Overview

Infinetics is pleased to present to OOI a proposal to aid in the development and prototyping of its Distributed Inter-process Communication (DIPC) architecture. This proposal has two broad aspects: 1) to aid OOI in refining the specification and development of a robust scalable DIPC architecture that will support OOI requirements in the long term, and 2) to get initial experience with the approach by doing a prototype implementation.

Phase 1 is expected to start on 8/15 and run through 12/15. Phase 2 will start at the end of phase 1, and is expected to run from 12/15/09 to 12/15/10. The scope of work and deliverables for phase 2 will be determined during phase 1.

As part of Phase 1, we will work closely with OOI personnel to understand how the PNA concepts can be integrated into the OOI architecture as it is currently understood to make it simpler, more robust, more secure, more flexible, and use fewer resources. Part of this work will be mapping AMQP concepts into PNA concepts and exploring the implications of that mapping and what additional capabilities and efficiencies would result.

The Phase 1 development contract will specify acceptance criteria for the software and documentation that is acceptable to both parties. This proposal outline will be expanded to include details of the documents, software modules, file formats, and APIs that will be implemented. The implemented DIF will be a faithful but minimal implementation of the architecture outlined in Chapter 7 of “Patterns in Network Architecture.”

Architecture Assumptions

We are building a reusable, component-based DIF framework with easily adjustable policies to allow a node to satisfy many roles without writing new code as much as possible. We will use a structured-data based application service interface to transfer data and metadata to the PNA services in a flexible and extensible manner. The application data transfer API will be an asynchronous and non-blocking interface with minimal impact on system and application performance to permit its use in data-transfer-intensive applications.

We will use FUSE, available on Linux/Unix/Mac OS/Windows, to allow rapid development of service components in user space to avoid kernel debugging. DIF functionality will be accessible from applications using Unix/Posix file system calls and protection mechanisms, wrapped in a client communications library to simplify use.
**Phase 1 Deliverables**

- A Client interface definition, followed by a review and signoff. The interface will include a message oriented API, with an abstract object model description using an XML DTD, ASN-1, or an equivalent description meta-language. Publish subscribe or filtering semantics will not be part of this definition. It will specify the interface to the transport architecture only.
- Software components to demonstrate basic DIF running PNA-DTP over UDP or TCP as necessary, running on Ubuntu 9.04
- Documentation of the public interfaces, object models, and configuration information
- Test programs that demonstrate compliance with the acceptance criteria in this Statement of Work
- All running on Linux, written in C/C++. SWIG is a publicly available tool that can be used by OOI to export C/C++ interfaces to Python and other languages
- User process to include bare bones DIF framework in which protocol machines and other DIF services operate
- Minimal DIF: UDP or TCP used for transport as appropriate to QoS, client-side API as described above, startup node address resolution and physical layer DIF joining
- Client library to wrap read/write/ioctl, C interface
- Minimal management interface that shows alive nodes and services
- Fail-safe watchdog automatic restart of DIF local IPC service
- Source code and build scripts to allow modification and evaluation by OOI
- Report on the role of DIFs in OOI

**Phase 2 Deliverables Working List**

- Full client API specification
- Authentication, encryption
- Proposal of store/forward/filter DIF semantics to satisfy OOI architectural needs
- Application view of store/forward/filter
- Software components: messaging service, hierarchical DIFs

**Schedule**

Start: 8/15  
8/15 Kickoff meeting  
9/15 Initial client API interface proposal  
10/1 Initial interface signoff  
11/1 Proposal for phase 2 work  
12/1 Demo first cut delivered  
12/15 Final demo signoff  
End: 12/15

**Resources**

Steve Bunch
John Day  
Chris Williams  
Gilbert Glick, Peter DeWolfe, Peter Sherbin, Mark Levine

**Cost**

Fixed price contract comprising:

$XXX Software license fee  
$XXX plus approved expenses (travel, etc)

$XXX Total contract price

<table>
<thead>
<tr>
<th>Date</th>
<th>Payment Type</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>8/15/09</td>
<td>Initial payment</td>
<td>$XXX</td>
</tr>
<tr>
<td>9/15/09</td>
<td>Progress payment</td>
<td>$XXX</td>
</tr>
<tr>
<td>11/1/09</td>
<td>Progress payment</td>
<td>$XXX</td>
</tr>
<tr>
<td>12/1/09</td>
<td>Demo delivery payment</td>
<td>$XXX</td>
</tr>
<tr>
<td>12/15/09</td>
<td>Final payment</td>
<td>$XXX</td>
</tr>
</tbody>
</table>

**Acceptance Criteria**

1. Compile and build all components using delivered build scripts  
2. Start DIF subsystem on 10 boxes  
3. Modify default configuration files to create 2 separate 5-node DIFs at the same rank as the 10-node DIF, and start them in parallel with the 10-node DIF  
4. Start demonstration service on two nodes of each DIF  
5. Start demonstration client application on all 10 nodes, accessing various services in the 2 DIFs  
6. Examine system log files to determine:  
   a. Service enrollment  
   b. Display of all visible nodes and services started on each DIF  
   c. Client locating service instance  
   d. Client sending request to service instance  
   e. Client receiving response from service instance  
   f. Measure bulk data transfer throughput  
   g. Client failure attempting to connect to service on the second DIF that it is not enrolled in

**Ownership of Deliverables**

The OOI and Infinetics will own all software created by Infinetics non-exclusively in accordance with terms and condition specified in the JOI subaward JSA 7-11, subject to the payment of the license fee specified above. Both parties will have full independent ownership rights to exploit said deliverables in any manner and are not required to obtain the consent of or permission from the other party as long as such exploitation is
consistent with the JOI subaward JSA 7-11. Source code will be delivered to JOI at the completion of the prototype phase of the project in December 2010.