

**The Bio-Networking Architecture**  
**Bi-weekly report #14 (Dec. 9, 2002): Distributed Service Composition Protocol**

PI: Tatsuya Suda (suda@ics.uci.edu)  
University of California, Irvine, <http://netresearch.ics.uci.edu/bionet/>

**Introduction:**

In the Bio-Networking Architecture, complex applications emerge through autonomous interactions of multiple service components (called “cyber-entities”), similarly to biological entities cooperating to achieve a complicated task in the real biological world. This concept of composing an application through an autonomous interaction of multiple service components at runtime is called Dynamic Service Composition.

Dynamic Service Composition reduces application development time as a large-scale complex application is autonomously created from simpler service components. Applications created through Dynamic Service Composition are flexible and adaptable. For example, depending on a specific type of a display device available to a user (e.g. a small PDA display or a full color PC display), Dynamic Service Composition allows creation of applications using a service component that matches the capability of the display device (e.g., simple interface for a PDA display, full color and high resolution interface for a PC display). In addition, in dynamically creating applications, service components may be selected based on various resource constraints and system configurations (e.g., CPU speed, memory size, network bandwidth) so that application’s functionality and performance adapt to the context (i.e., various resource constraints and system configurations) that they run in.

**New Achievements:**

In the previous bi-weekly report #4 (submitted on July 22, '02), the PI has proposed Flexible Interface Definition, which defines interface of a service component in a flexible manner so that service components with different interfaces can communicate and interact with each other. In the bi-weekly report #9 (submitted on Sep.30, '02), the PI has also proposed Component Service Description Framework (CSDF), which allows a service component designer to describe service components so that the functionality and semantics of the components are obtained at runtime. Flexible Interface Definition is integrated into CSDF so that a service component described by CSDF exposes its interfaces (methods) in a flexible manner.

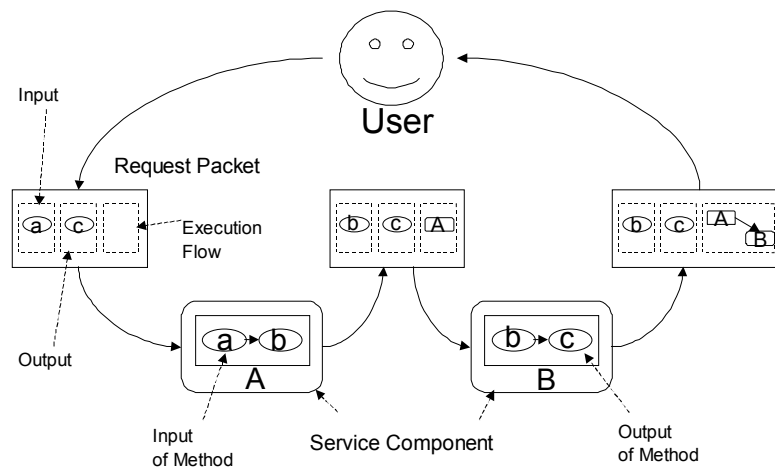
In addition to designing Flexible Interface Definition and CSDF, the PI is currently developing a mechanism (referred to as Distributed Service Composition Protocol, or DSCP, in this report) to compose an application from a group of interacting service components. This mechanism relies on CSDF such that it assumes that every service component has a CSDF file expressing its functionality. Therefore, the DSCP mechanism described in this report benefits from the features of both Flexible Interface Definition and CSDF described in the previous reports #4 and #9. This DSCP mechanism is designed as a fully distributed mechanism for better scalability and robustness. The following describes the algorithm and the features of DSCP.

**Distributed Service Composition Protocol (DSCP):**

Distributed Service Composition Protocol (DSCP) is a protocol that discovers an execution flow of a composite application, which consists of several service components. DSCP is based on Component Service Description Framework (CSDF) and assumes that a service component is

defined and described as a set of Methods, Variables and Assertions. (See the bi-weekly report #9 for detailed explanation of CSDF.) DSCP also assumes that, when a user requests for a composite application, he specifies the Input and the Output of the composite application. For instance, if a user requests a travel planner application, he requests by specifying “an itinerary” as the Input, and “travel plans” as the Output.

In DSCP, when a user specifies the Input and the Output of the composite application, he broadcasts a composition request packet containing the Input and the Output. The composition request packet is forwarded among service components, following a given routing rule. At each service component, the Input in the composition request packet is compared against the Input of the Method of the component. If the two match, then the Output in the composition request packet is compared against the Output of the Method of the component. If they match, the composite application that the user requested is found, and the composition request packet is sent back to the user. Since the composition request packet records which service components participated in providing the requested composite application, the user obtains the execution flow of the composite application he requested from the returned composition request packet. If the Output in the composition request packet does not match the Output of the Method of the component, the Input in the composition request packet is replaced by the Output of the Method of the component, and then the composition request packet is broadcasted to other service components again. The composition request packet is forwarded among service components until the requested composite application is found.



**Figure 1: Distributed Service Composition Protocol**

There are two main features that make DSCP unique among other existing service composition mechanisms. First, DSCP is a fully decentralized protocol. Because a request packet is directly passed among service components, no centralized entity is required to compose an application. Because of this decentralization feature, DSCP is scalable to the size of the network. In addition, DSCP becomes more efficient than centralized mechanisms when the frequency of the composition request becomes higher. This is because the task of composing applications is distributed among service components in DSCP. Second, DSCP does not require any template to compose an application. A template is a description that defines the structure of a composite application. Although a template allows creating a complex application, it reduces the adaptability of the application created with a template, and types of applications created with a template may be limited. DSCP does not require any templates and is still able to compose complex applications.