



Richard

JPL

SOIF

Message Transfer Service

Overview

Ashton Vaughs, NASA/JPL

October 23, 2003



Message Transfer Service Agenda

JPL

- Overview
- Draft API
- Required Infrastructure
- Prototype
- Future Work

October 23, 2003

SOIF Message Transfer Service



Pg 2



Message Transfer Service

Overview

October 23, 2003

SOIF Message Transfer Service



Pg 3



Message Transfer Service (MTS) Definition

A set of common functions for message transfer and Quality of Service (QoS) management that sits above the transport layer in the communications stack and provides a service API for mediating the transfer of data between processes in a distributed system.

Assumptions:

- The processes run on a single system or multiple systems that may be heterogeneous
- The network can be LAN, WAN (Wire or RF) and Internet
- The Transport Layer functionality may be provided by TCP/IP or SCPS, or by a direct Data Link, or by efficient mechanisms such as shared memory, pipes, or other IPC.

October 23, 2003

SOIF Message Transfer Service

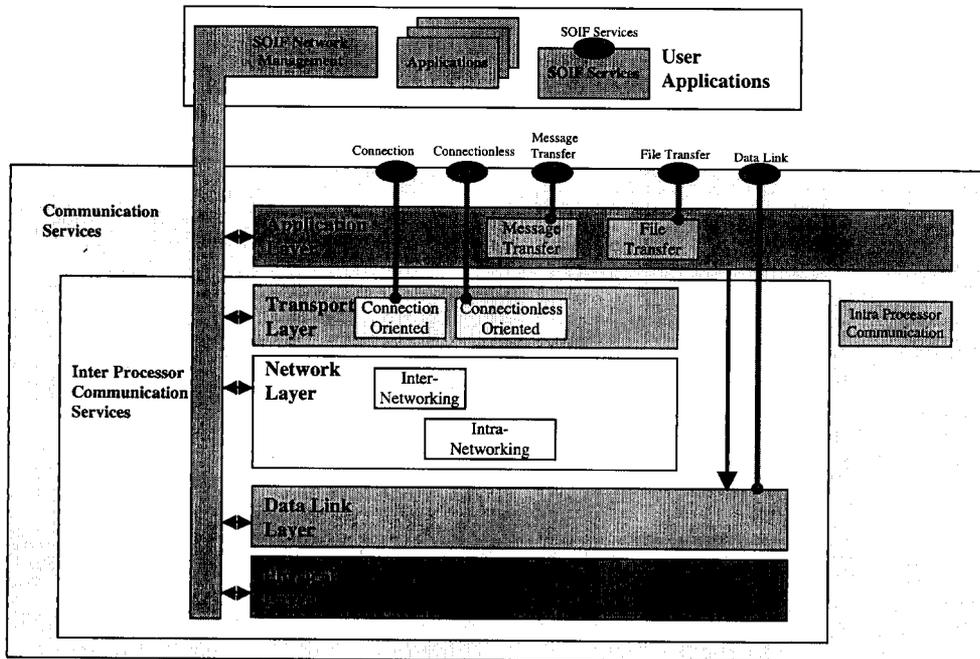


Pg 4



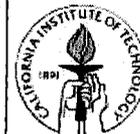
Message Transfer Service

SOIF Reference Model



October 23, 2003

SOIF Message Transfer Service



Pg 5

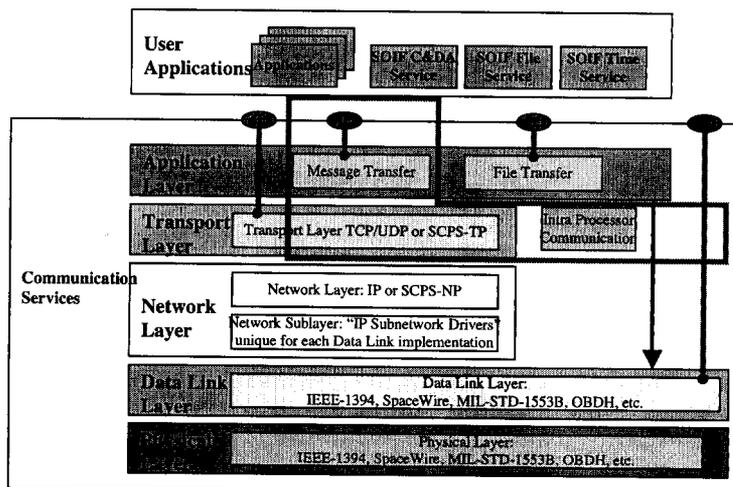


Message Transfer Service

Context



- Provides Message Transfer interface and confirmed & unconfirmed message delivery
 - Compliance Test Port for Message API
 - Socket I/F to top of TCP/UDP
 - Or interface to shared mem / pipes / data link



October 23, 2003

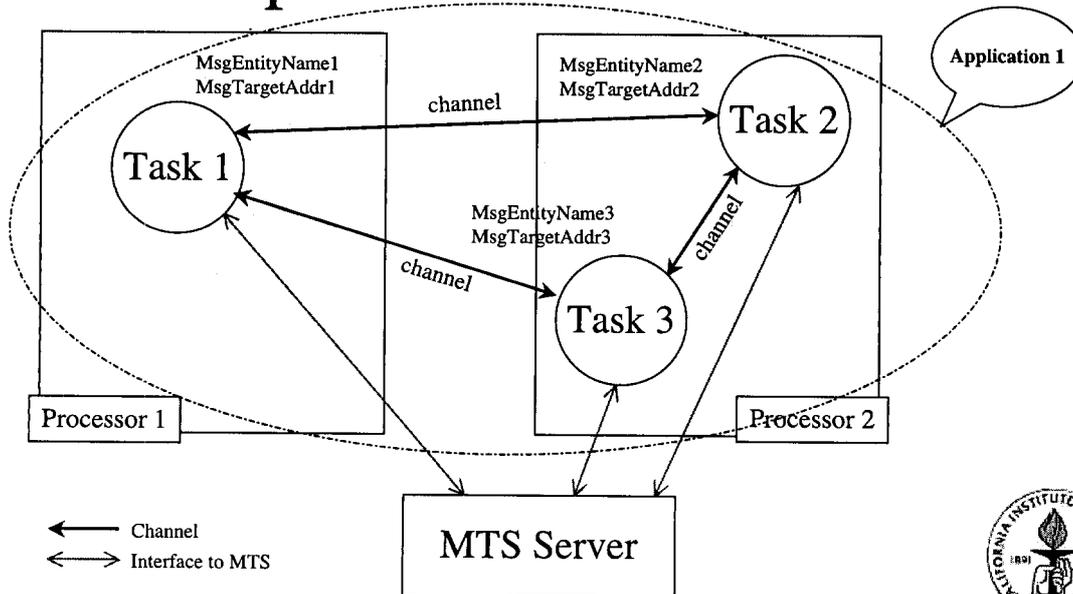
SOIF Message Transfer Service



Pg 6



Message Transfer Service Concept Illustration¹



October 23, 2003

SOIF Message Transfer Service

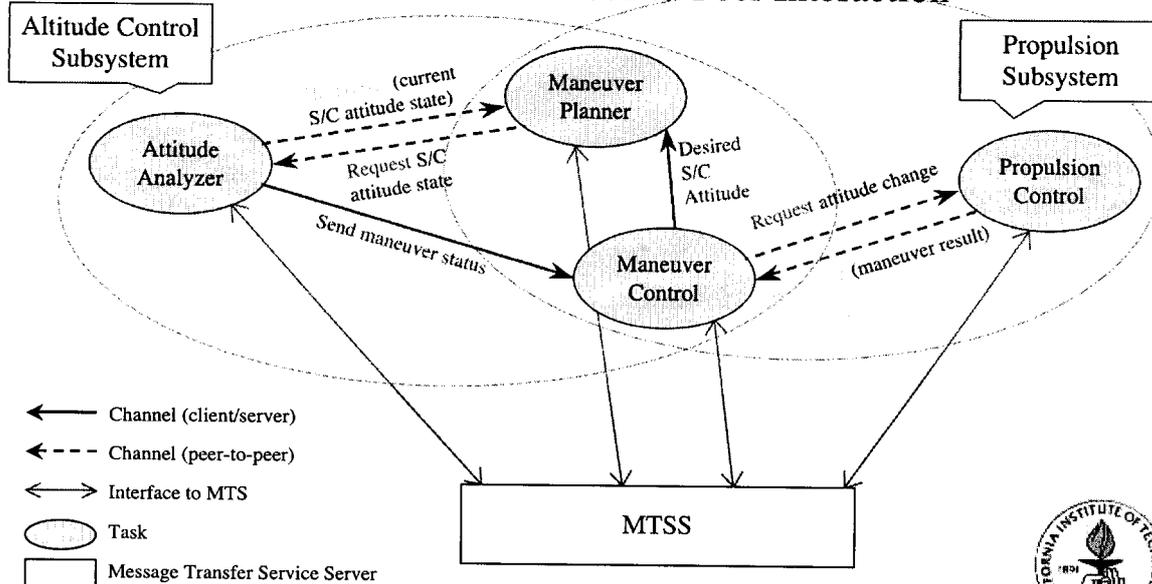


Pg 7



Message Transfer Service

Use Case: Client/Server & Peer-to-Peer Interaction²



October 23, 2003

SOIF Message Transfer Service



Pg 8



JPL

Message Transfer Service

Draft API

October 23, 2003

SOIF Message Transfer Service



Pg 9



JPL

Message Transfer Service

Draft API

- 1.0 Service Setup
- 2.0 Level of Service (LoS)
- 3.0 Communication
- 4.0 Send/Receive (Blocking)
- 5.0 Send/Receive (Non-blocking)
- 6.0 Event Notification (Callback)
- 7.0 Status Test

October 23, 2003

SOIF Message Transfer Service



Pg 10



Message Transfer Service

Draft API - 1.0 Service Setup

- 1.1 initializeMsgService (char *mtss_name[in], MSHandle **msh[out])
establish a connection to the specified Message Transfer Service Server (MTSS)
- 1.2 finalizeMsgService (MSHandle **msh[in/out])
co-ordinate the termination of all of the calling Application's MTS related connections and resources
- 1.3 resolveName (MSHandle *msh[in], MsgEntityName entityName[in], MsgTargetAddr targetAddr[out])
return the targetAddr associated with the supplied entityName if the entityName exists within the MTSS
- 1.4 bindName (MSHandle *msh[in], MsgEntityName entityName[in])
establish an association of the between the calling Application and the entityName within the specified MTSS
- 1.5 unbindName (MSHandle *msh[in], MsgEntityName entityName[in])
remove the association between the calling Application and the entityName within the MTSS

October 23, 2003

SOIF Message Transfer Service



Pg 11



Message Transfer Service

Draft API - 2.0 Level of Service

- 2.1 queryLoSCapability (MSHandle *msh[in], MsgTargetAddr *targetAddr[in], LoS *los[out])
inquire the MTSS about the Level of Service capabilities supported by the specified target address
- 2.2 setLoSCapability (MSHandle *msh[in], LoS *los[in])
register calling applications Level of Service capabilities with the MTSS associated with the given MS Handle

October 23, 2003

SOIF Message Transfer Service



Pg 12



Message Transfer Service

Draft API - 3.0 Communication

- 3.1 `openChannel` (MSHandle *msh[in], MsgEntityName from, MsgEntityName to, MSChannel **channel[out], LoS *los[in], struct timeval *timeout[in])
Establish a unidirectional communication channel, between the calling application and another application
- 3.2 `closeChannel` (MSChannel **channel[in/out])
close the given channel established with another application

October 23, 2003

SOIF Message Transfer Service



Pg 13



Message Transfer Service

Draft API - 4.0 Send/Receive (Blocking)

- 4.1 `sendMsg` (MSHandle *msh[in], MSChannel *channel[in], char *buffer[in], int datalen[in], struct timeval* timeout[in])
send a message through the channel in a blocking mode with the given message buffer and size within the specified time interval
- 4.2 `receiveMsg` (MSHandle *msh[in], MSChannel *channel[in], char *buffer[in/out], int buflen[in], int *datalen[out], struct timeval* timeout[in])
receive a message through the given channel in a blocking mode within the specified time interval

October 23, 2003

SOIF Message Transfer Service



Pg 14



Message Transfer Service

Draft API - 5.0 Send/Receive (Non-blocking)

- 5.1 `sendMsgPoll` (MSHandle *msh[in], MSChannel *channel[in], char *buffer[in], int datalen[in], RequestID reqID[in])
Send a message from the calling application to another application using the given channel returning control to the calling application immediately without waiting for the transfer of bytes to complete. Will check the send status periodically.
- 5.2 `sendMsgAsync` (MSHandle *msh[in], MSChannel *channel[in], char *buffer[in], int datalen[in], RequestID reqID[in], (void*) *asyncHandler[in](void), void* arg)
Send a message without waiting for its completion. An event handler is specified to be invoked when the send operation completes with or without an error.
- 5.3 `receiveMsgPoll` (MSHandle *msh[in], MSChannel *channel[in], char *buffer[in], int buflen[in], int datalen[out], RequestID reqID[in])
Receive a message through a given channel without waiting for its completion. Will check for its completion periodically
- 5.4 `receiveMsgAsync` (MSHandle *msh[in], MSChannel *channel[in], char *buffer[in], int buflen[in], int datalen[out], RequestID reqID[in], (void*) *asyncHandler[in](void), void* arg)
Receive a message through a given channel without waiting for its completion. An event handler is specified to be invoked when the receive operation completes with or without an error.

October 23, 2003

SOIF Message Transfer Service



Pg 15



Message Transfer Service

Draft API - 6.0 Event Notification (Callback)

- 6.1 `registerEventHandler` (MSHandle *msh[in], MSChannel *channel[in], (void*) *eventHandler[in](void), void* arg) register an event handler specifically for asynchronous request (send/receive) events
- <eventHandler> (MSHandle *msh[in], MSChannel *channel[in], RequestID reqID[in], char *buffer[in], int len[in])
an event handling function prototype
- 6.2 `eventLoop` (struct timeval timeout)
loop to receive and dispatch events, return when timeout or error

October 23, 2003

SOIF Message Transfer Service



Pg 16



Message Transfer Service

Draft API - 7.0 Status Test

7.1 pollChannel (MSChannel *channel[in], RequestID reqID[in], RequestStatus* reqStatus(out))
check the status of a request for the given channel and requestID

October 23, 2003

SOIF Message Transfer Service



Pg 17



Message Transfer Service

Draft API - 8.0 Terminology

- **Application** - a application may consist of one or more tasks that run on a single processor or multiple processors to achieve a specific set of functional requirements
- **Channel** - a logical connection between two or more applications
- **CoS** - Class of Service, e.g. guarantee/non-guarantee, reliable/unreliable, store and forward
- **LoS** - Level of Service is a table of CoS and QoS that are provided to channels between two message entities
- **MS Channel** - a unique handle to a channel information block that is generated by and assigned to a channel by MTS

October 23, 2003

SOIF Message Transfer Service



Pg 18



Message Transfer Service

Draft API - 8.0 Terminology (cont.)

- **MS Handle** - an association between the Application and Message Transfer Service Server
- **Msg Entity** - an application task unit that uses the Message Service to send/receive messages from another entity
- **Msg Entity Name** - a user-friendly name to identify the entity that uses the Message Service. It can be a service function that the application provides.
- **Msg Target Addr** - a unique handle for a specific message peer corresponding to one and only one symbolic name
- **Msg Svc Handle** - a handle for the message service returned to a task by the initializeMsgService call



Message Transfer Service

Draft API - 8.0 Terminology (cont.)

- **Name Format** - a hierarchical name format structure which may include the group name for the multicast purposes
- **Name Service** - an independent software module that maps a message entity name to its target address
- **QoS** - Quality of Service
- **Request ID** - unique identifier for messages sent via non-blocking calls
- **Task** - a process that owns its own address space and acts as an entity to communicate with other tasks through channels (connections)





JPL

Message Transfer Service

Required Infrastructure

October 23, 2003

SOIF Message Transfer Service



Pg 21



JPL

Message Transfer Service

Required Infrastructure

- Mechanisms to support early and late binding of connections
- Mechanisms to support dynamic (re-)binding of connections
- Support the guaranteed bandwidth and latency for resource reservation
- Mechanisms for negotiation of connection characteristics and QoS parameters
- Means for signaling the result of data transmission which may be link dependent (at least Red-failure, Yellow-don't know or partial, Green-ok)
- Mechanisms to handle node failures (MTS Servers and Applications)

October 23, 2003

SOIF Message Transfer Service



Pg 22



JPL

Message Transfer Service Required Infrastructure (cont.)

- Mechanisms to provide Secure Access to and Control of MTS servers
- Mechanisms to Configure MTS Servers and Federations of nodes
- Mechanisms to communicate “knowledge” among MTS Servers
- Mechanisms for Applications to discover MTS Servers or other Applications in the absence of MTS Servers
- Mechanisms for searching by other criteria (e.g. type and attribute)
- Mechanisms for specifying temporary connections (e.g. lease durations & renewals)

October 23, 2003

SOIF Message Transfer Service



Pg 23



JPL

Message Transfer Service

Prototype

October 23, 2003

SOIF Message Transfer Service



Pg 24

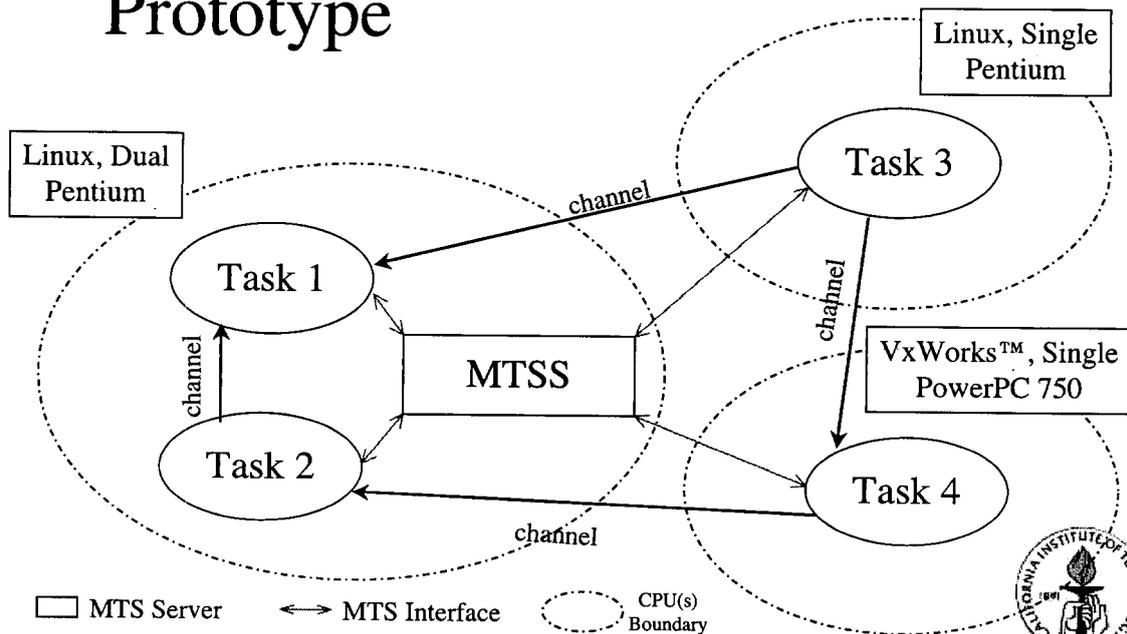


Message Transfer Service Prototype

- Heterogeneous Operating Systems
 - Linux
 - VxWorks™
- Heterogeneous Processor Architectures
 - Intel™ x86
 - Motorola PowerPC™ 750
- Heterogeneous Networks³
 - Ethernet
 - 1394a (FireWire™)



Message Transfer Service Prototype





JPL

Message Transfer Service

Future Work

October 23, 2003

SOIF Message Transfer Service



Pg 27



JPL

Message Transfer Service

Future Work

- Complete Infrastructure Requirements
 - Fault Tolerance
 - Name and other Attributes Updating
 - Information Sharing among Applications & Servers
 - Management of Groups of Nodes
- Determine Server Management Model
 - Peer-to-Peer
 - Client-Name Server(s)
 - Peer-to-Peer/Client-Name Server(s) Hybrid
- Extend and Finalize Draft API

October 23, 2003

SOIF Message Transfer Service



Pg 28



JPL

Message Transfer Service

Backup Slides

October 23, 2003

SOIF Message Transfer Service



Pg 29



JPL

Message Transfer Service Technical Contributors

- Chialan Hwang, NASA/JPL
- Ricardo Hassan, NASA/JPL

October 23, 2003

SOIF Message Transfer Service



Pg 30



Message Transfer Service Notes

^{1,2} Based on original diagram by C. Hwang, NASA/JPL

³ Work on 1394a network not completed at time of printing

October 23, 2003

SOIF Message Transfer Service



Pg 31