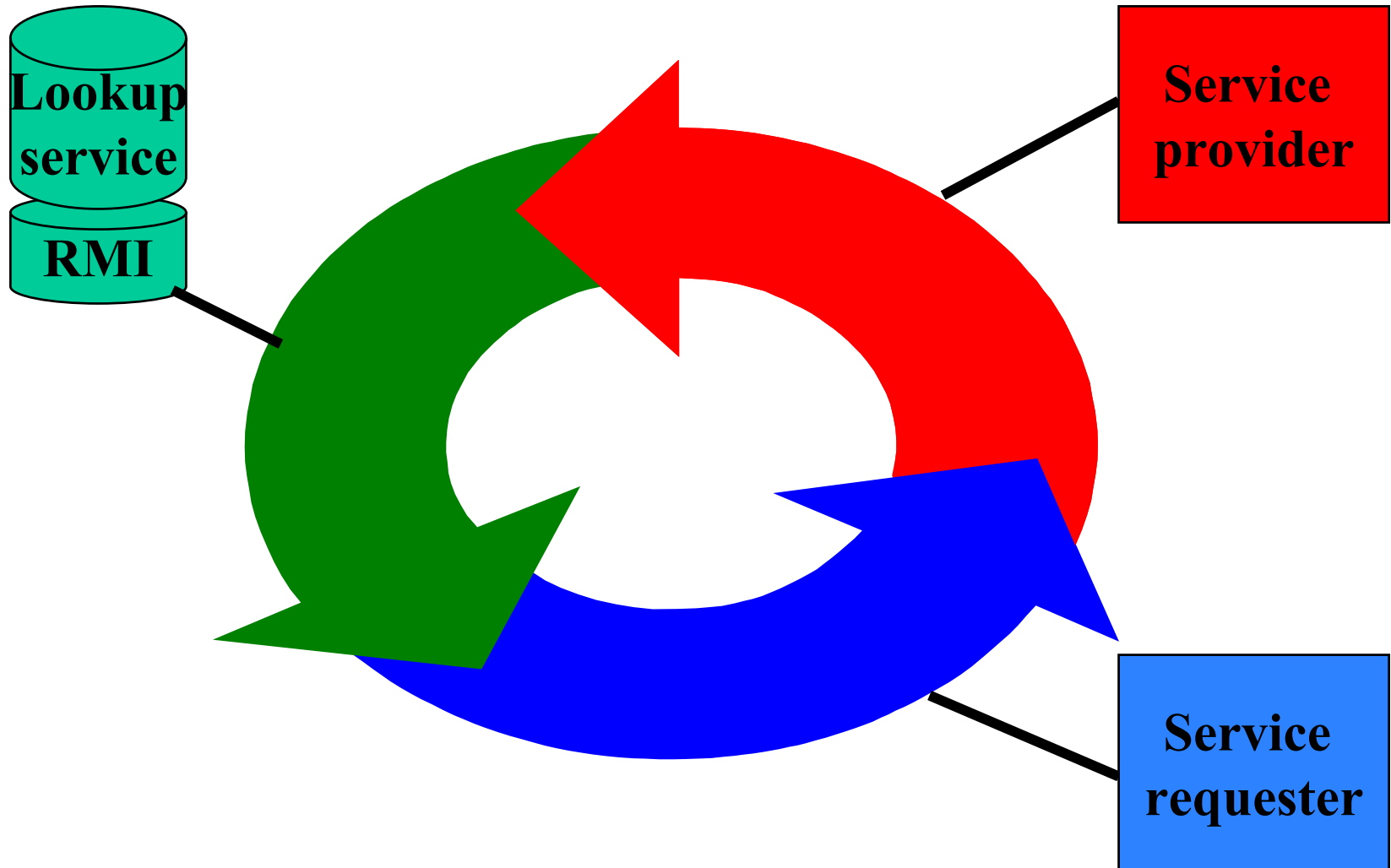


Goals for Jini

- To provide an infrastructure to connect
 - anything
 - anytime
 - anywhere
- To provide an infrastructure to enable „network plug and play”
 - 0% installation
 - 0% configuration
- To support a service-based architecture by abstracting the hardware/software distinction
- To provide an architecture to handle partial failure



Three basic elements of Jini



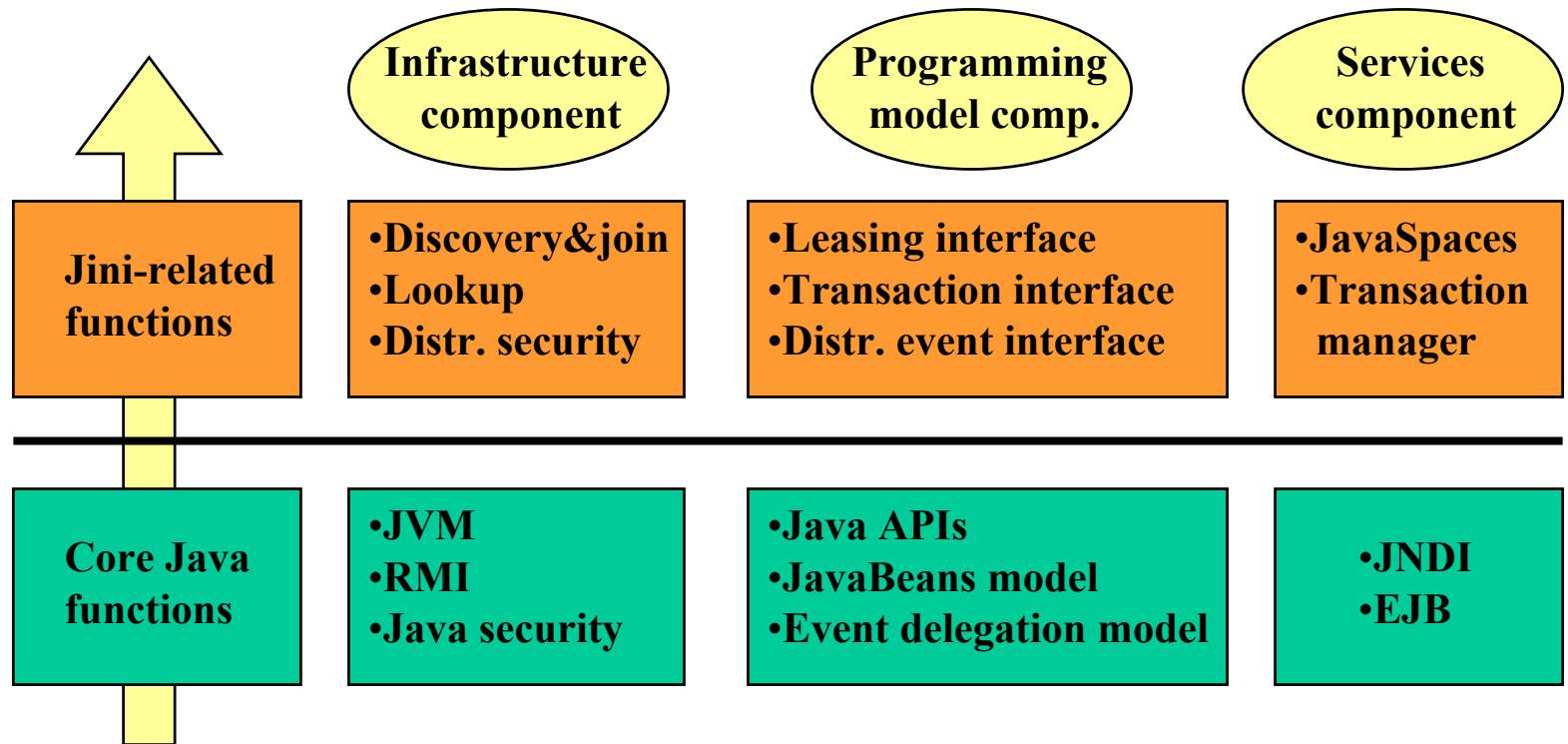


System Assumptions

- The existence of a network with reasonable latency
 - Jini relies heavily on **Java's mobile-code feature**
- Each Jini-enabled device has some memory and processing power
- Each device should be equipped with a JVM (Java Virtual Machine)
- Service components are implemented using Java
 - All the service components should live as Java objects to facilitate the service requester **to download and run code dynamically.**
 - However, Java does not expect a Java service implementation, only a **Java wrapper**

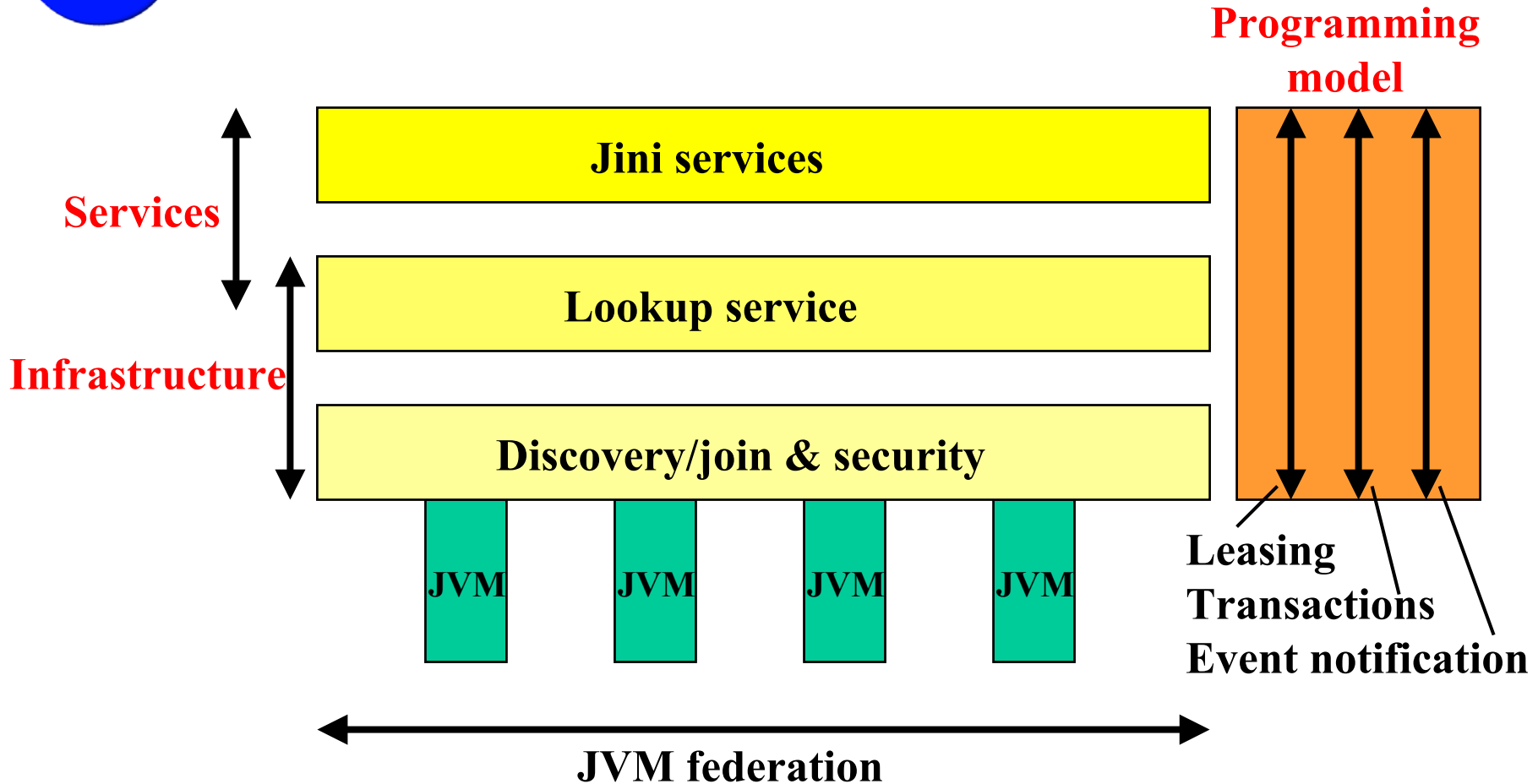


Relationship between Java and Jini





Jini components





Infrastructure component

- **Discovery and join protocol**: defines the way how
 - services **discover** other services
 - services **advertise** themselves
 - services **become part** of the federation
- **RMI**: enables service proxies to be downloaded
- Distributed security model
- Lookup service
 - serves as a **repository** of services
 - helps members **to find each other** in the Jini federation
 - **entries** are Java-compliant byte-code objects, which can be written in Java or wrapped by Java



Services component

- Services component denotes the services that together form the Jini community
- The services are identified as Java objects
- Each service has an **interface**, which defines the **operations** that can be requested from the service
- The interface also reflects the **service type**
- Basic services of Jini:
 - **Lookup service**
 - **JavaSpaces service**: optional distr. Persistence mechanism
 - **Transaction manager service**



Programming model component

- The programming model supports the following interfaces:
 - **Lease** interface, which extends the Java programming model by adding time to the notion of holding a reference.
Duration based model for allocating and freeing the resource references.
 - **Event notification** interface, which extends the JavaBeans event delegation.
 - **Transaction** interface, which allows OO transaction handling.



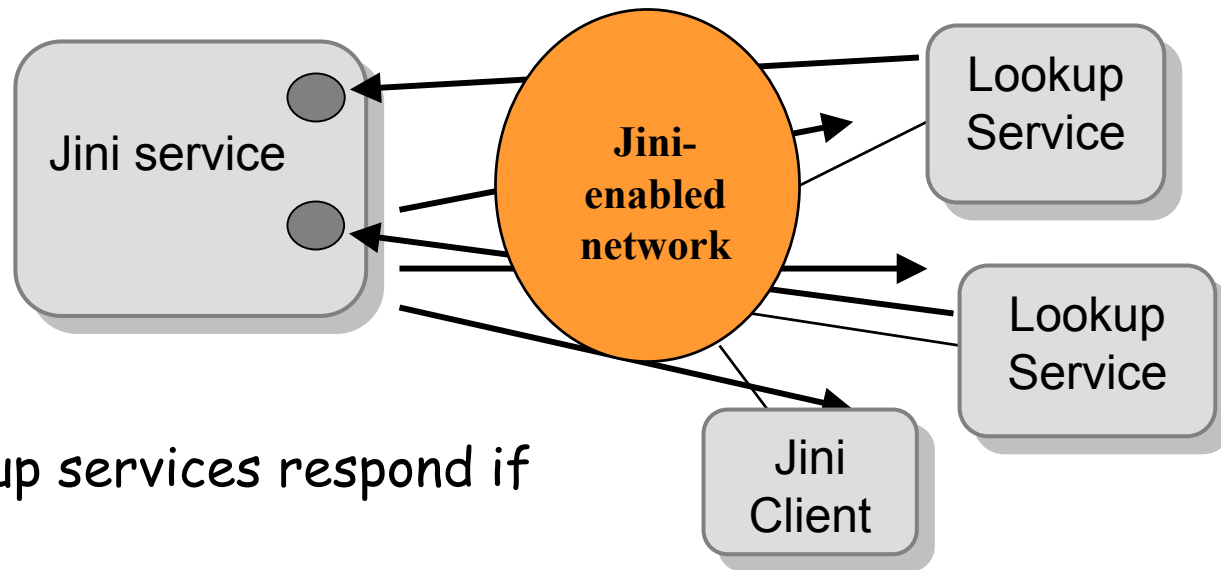
Operation in more detail

- The fundamental behaviour is defined by three protocols
 - **Discovery** - how to locate the Lookup Service
 - **Join** - how to register with the LS and export services
 - **Lookup** - how to find suitable services
- **Main operation steps**
 - Services export their services (in the form of Java objects)
 - Clients locate services and download objects or execution
 - Client-Service interaction (formation of a federation) is governed by need



Service provider registering (discovery protocol)

- The service performs a multicast discovery to find lookup services on the network.
- A multicast message is received by everyone on a network

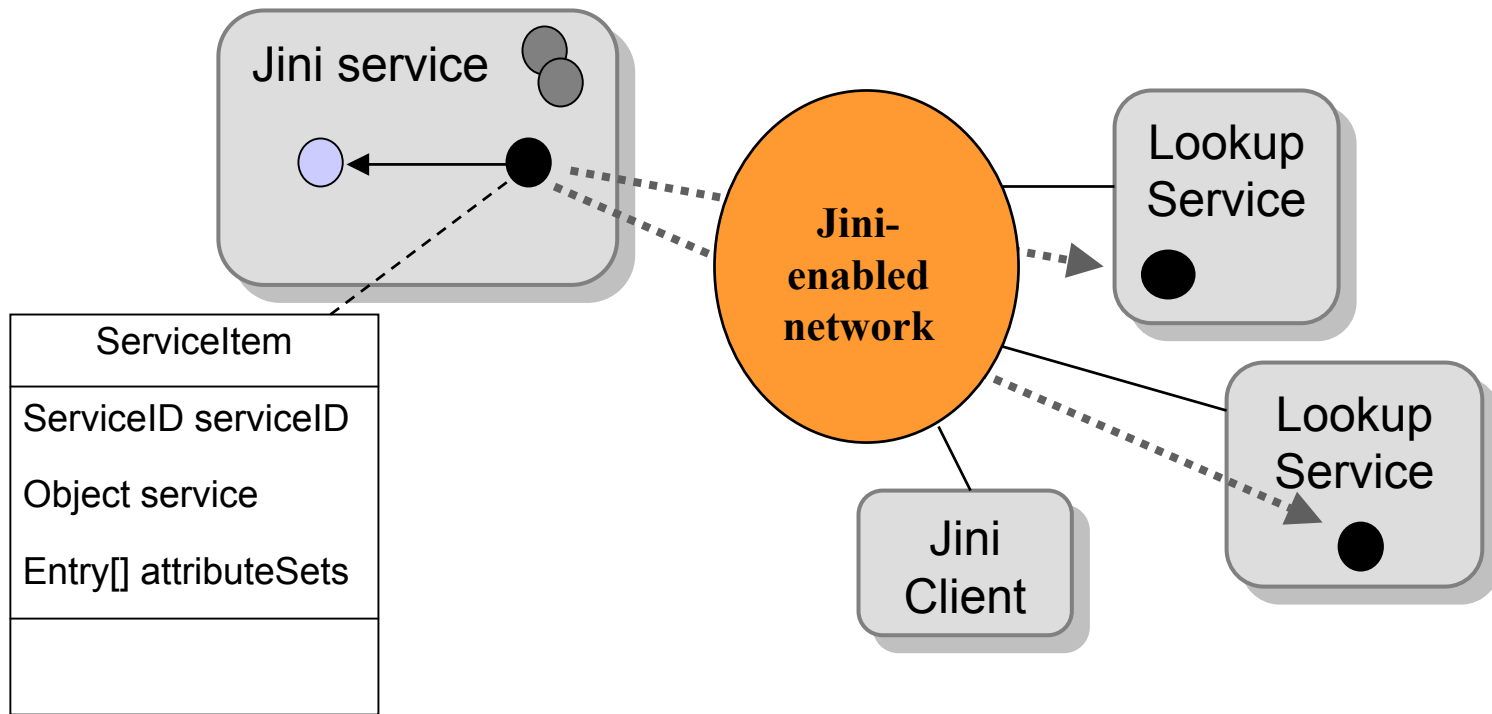


- Lookup services respond if alive
- Service receives a proxy object of the LS



Service provider registering 2 (join protocol)

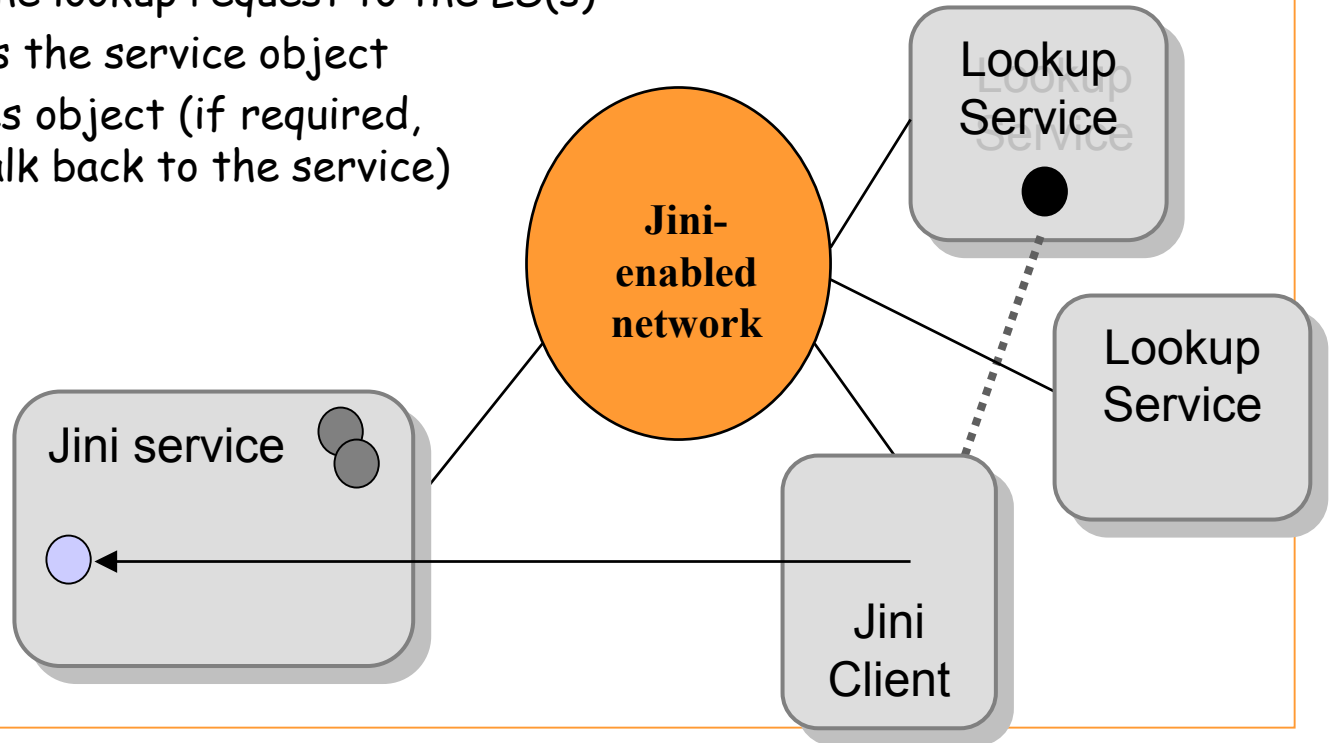
- Using the LS proxy, the service uploads its object and attributes to the lookup service





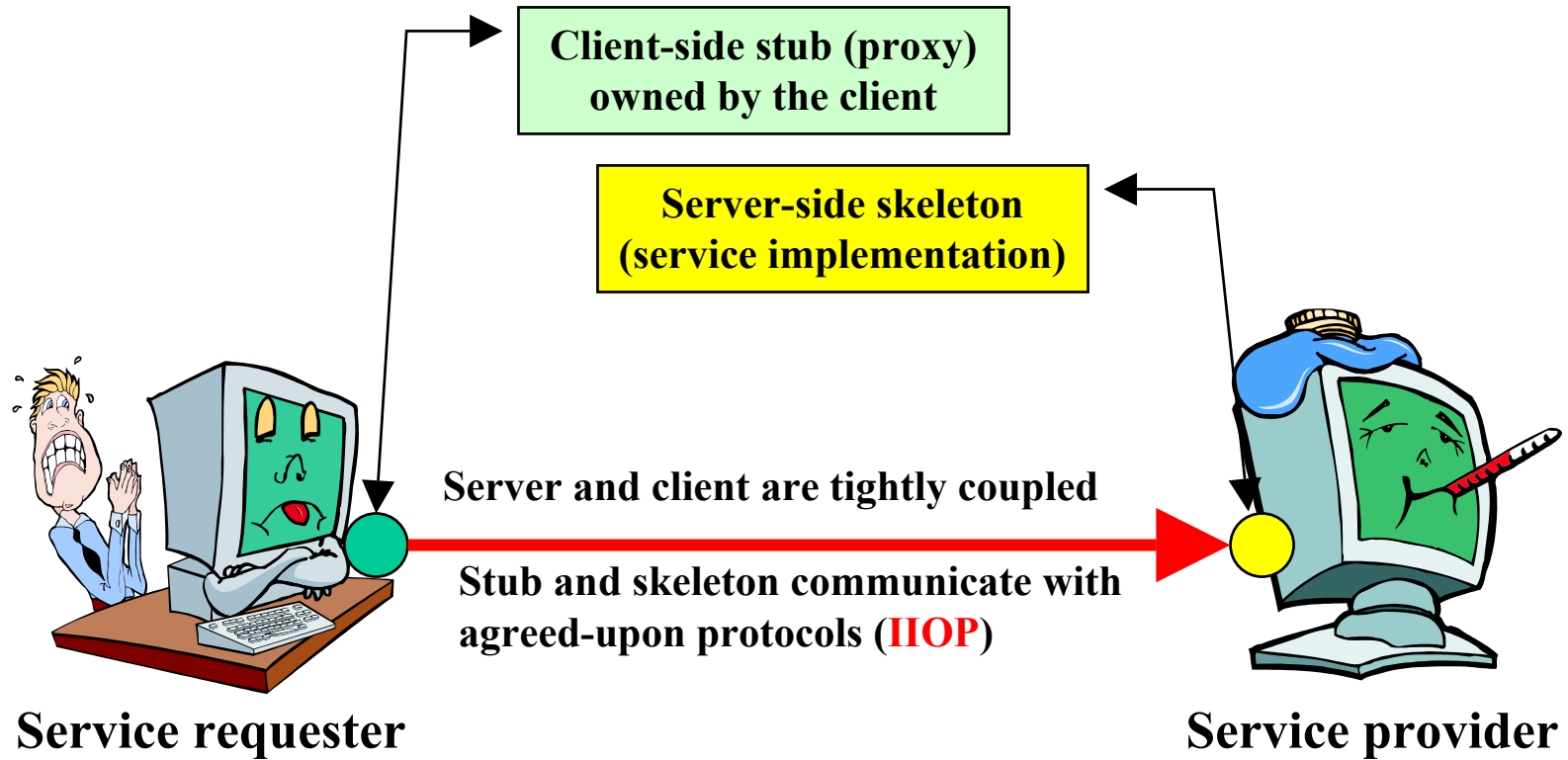
Client requesting service (lookup protocol)

- **Finding a service**
 - The client (already discovered the LS(s)) specifies the interface of the required service
 - Sends the lookup request to the LS(s)
 - Receives the service object
 - Executes object (if required, it can talk back to the service)



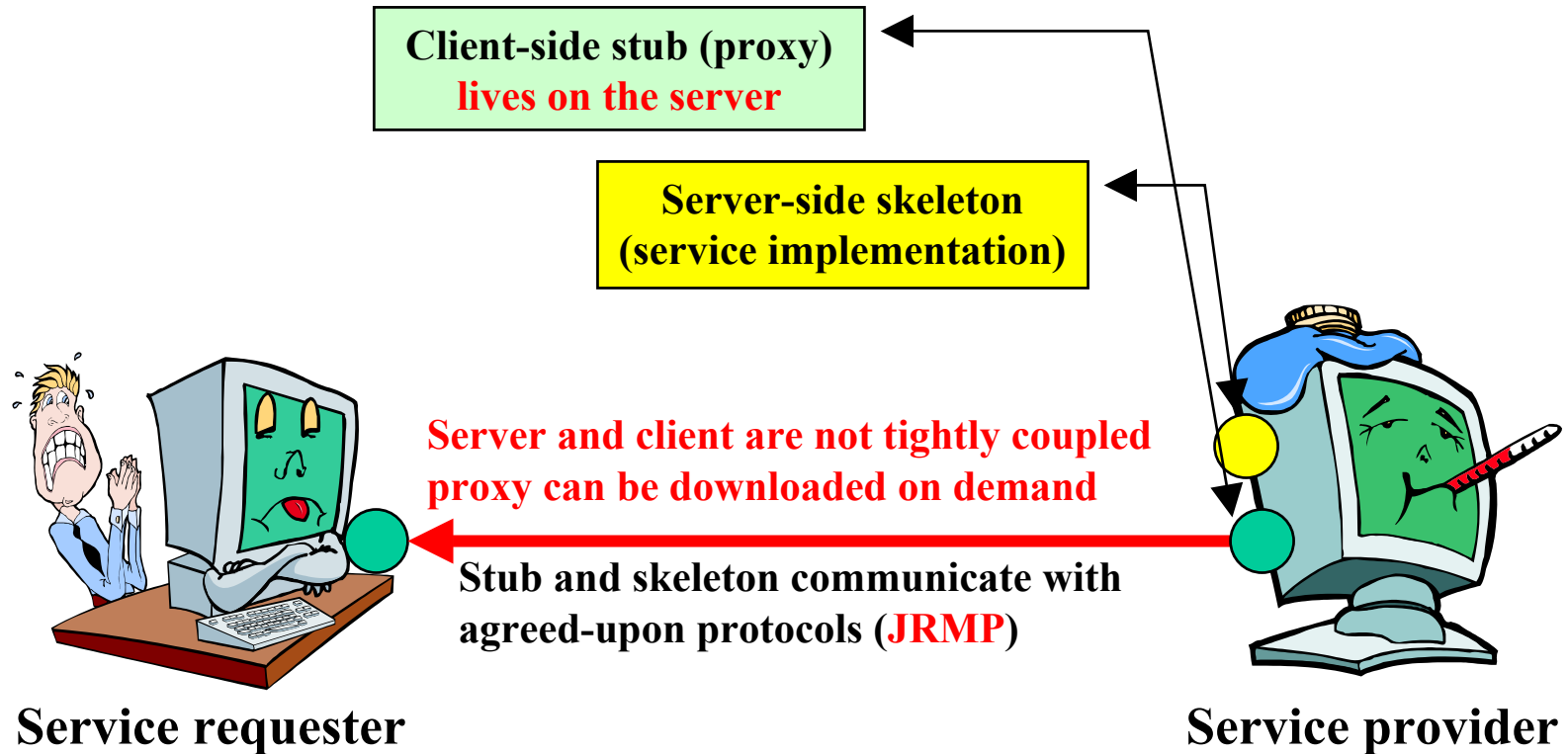


CORBA: Protocol-dependent system



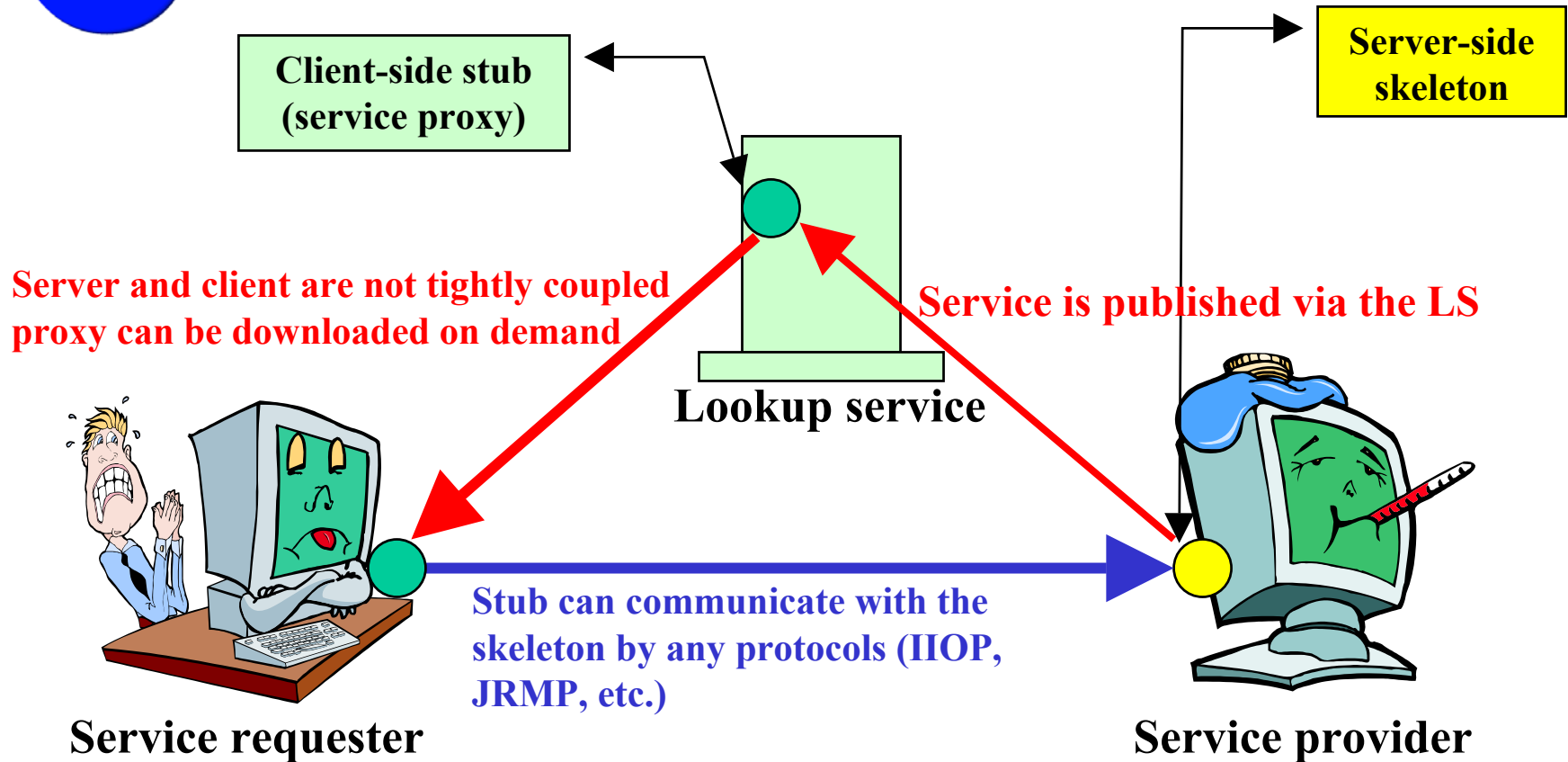


RMI: Protocol-dependent system





Jini: Protocol-independent system





Comparison of Jini and RMI

- **Similarity**
 - Both provide a mechanism for Java objects and applications (services) to communicate across JVMs.
- **Differences 1**
 - RMI handles protocol level issues
 - method invocation
 - passing parameter values
 - Jini handles higher-level issues like service interaction
- **Differences 2 (Fig. 3-29):**
 - RMI protocol-dependent, uses the JRMP protocol
 - Jini is protocol-independent, acts with any distributed protocol (JRMP, IIOP, ORPC)



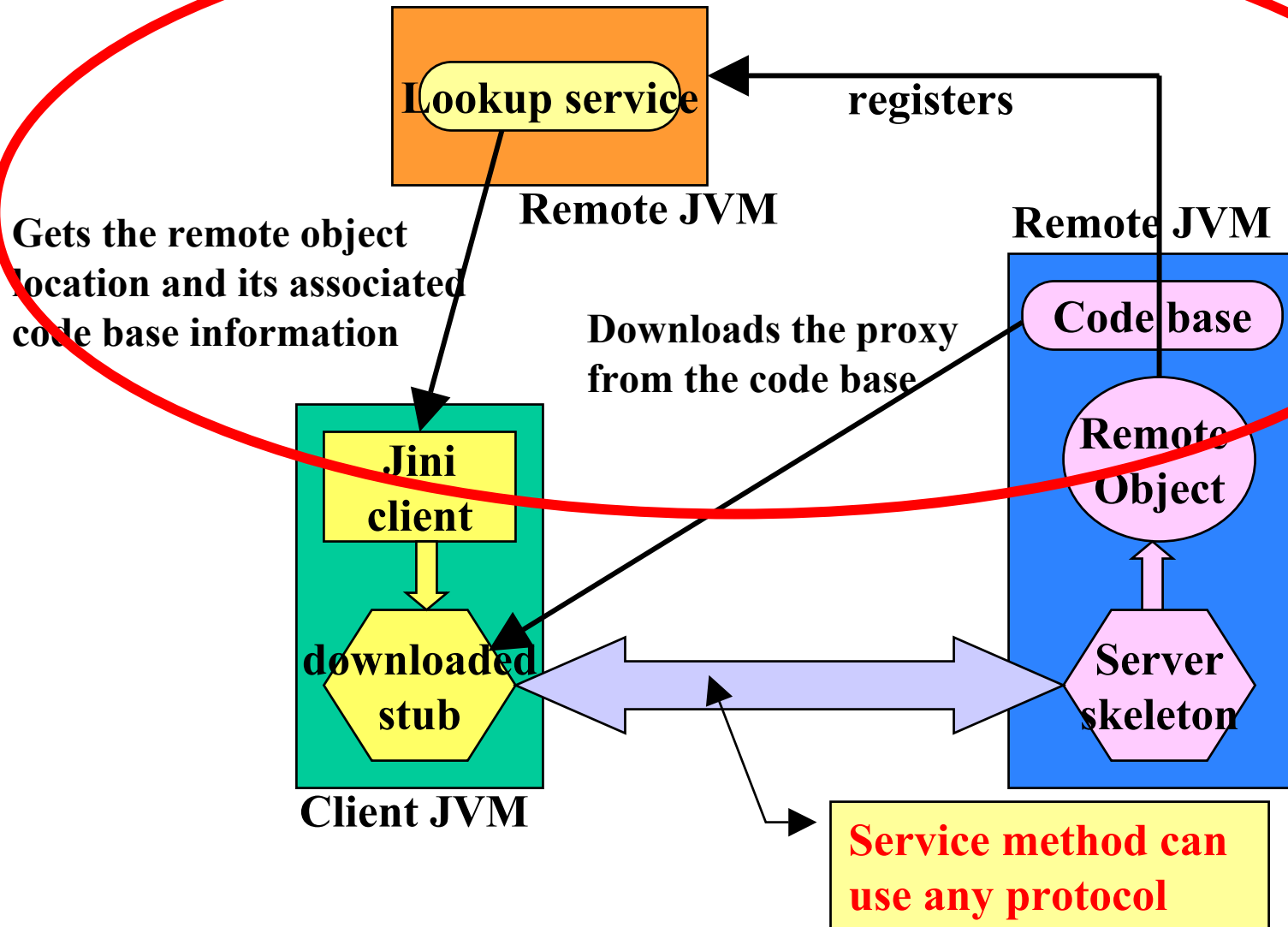
The role of RMI within Jini

- Jini is dependent on the RMI/Java environment and not the RMI protocol
 - Jini **uses the RMI/Java environment** to download the stub
 - but once the stub is downloaded it **can use any protocol** to communicate to its remote object



Jini on top of RMI

RMI environment





Differences between Jini and CORBA

- **CORBA** provides a **location- and protocol-specific** distr. Architecture (Fig. 3-29):
 - client must be prewired with the naming service
 - client must have relevant ORB libraries (client stubs)
 - client must use the IIOP protocol
- **Jini** is a **location- and protocol-independent** distr. Architecture (Fig. 3-29):
 - client need not know the location of a lookup service
 - client need not have stub libraries
 - stub can use any protocol
- **CORBA** can support many languages through IDL
- **Jini** requires Java Object wrapping

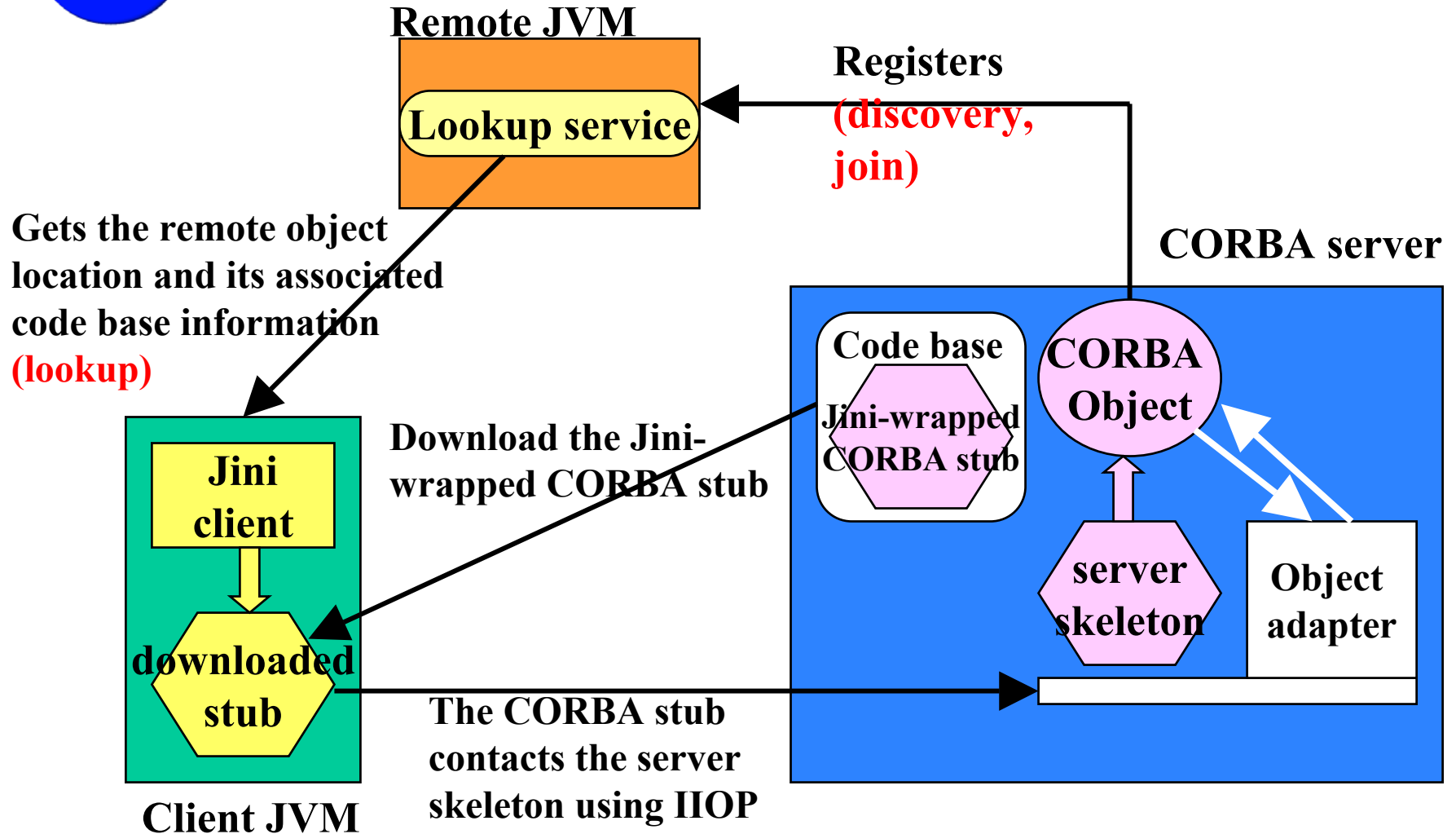


Co-operation between Jini and CORBA

- A Jini proxy can be written that can wrap the CORBA client proxy and can be registered in a Jini lookup server
- Discovery, registration, reregistration and leasing can be handled by a third-party Java component.
 - Any Jini client who would like to use the service can locate a lookup service and download the Java-wrapped CORBA client proxy
 - The client proxy can now communicate to its CORBA server working in the same way as in a CORBA environment
- This design is suitable if you have an ORB running on the client system.
- Otherwise use a bridge scheme.



Co-operation between Jini and CORBA

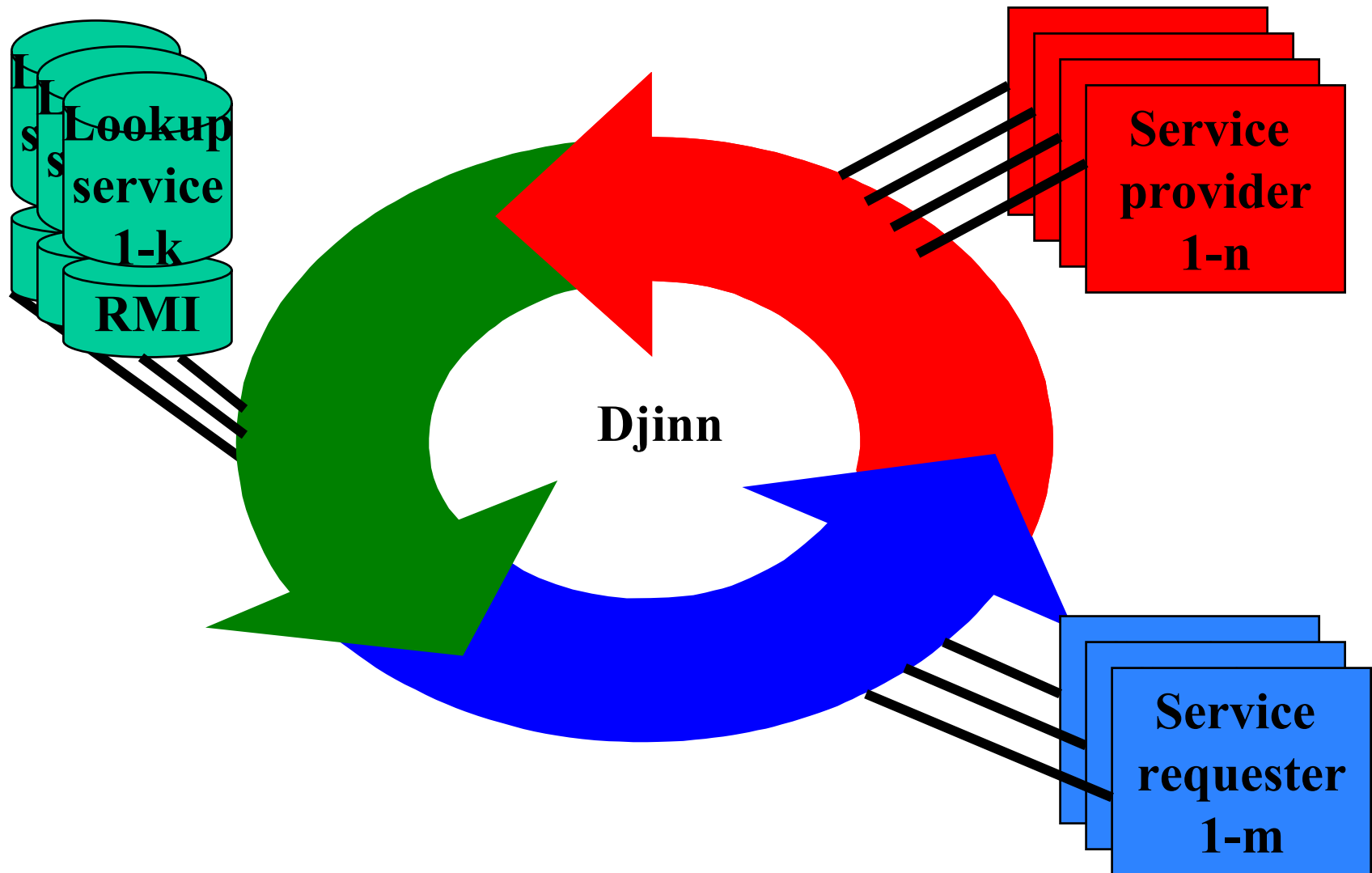




Three levels of scalability in Jini

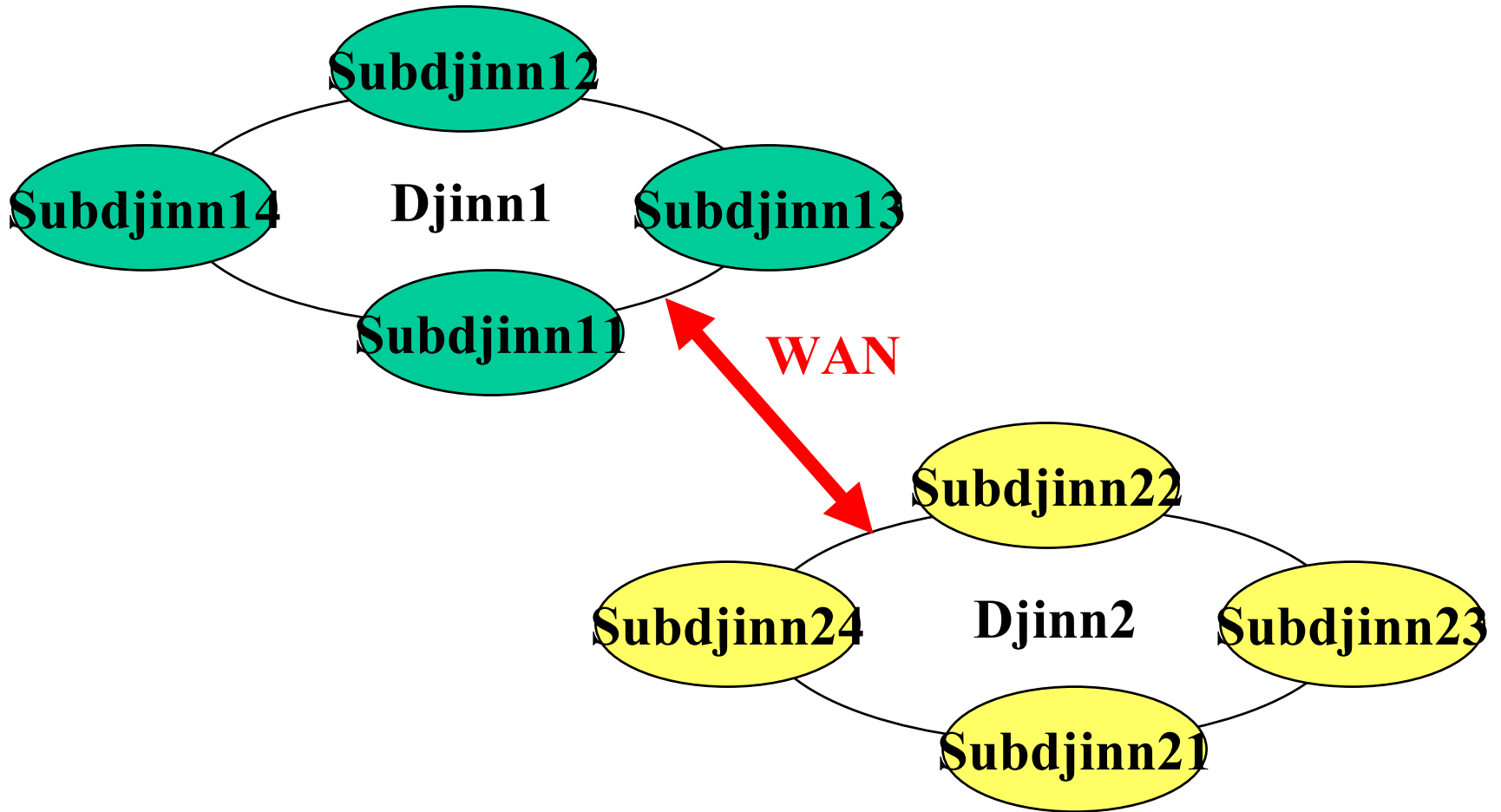
- Jini community (or **Djinn**): is a group of
 - lookup services
 - service providers
 - service requesters
- A djinn can contain smaller **subdjinns**
- Djinns can be connected to form larger distributed Jini service networks

Structure of a djinn





Levels of scalability in Jini





Applications and literature of Jini

- Remote access to clinical data
- Large-scale wish fulfilment support for organizing holidays, wedding, etc.
- Jini on wheels - the car as a mobile infrastructure
- Using Jini to enable a framework for agent-based systems
- Sing Li: Professional Jini, Wrox Press, 2000
- S. I. Kumaran: Jini Technology, Prentice Hall, 2002



Thank you