cMsg - a general purpose, publish-subscribe, interprocess communication implementation and framework

Carl Timmer, D Abbott, V Gyurjyan, G Heyes, E Jastrzembski and E Wolin

Thomas Jefferson National Accelerator Facility, Newport News, VA 23606

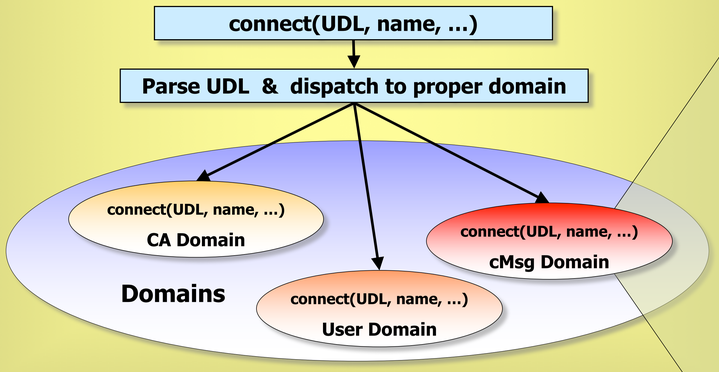
cMsg aims to unify communication packages under a single, message-passing API (Java, C, C++). An underlying package (or domain) may not implement all functions. This table contains a simplified list of the major client functions. The messages contain several user-settable fields*.

Function	Description
connect(UDL, myName)	Connect to a cMsg system specified by the UDL for client myName
disconnect()	Disconnect from the cMsg system
send(msg)	Send a message synchronously
flush(timeout)	Flush messages to send from client
syncSend(msg, timeout)	Send a message and wait for server response
sendAndGet(msg, timeout)	Send a message and wait for receiving client to send a response
subscribe(subject, type, callback)	Subscribe to messages of a given subject & type, registering a callback for incoming messages
unsubscribe()	Remove a subscription
subscribeAndGet(subject, type, timeout)	Subscribe to a subject & type and wait for one response
start()	Start receiving messages
stop()	Stop receiving messages
monitor(command)	Synchronous call to request monitoring information

***User-settable fields include strings, an int, a time, and a byte array. The next release of cMsg will allow any number of strings, primitive types, messages, binary and arrays of each.**

an API

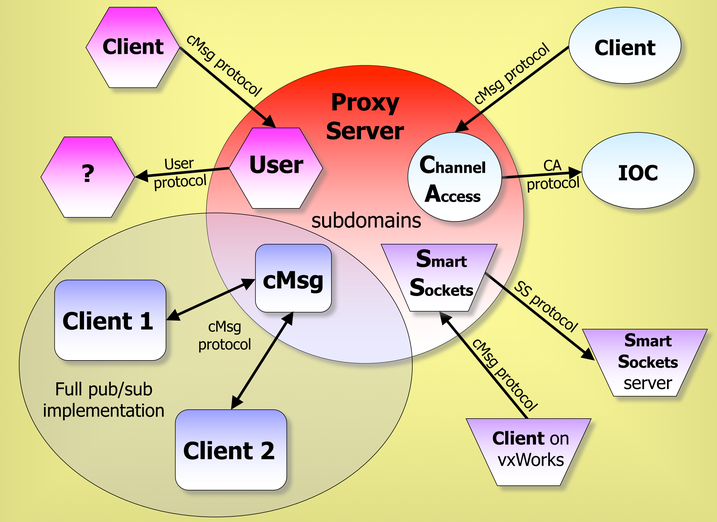
The cMsg client API is implemented as a thin dispatching layer to the underlying domains.



a Framework

cMsg is …

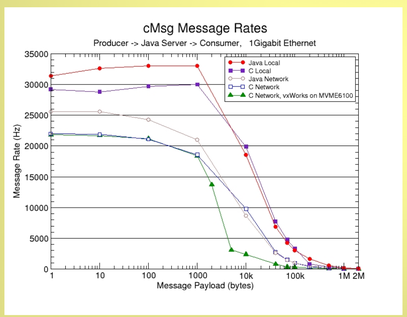
This is a view of the client and server sides of the cMsg domain, which implements pluggable subdomains, including a full publish/subscribe messaging subdomain. The proxy server is written in Java.



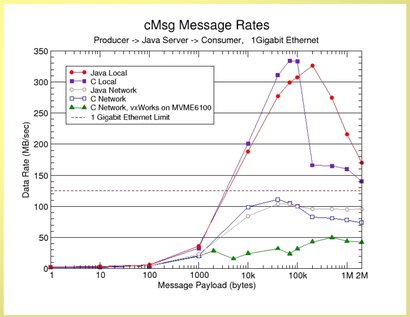
a Proxy Server Containing a Full Publish/Subscribe Implementation

Performance

For small messages, rates over the network between 1 producer and 1 consumer are over 20kHz (100x above our design spec).



For large messages (>10K bytes), data rates over the network between 1 producer and 1 consumer reach 80% of the network bandwidth.



Downloads

Download and give cMsg a try! You can get your free copy today at ftp://ftp.jlab.org/pub/coda/cMsg

More features are currently being added so stay tuned. Contact:

Carl Timmer, (757) 269-5130, timmer@jlab.org, or

Elliott Wolin, (757) 269-7365, wolin@jlab.org

Conclusions

The cMsg system is a simple, powerful, and flexible open-source framework within which one can deploy multiple underlying IPC systems. It includes a built-in, full-featured, asynchronous publish/subscribe component, support for a number of commonly used IPC systems, as well as a number of useful utilities. It supports C, C++ and Java clients, and runs on Unix and vxWorks.

cMsg performance approaches network bandwidth limits, and generally is only limited by the networking ability of the server machine. Indeed it exceeds our requirements by two orders of magnitude.

The use of Java in cMsg greatly reduced our development time compared to C, and Java performance has proven to be excellent, generally exceeding C performance (although this may change with further tuning of the C components). Our results clearly demonstrate that Java is a serious contender for almost any DAQ or online requirement.