CMSG – A PUBLISH/SUBSCRIBE INTERPROCESS COMMUNICATION PACKAGE

Elliott Wolin

PCaPAC 2008

Ljubljana, Slovenia

Outline

1. Introduction
2. What is Publish/Subscribe Messaging
3. What is cMsg
   a) Client view
   b) Developer view
   c) Performance
4. Conclusions

Jefferson Lab: Newport News, Virginia

6 GeV Continuous Electron Beam Accelerator Facility

Superconducting RFQ's

Three existing experimental halls

Approved 12 GeV upgrade and new hall $310M

GlueX experiment:

200 kHz trigger

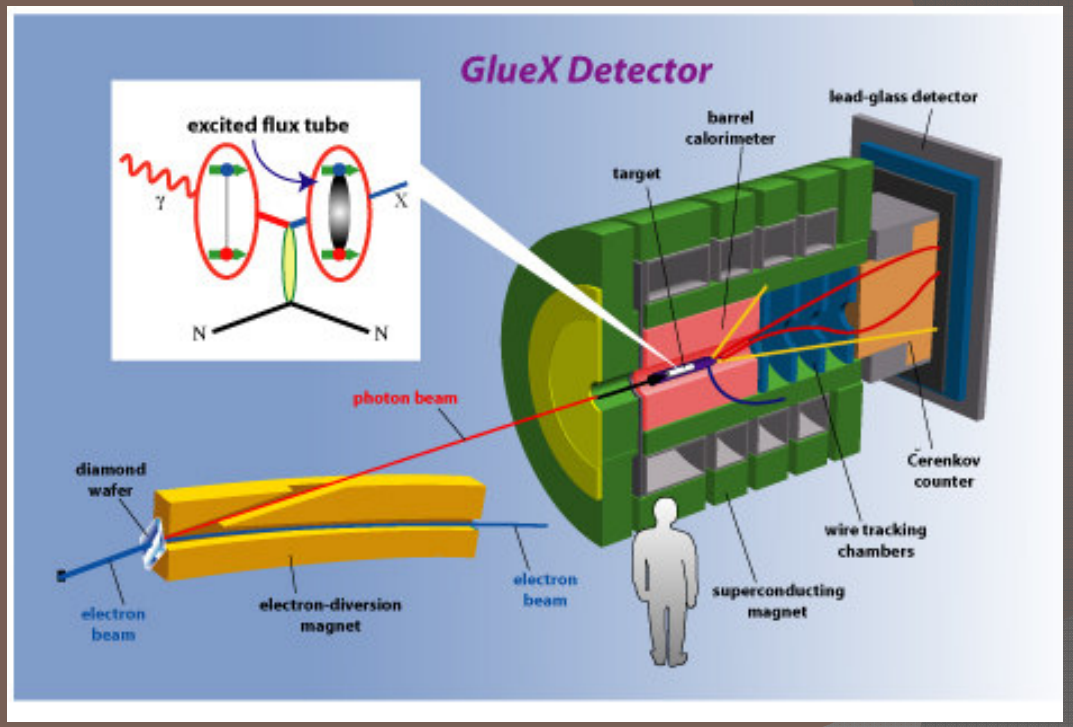
3 GB/s off detector

300 MB/s to tape

GlueX Experiment

Search for mesons with gluonic excitations

- 200 kHz trigger rate
- Deadtimeless readout
- 15 kB event size
- 3 GB/sec to L3 farm
- Factor 10 L3 rejection
- 300 MB/s to tape

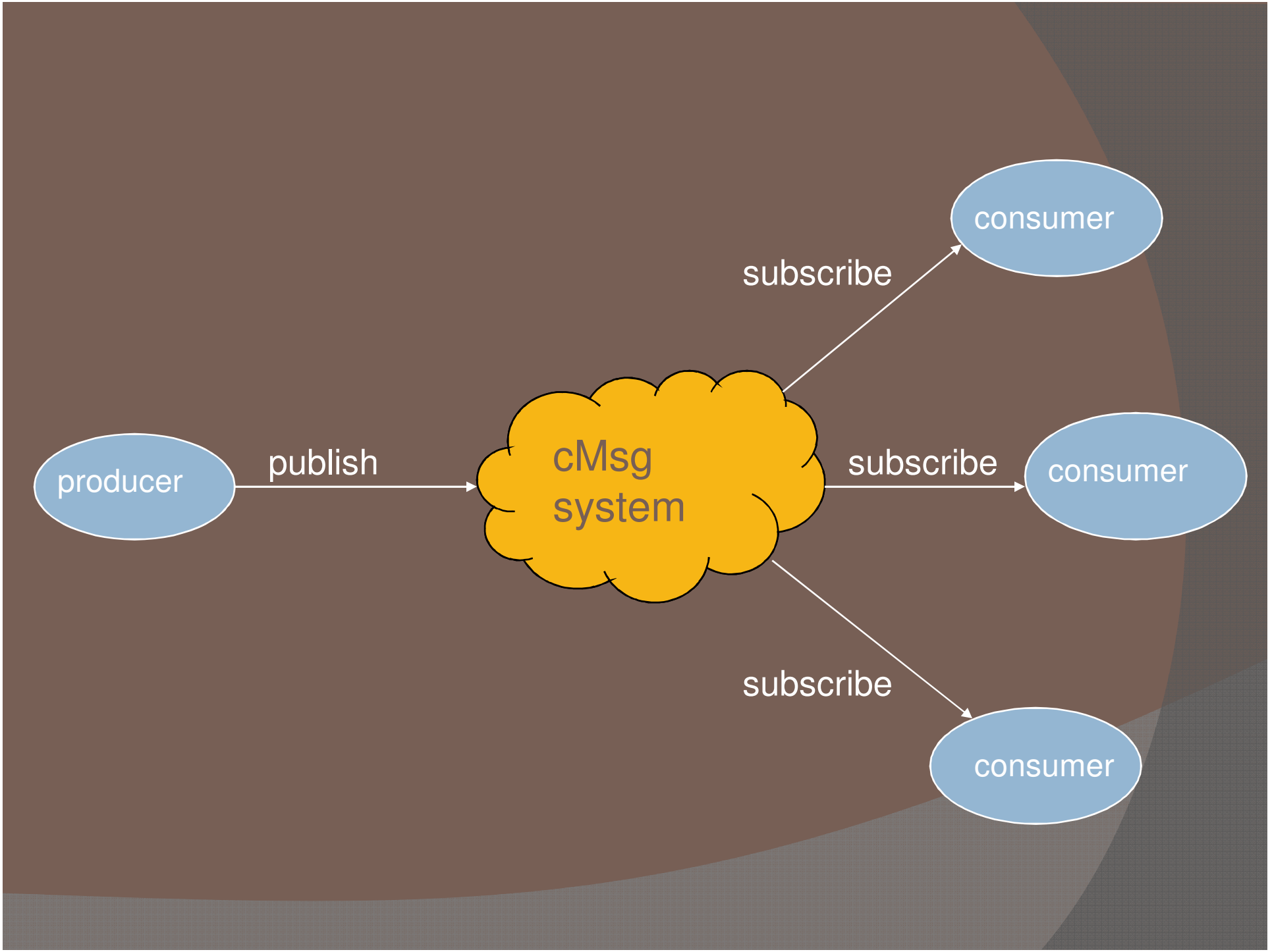


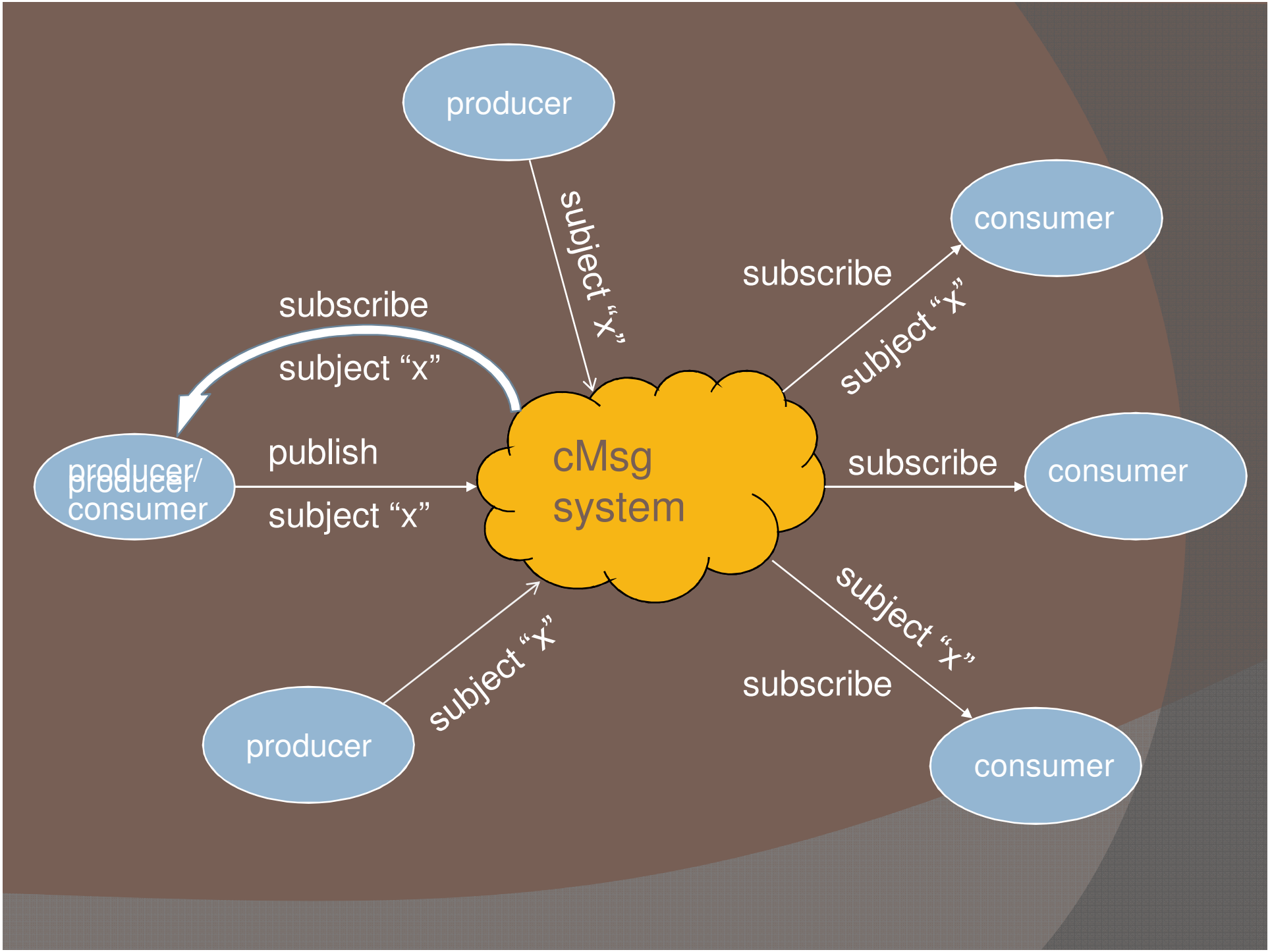
2. What is Publish/Subscribe Messaging

- Asynchronous, distributed, location transparency
  - Distinct from client/server – no notion of "service" or "service provider"
- Producers: publish or send messages to "subjects"
  - "launch-and-forget" mode
  - many producers can publish to the same subject
- Consumers: subscribe to subjects and supply callbacks
  - "subscribe-and-forget" mode
  - many consumers can subscribe to the same subject
- A process can be both a consumer and producer
  - a process can even receive messages it produces!

→ **Deceptively simple, but very powerful!** ←

See Wikipedia article under "Publish/Subscribe"


consumer
subscribe
producer
publish
cMsg
system
subscribe
consumer
subscribe
consumer


producer
subject "x"
consumer
subscribe
subscribe
subject "x"
subject "x"
publish
producer/
consumer
cMsg
system
subscribe
consumer
subject "x"
subject "x"
subscribe
producer
consumer

3. What is cMsg – client view

A complete publish/subscribe messaging system

- Very simple API (unlike CORBA and others)
- C/C++, Java
- Many Unix flavors, VxWorks
- Has some synchronous capabilities
- Highly customizable and extendable

→ Can satisfy all your messaging needs! ←

To send a cMsg message

```
#include <cMsg.hxx>

// connect to system via Universal Domain Locator
cMsg c(UDL, “myName”, “My description”);
c.connect();

// create and fill message
cMsgMessage msg;
msg.setSubject(“mySubject”);
msg.setType(“myType”);
msg.setText(“This is my text”);


// send message
c.send(msg);
```

To receive a cMsg message

```
#include <cMsg.hxx>

// connect to cMsg system
cMsg c(UDL,  “myName”, “My description”);
c.connect();

//  subscribe to subject/type combination and start receiving
c.subscribe(“mySubject”, “myType”, new myCallback(), NULL);
c.start();

    //  do something else…
```

Where callback is:

```
class myCallback : public cMsgCallback {
    void callback(cMsgMessage *msg, void* userArg) {
        cout << “Message subject is:   " << msg->getSubject() << endl;
    }
};
```

3. What is cMsg – developer view

- Framework for deploying multiple underlying IPC packages
  - plug in “domain” handlers at client level
- Proxy server facility
  - Plug in “subdomain” handlers at server level
  - Server written in Java
- UDL specifies domain, server, and subdomain

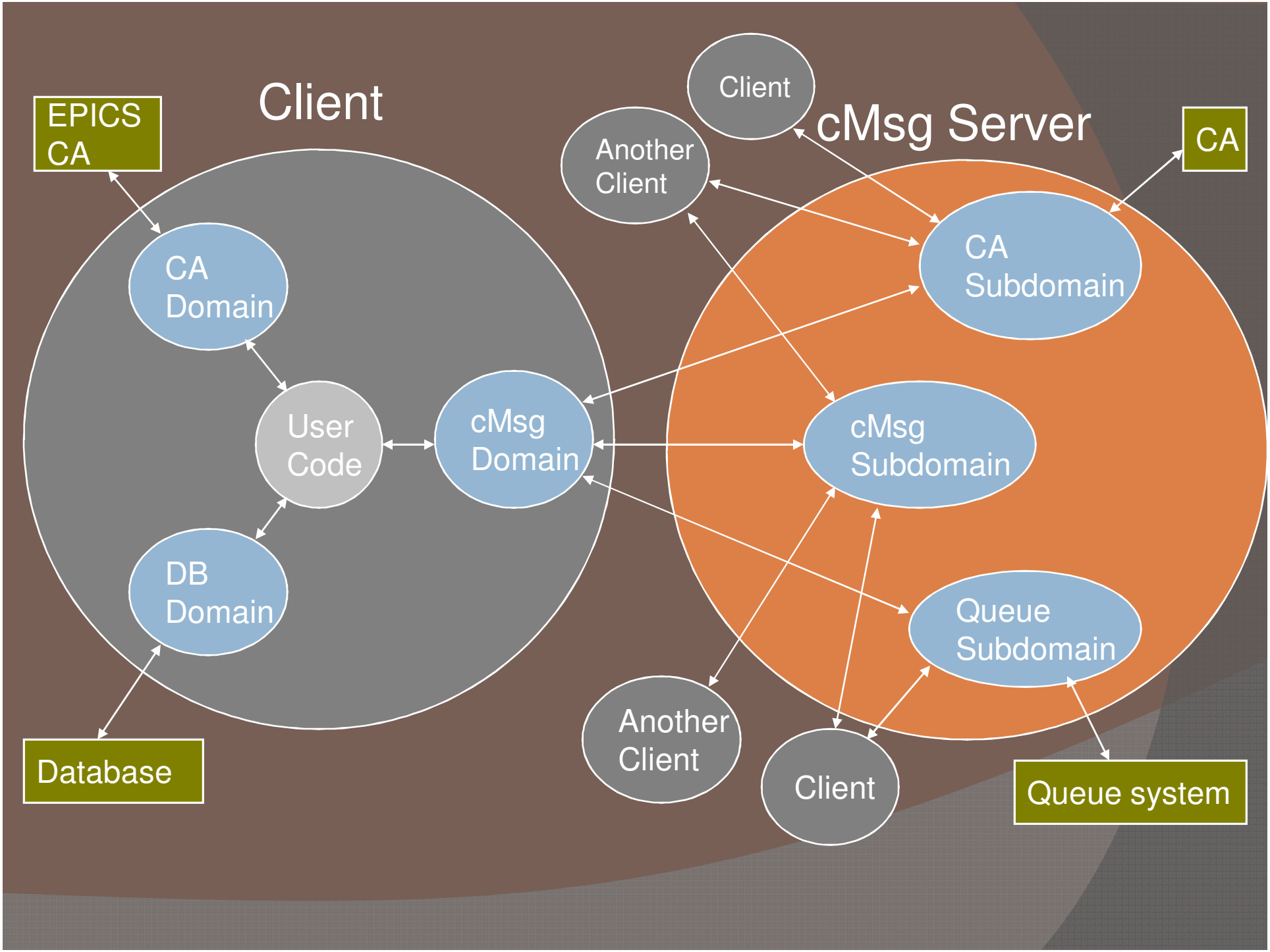
- Domains
  - Implemented in client
  - Java, C/C++
  - File domain – write to a file
  - CA domain – Channel Access (get/put/monitorOn/Off)
  - cMsg domain – connect to proxy server
    - Here message is transported to server and handed off to subdomain handler within server
  - Others, easy to write new ones

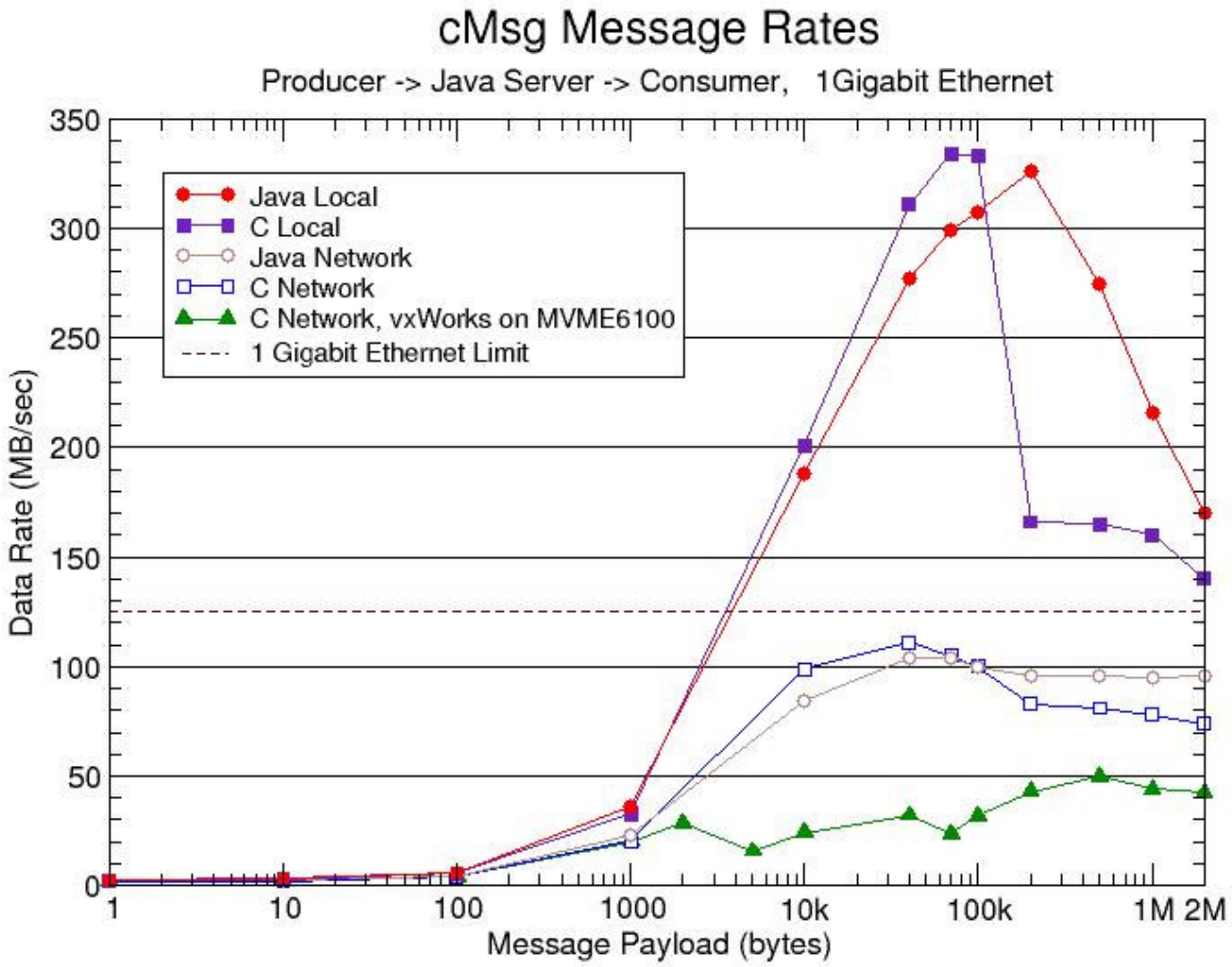
- Subdomains
  - Implemented in cMsg domain server
  - Java only

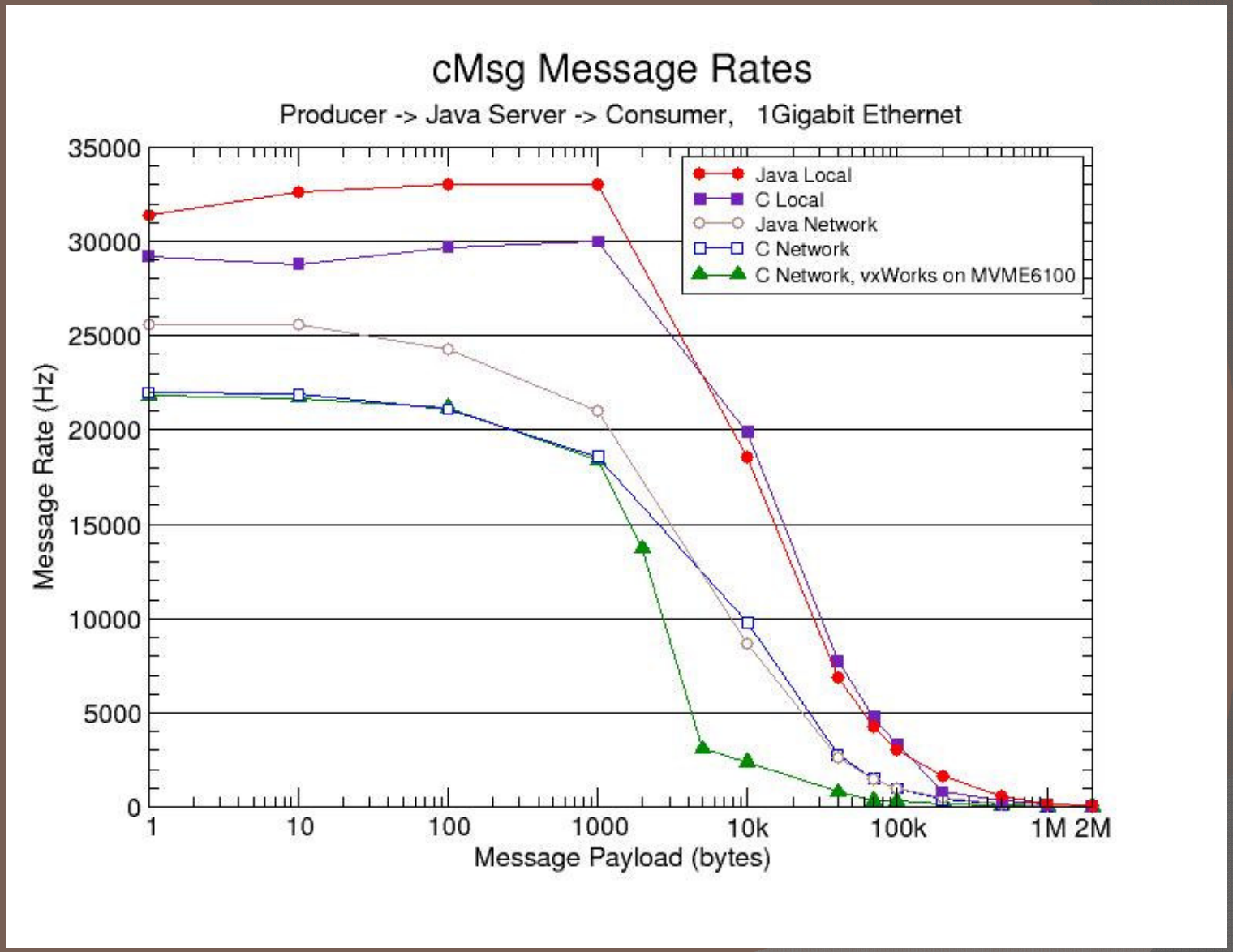
  - LogFile – many processes write to common file
  - CA – Channel Access (get/put/monitorOn/Off)
  - Database – execute database commands
  - Queue – read/write to database-based queue

  - cMsg – full pub/sub + synchronous IPC
    - UDL: cMsg://host:port/cMsg/namespace

  - Others, easy to write new ones


Client
EPICS CA
CA Domain
User Code
cMsg Domain
DB Domain
Database
Another Client
Client
cMsg Server
CA
CA Subdomain
cMsg Subdomain
Queue Subdomain
Another Client
Client
Queue system


cMsg Message Rates
Producer -> Java Server -> Consumer, 1Gigabit Ethernet
Java Local
C Local
Java Network
C Network
C Network, vxWorks on MVME6100
1 Gigabit Ethernet Limit
Data Rate (MB/sec)
350
300
250
200
150
100
50
0
1
10
100
1000
10k
100k
1M 2M
Message Payload (bytes)


cMsg Message Rates
Producer -> Java Server -> Consumer, 1Gigabit Ethernet
Java Local
C Local
Java Network
C Network
C Network, vxWorks on MVME6100
Message Rate (Hz)
35000
30000
25000
20000
15000
10000
5000
0
1
10
100
1000
10k
100k
1M 2M
Message Payload (bytes)

IV. Conclusions

- cMsg
  - Includes a complete publish/subscribe package
  - Simple C/C++ and Java API's
  - Highly customizable and extensible
  - Framework for deploying IPC systems
  - Powerful proxy server
  - Very good performance
  - Free

→ Download from: ftp://ftp.jlab.org/pub/coda/cMsg

Backup Slides

Universal Domain Locators

UDL specifies your messaging “world”

In general (in analogy to URL):

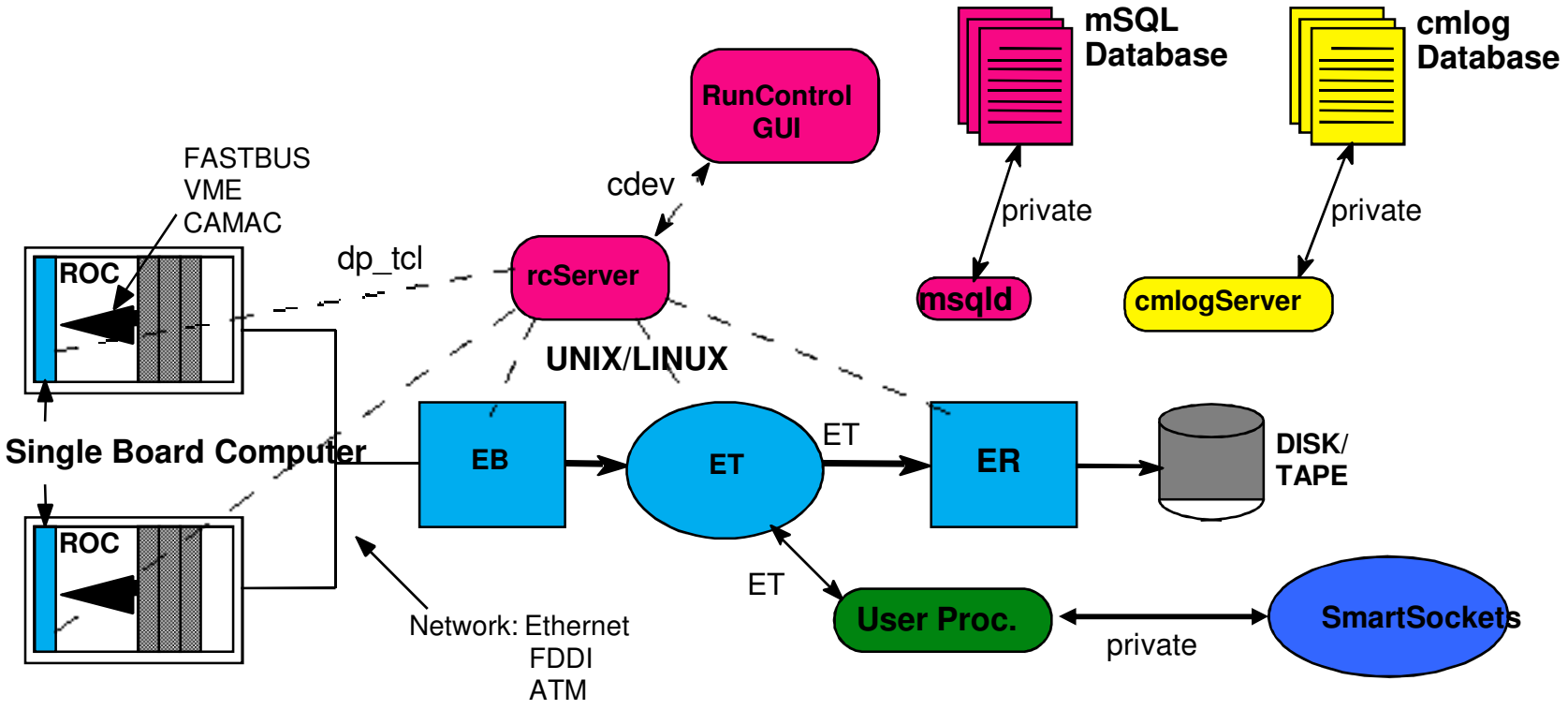
domainName://domainString?p1=v1&p2=v2&…

Examples:

```
FILE://myfile.txt                          // file domain
CA://channelName?addr_list=list            // channel access domain
```

Note: UDL is specified at runtime

IPC and Controls problem: too many protocols!

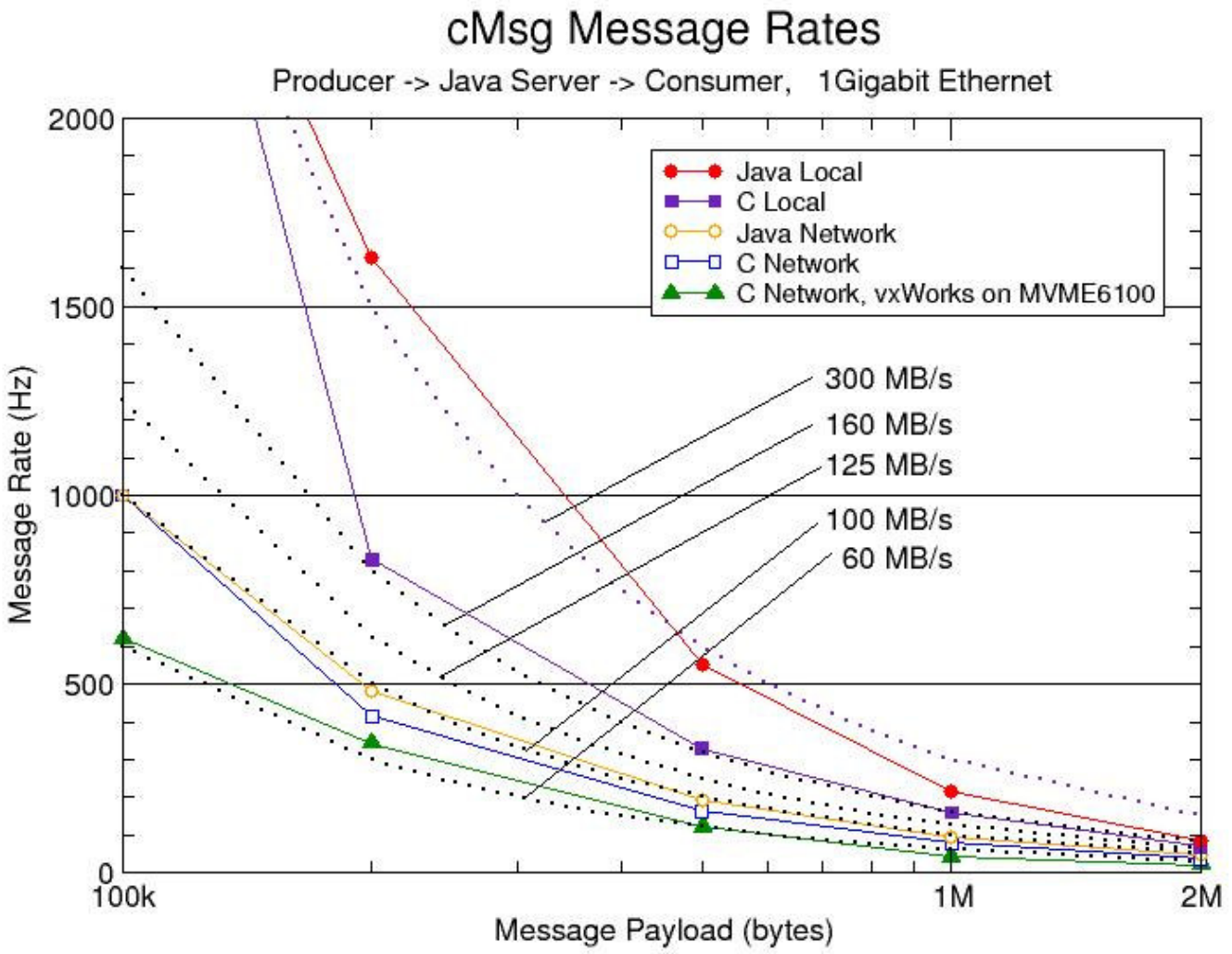


Synchronous features:

```
cMsgMessage response = c.sendAndGet(msg, myTimeout);

    // timeout exception thrown if no message arrives within timeout
```

- Utilities – all are short programs
  - cMsgLogger – subscribe, write msg to database/file/screen
  - cMsgQueue – subscribe, read/write to database or file system
  - cMsgGateway – subscribe to two domains and cross-post
  - cMsgCommand – read command-line parameters, send message
  - Others…


cMsg Message Rates
Producer -> Java Server -> Consumer, 1Gigabit Ethernet
Java Local
C Local
Java Network
C Network
C Network, vxWorks on MVME6100
300 MB/s
160 MB/s
125 MB/s
100 MB/s
60 MB/s
Message Rate (Hz)
2000
1500
1000
500
0
100k
1M
2M
Message Payload (bytes)