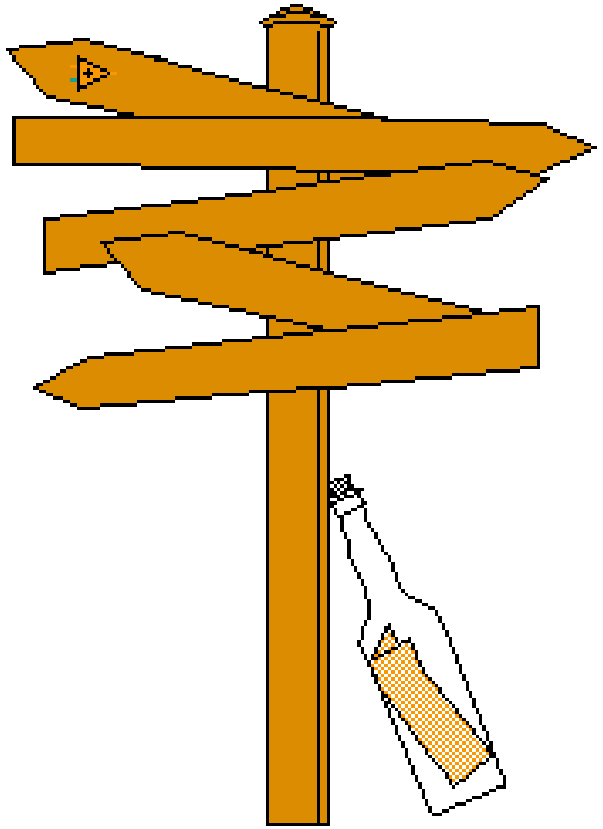
Distributed Information Management



DIM

"A Portable, Light Weight Package for Information Publishing, Data Transfer and Inter-process Communication"

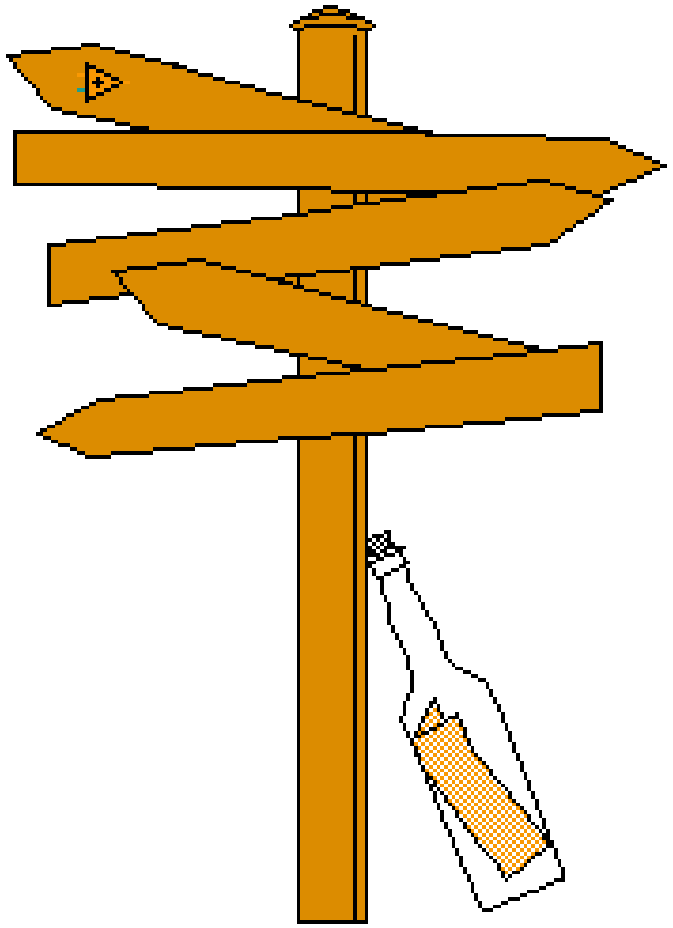
about DIM...

- triggered by control and data acquisition system of DELPHI @ CERN
- developed at CERN around 1992 by M. Dönszelmann, C. Gaspar and others
- copyright CERN, GPL licensed
- today:
  - CERN: widely accepted and used by LHC experiments
  - Babar
  - Sky Guide
- many slides of this presentation have been recuperated from talks by Clara Gaspar (thanks!)

What are our (KS/EE) typical problems?

- communication protocols bound to specific platforms
  - OPC, based on DCOM, limited to MS-Windows OS
  - Datasocket, proprietary by National Instruments, limited to platforms and programming languages supported by NI
  - ...
- communication protocol ABC has bad performance
- communication protocol DEF requires expensive licenses
- bad performance of TCP/IP implementation used by NI
- using TCP/IP on systems with many nodes
  - need to know name of node hosting process XYZ
  - need to handle multiple connections
  - what to do, if process XYZ migrates to another node?
  - what to do, if connection between process XYZ and UVW brakes?
  - byte order (Motorola ⇔ Intel), byte alignment, floating point representation
  - ...
- if process variables are static and can not be created on-the-fly, ...
- if number of process variables is limited, ...

Why are we (KS/EE) interested in DIM?



different possibilities in terms of

- communication: peer to peer
  - no central server/"event manager"
  - no intrinsic bottle neck
- platforms and operating systems
  - DIM only requires TCP/IP
  - (and "threads" or "signals")
- programming languages

simple and robust communication

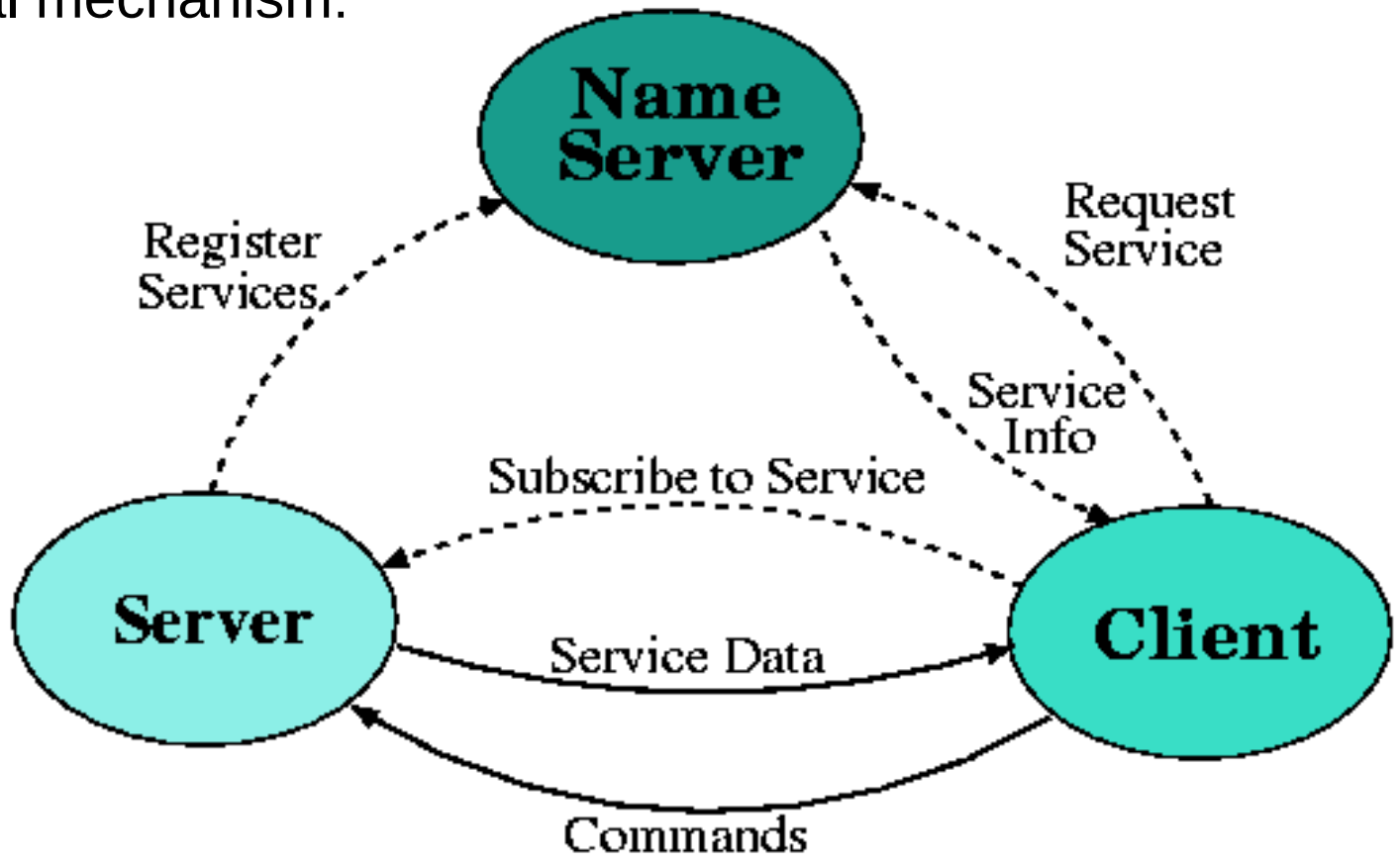
- based on pure TCP/IP (no real-time)
- DIM is "old": well established/debugged

DIM's philosophy

- Publish/Subscribe (client/server)
- Services (server → client)
  - One to many
  - Single items, arrays or structured data
  - Identified via a name
  - Free name space
  - Can be time-stamped (and have a quality flag)
- Command-Services (client → server)
  - One to one
  - Just like "Services", but
  - No time-stamp or quality flag
- Name Server
  - Keeps the coordinates of available (Command-)Services

DIM Dataflow

Internal mechanism:

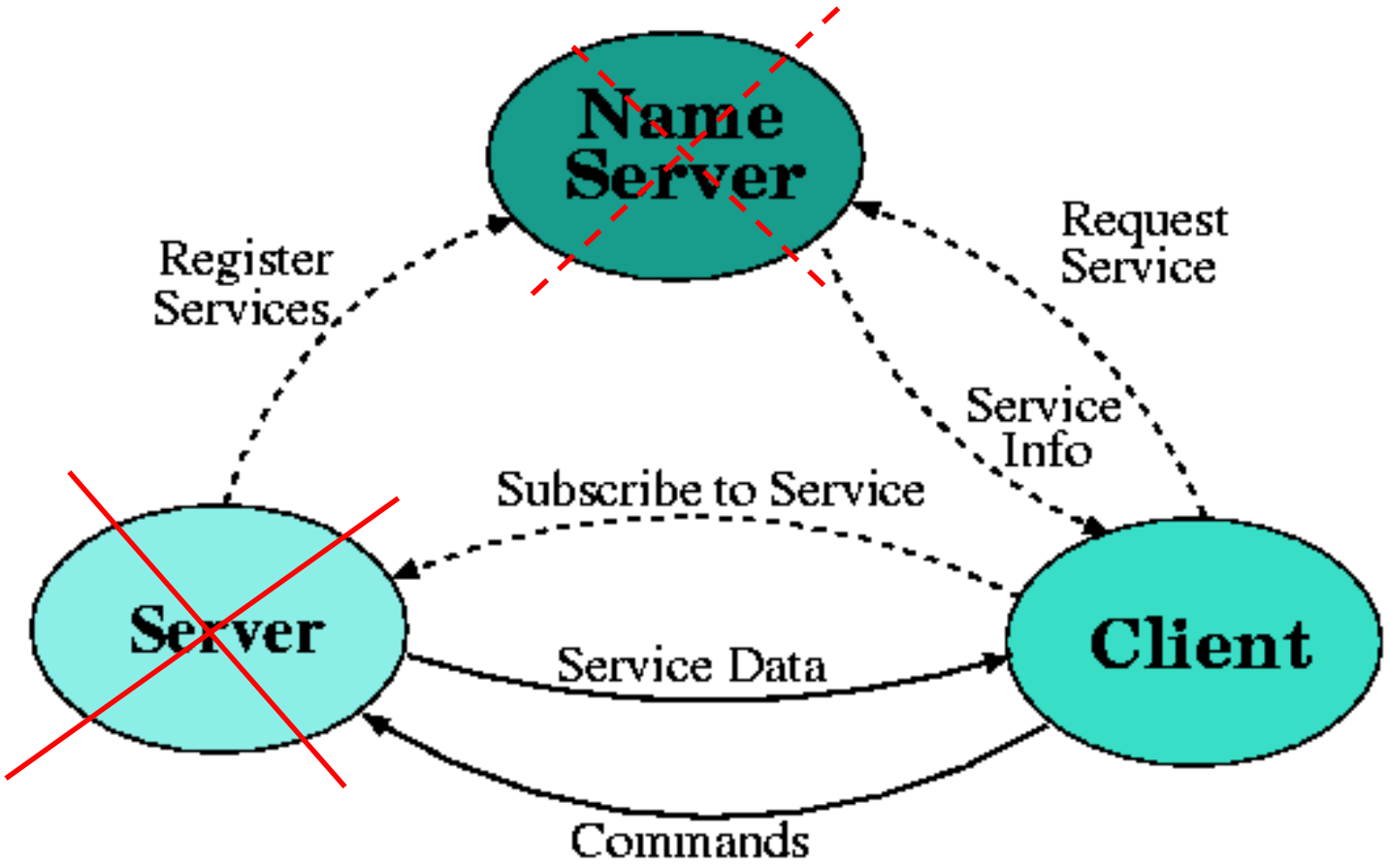


⇒ No node names, IP numbers, ... (except Name Server node)

DIM's functionality

- Publisher
  - Declares services
  - Can update services on change
  - DIM is multithreaded: can execute a "callback routine", when service needs to be updated
  - Multiple clients are handled inside DIM
- Subscriber
  - Can subscribe on change or on time basis or both.
  - DIM is multithreaded: Can execute a "callback routine" when new data arrives.
  - Will be informed when service is not available

Client Error Handling



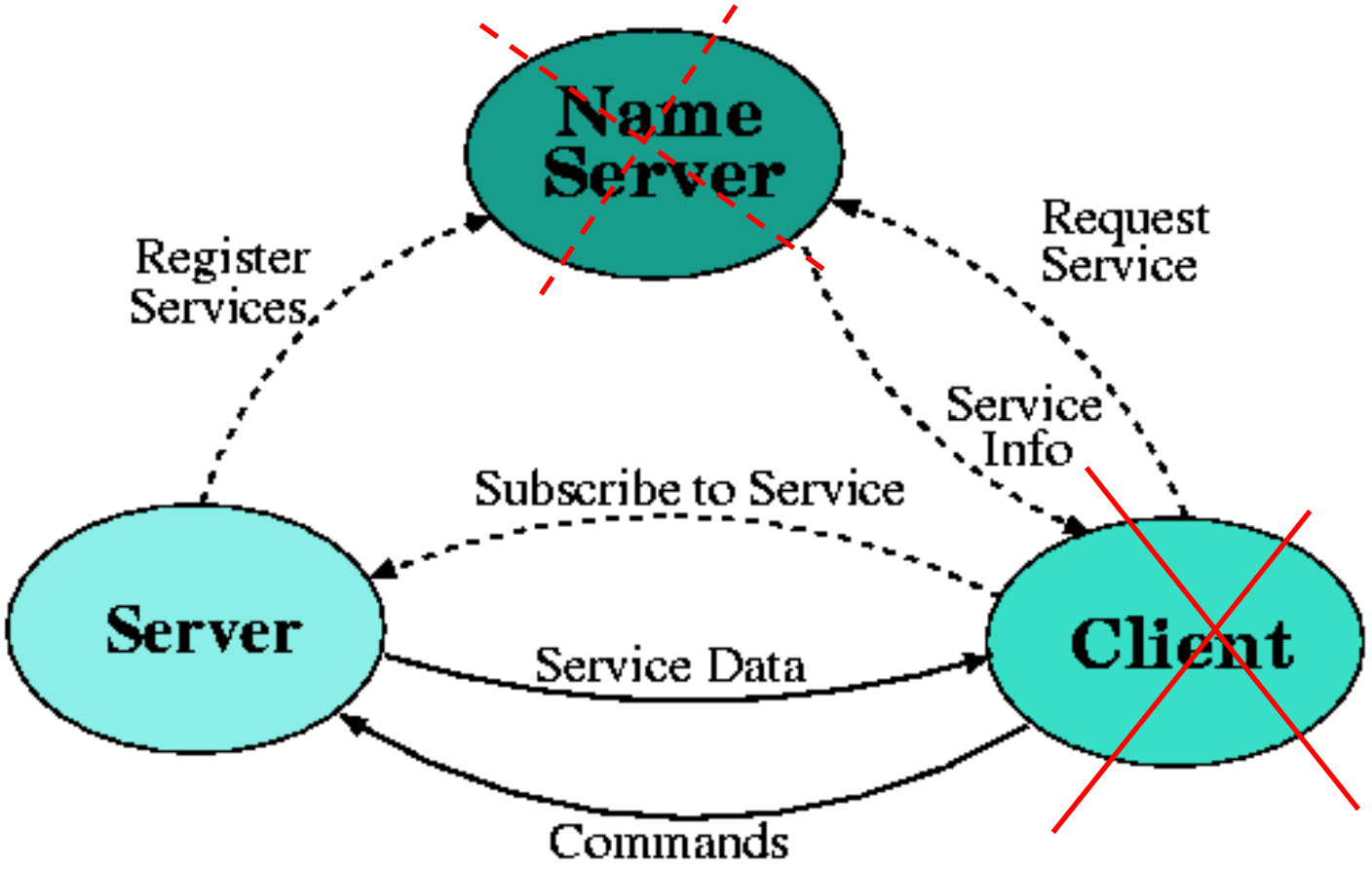
Name Server Error Handling ...

- Servers have a watchdog mechanism
  A hanging server will automatically be “removed” from the system
- A server trying to publish an existing service will be killed (service names must be unique). Note: DIM allows to specify error and exit routines for DIM servers.

Client Error Handling ...

- Server dead:
  - Client error notification (callback) in case of
    - Network problems
    - Server (or it's machine) down
  - Client error recovery – Automatic reconnection when
    - Server up (also if it moved to a new machine)
    - Name Server restart (also if it moved)
- Name Server dead: Any established connections will continue working without Name Server

Server Error Handling



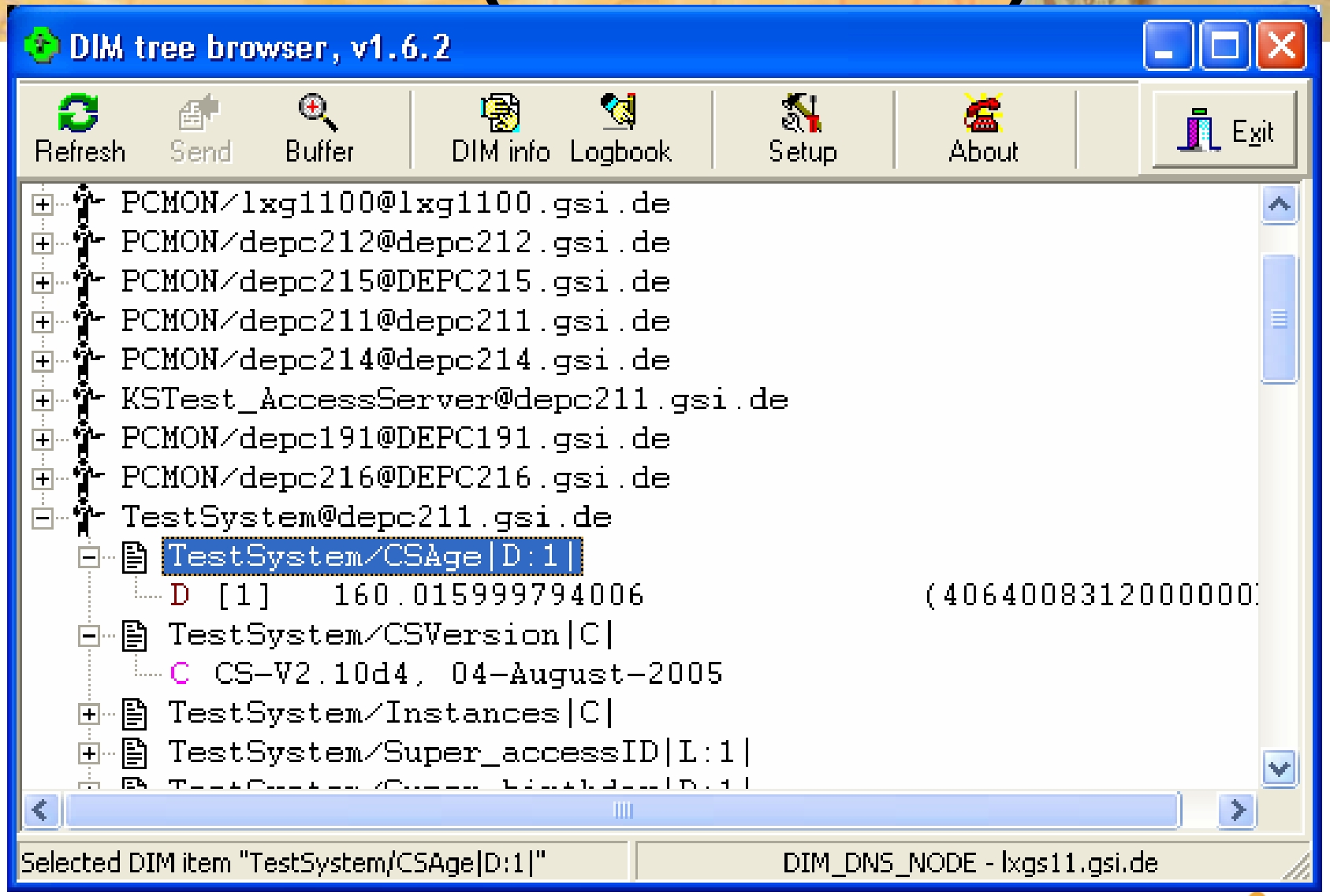
Server Error Handling and Recovery

- Client dead: Automatic disconnection from a “blocked” client (the client will automatically reconnect if it comes back)
- Name Server dead: Automatic (re)declaration of services when Name Server (re)starts

Dynamic Behavior Summary

- Servers
  - Can declare and un-declare services dynamically
  - Can be started/restarted anywhere anytime
- Clients
  - Can subscribe or un-subscribe dynamically
  - Will be informed in case of errors
- Name Server
  - Can be restarted (on a different machine)

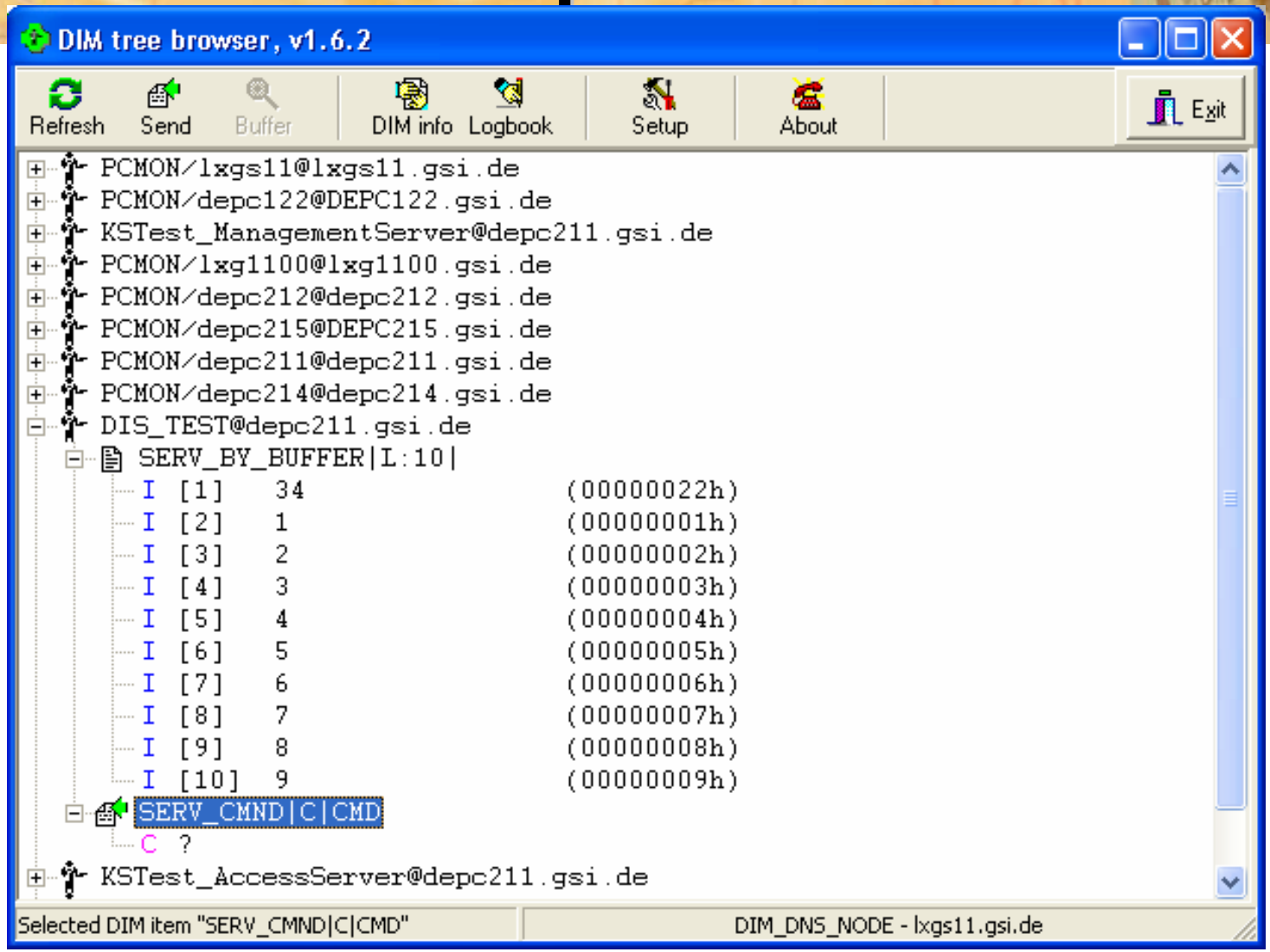
DIM Utilities... (here: DIMTree)



A Simple DIM Server

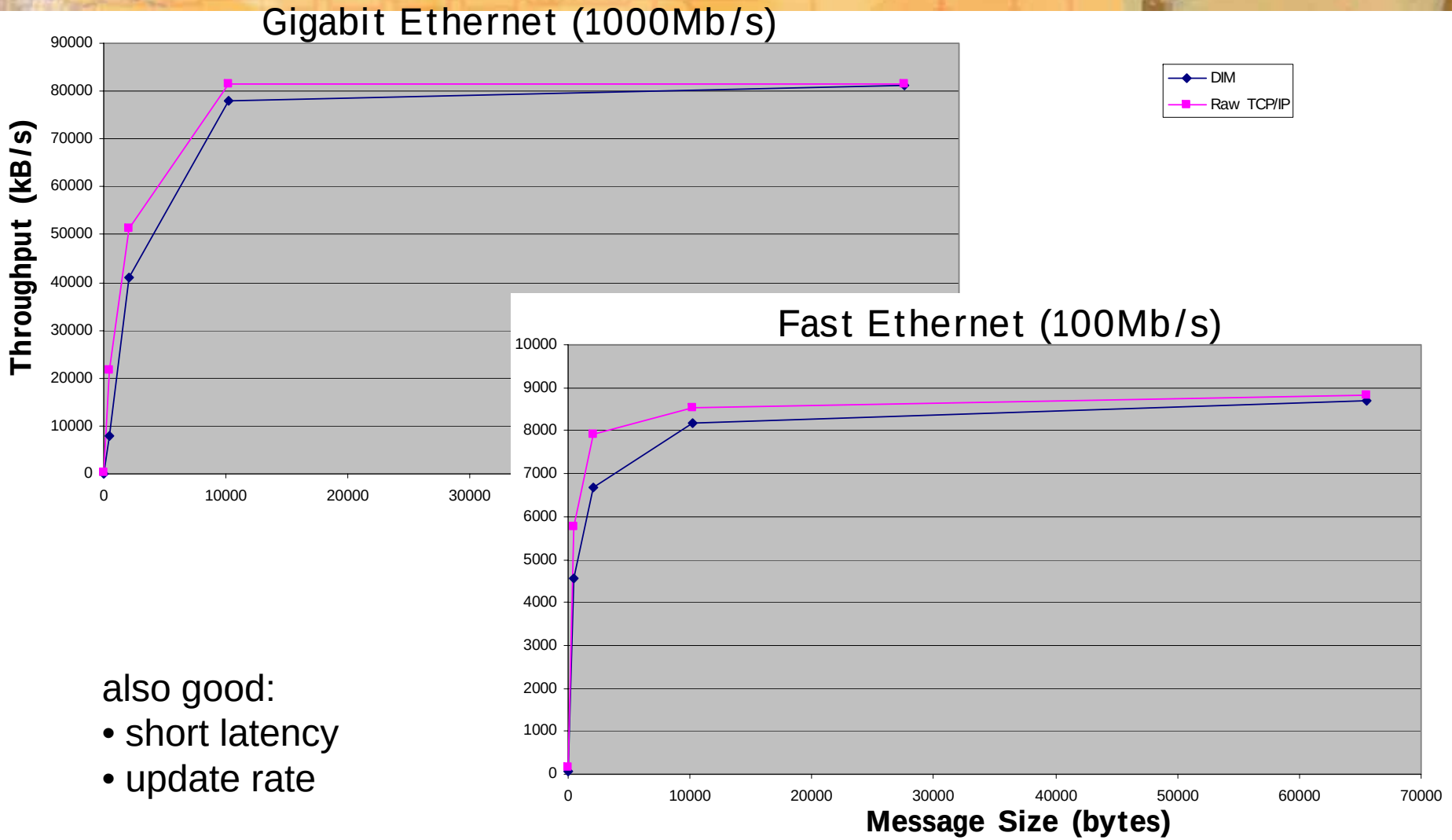
```
1 : #include <dis.h>                                                    // dll or shared library
2 :
3 : int buffer[] = {0,1,2,3,4,5,6,7,8,9};
4 : int service_id;
5 :
6 : void execute_command(int *tag, char *cmnd_buffer, int *size)       // callback function
7 : {
8 :    if (*tag == 1)
9 :    {
10:      buffer[0]++;
11:      dis_update_service(service_id);                               // direct access of memory!
12:    }
13: }
14:
15: void main()
16: {
17:    service_id = dis_add_service("SERV_BY_BUFFER","L:10",buffer,40,0,0); // declare service
18:    dis_add_cmnd ("SERV_CMND", "C", execute_command, 1);            // declare command service
19:    dis_start_serving("DIS_TEST");                                  // start my server
20:    while (1) sleep(10);                                            // multithreading via DIM!
21: }
```

"Demo" of Simple DIM Server



Note: Byte order and alignment as well as floating point representation handled by DIM!

DIM Performance (in 2002)



also good:
- short latency
- update rate

DIM is available under GPL

- As Server and Client libraries:
  - API: (Fortran,) C, C++ and Java (needs integration) at http://www.cern.ch/dim
- For the platforms:
  - VMS (VAX and APHA)
  - UNIX flavours (HP-UX, Sun-OS, Sun-Solaris, IBM-AIX, DEC-OSF, Linux)
  - Windows NT/2000/XP
  - Real-time OSs (OS9, LynxOS, VxWorks)
- LabVIEW interface (GSI) for WinXP and Linux

Concluding Remarks

- Performance
  - Throughput and update rate is almost the same as with pure TCP/IP (the Ethernet interface is the limit ☺)
  - Creating 1,000,000 DIM-Services (or "process variables") on-the-fly takes about 2 minutes.
- Usage
  - Very simple implementation of code
  - DIM takes care of lots of typical "problems" in distributed/mixed environment
  - DIM is robust and survives nasty things
- Light weight:
  - no security (like username and password), ...
  - fits on a credit card size computer
- Well Maintained, Excellent support (thanks a lot)
- The next *CS* release will be based on DIM