MATLAB, C/C++ communications and applications

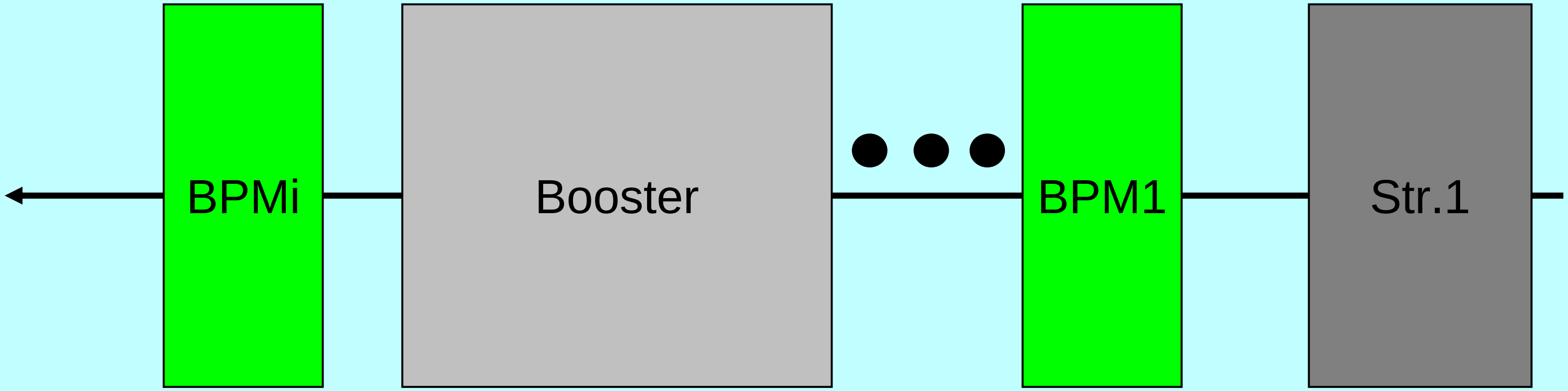
17 November 2011

Davit Kalantaryan

Content

1. Introduction
2. Why MATLAB ?
3. Why C/C++ functions needed ?
4. Integration of C/C++ functions to MATLAB
5. Additional tools created during work
6. System of parallel computing
7. Some examples

Introduction



The task is:

- To create functions, which are able to set simultaneously values of different properties of steerers and booster phase
- To read beam positions from BPMs
- To do some analysis
- Finally all of these functions must work in MATLAB

Why MATLAB ?

1. Multiplatform ● ● ●

2. Rich GUI

3. Some existing functions (already created, or existing in MATLAB)

4. Friendly Interface with many programs and programming languages

5. ?????????

Why are C/C++ functions necessary

Because MATLAB is a scripted language, it usually works slow, and sometimes it can be unacceptably slow.

1. So during steering process some functions can be called many times repetitively, and it is more preferable to have these functions binaries.
2. Absence of good interpretation of some tools. For example good interpretation of Hash tables, that is needed, for some functions.
3. Easy multithreading, specially in this task it is usually needed to set many values simultaneously.
4. If in the future will be necessary to create stand alone application most of functions will be ready.

Integration of C/C++ library functions into MATLAB

For using library functions from MATLAB, MEX file interface was used.

There is a MATLAB function "loadlibrary", that allows to load library in MATLAB. But this function normally works for 64 bit systems only for the MATLAB versions newer than R2008. I created mex function which allows to load libraries in 64 bit systems with old MATLAB versions.

So the new function for loading libraries in MATLAB is

```
function LoadLibraryX( LibraryName )
```

List of steering functions

1. succ = **LoadSteeringLib**(Ini_File_Name, varargin)
2. bIsExist = **IsSteeringLibraryLoaded**(varargin)
3. succ = **UnLoadSteeringLib**(varargin)
4. [output, varargout] = **boorespfun**(Phase0, PhaseF, Phase_Step, Steerer1Name, Prop1Name, SetPoint1, varargin)
5. [output, varargout] = **scalibur**(Steerer, Momentum, varargin)
6. outp = **sinfx**(Steerer)
7. succ = **sss**(varargin)
8. succ = **lss**(varargin)

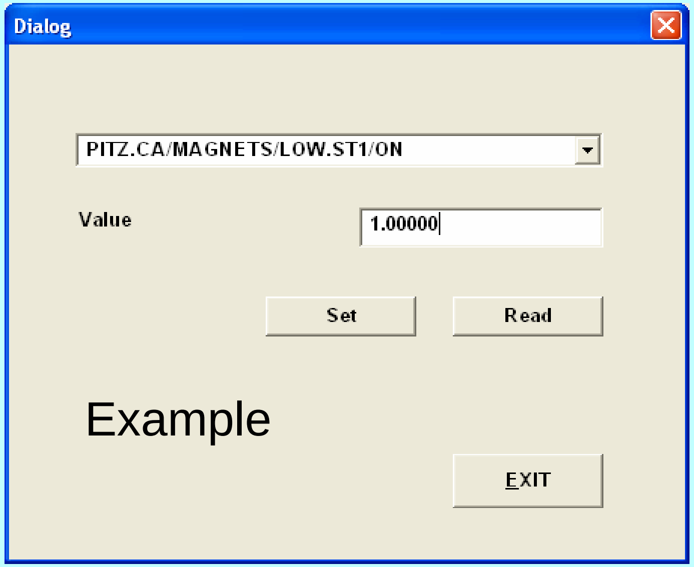
A manual for using these functions can be found in www-zeuthen.desy.de/~kalantar/public

Additional tools created during work

1. function CreateLibrary(Matlab_File)
2. function CreateThreadPoolX(Matlab_File, Threads_Number)
3. function LoadLibraryX(LibraryName)

www-zeuthen.desy.de/~kalantar/public

C Libraries for steering functions are libSteering64.so for Linux and libSteering32.dll for 32 bit Windows and libSteering64.dll for 64 bit Windows



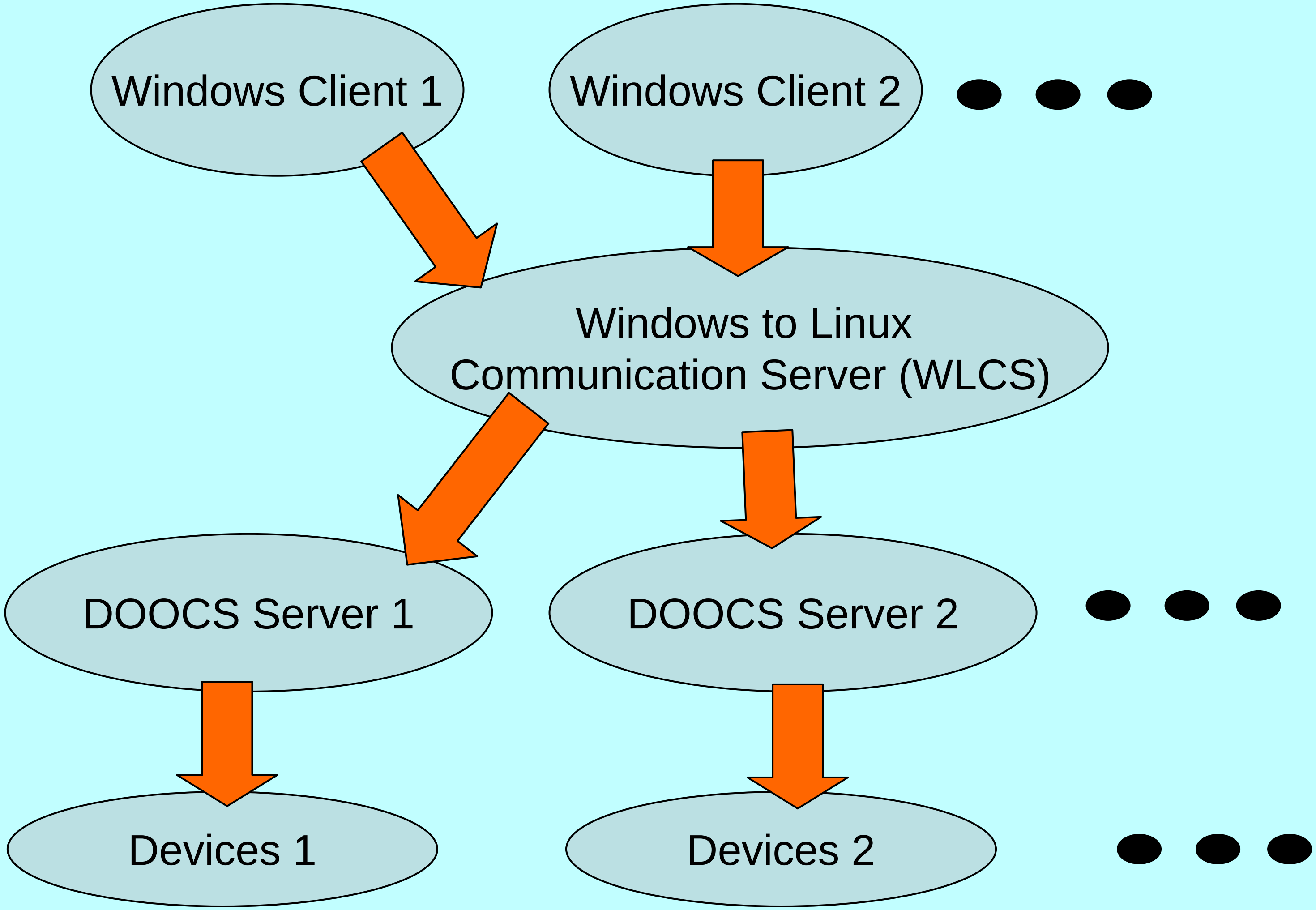
PITZ.CA/MAGNETS/LOW.ST1/ON

Value 1.00000

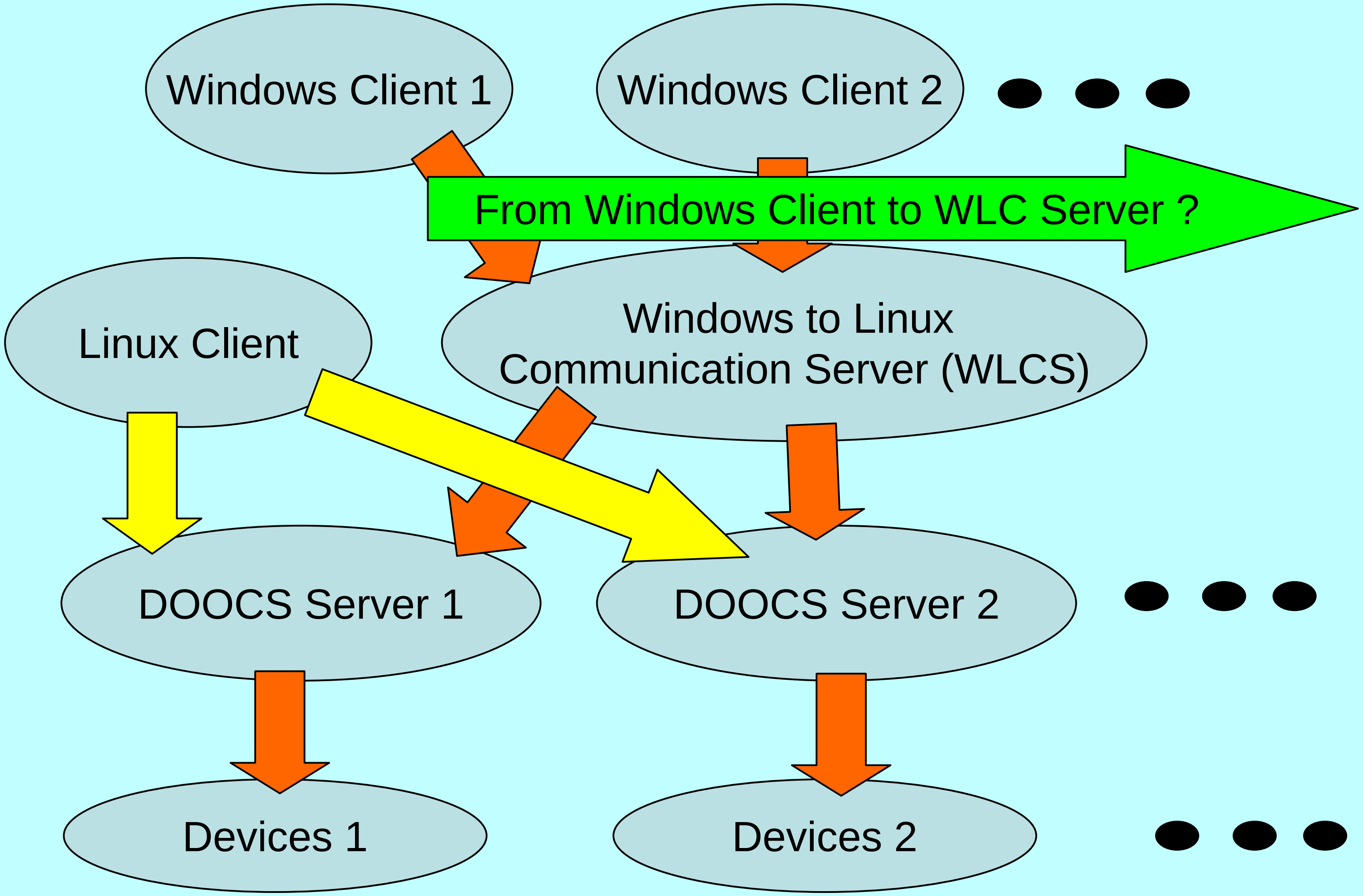
Example

All of these functions as well as reading and writing values from (into) DOOCS addresses are valid in Windows !

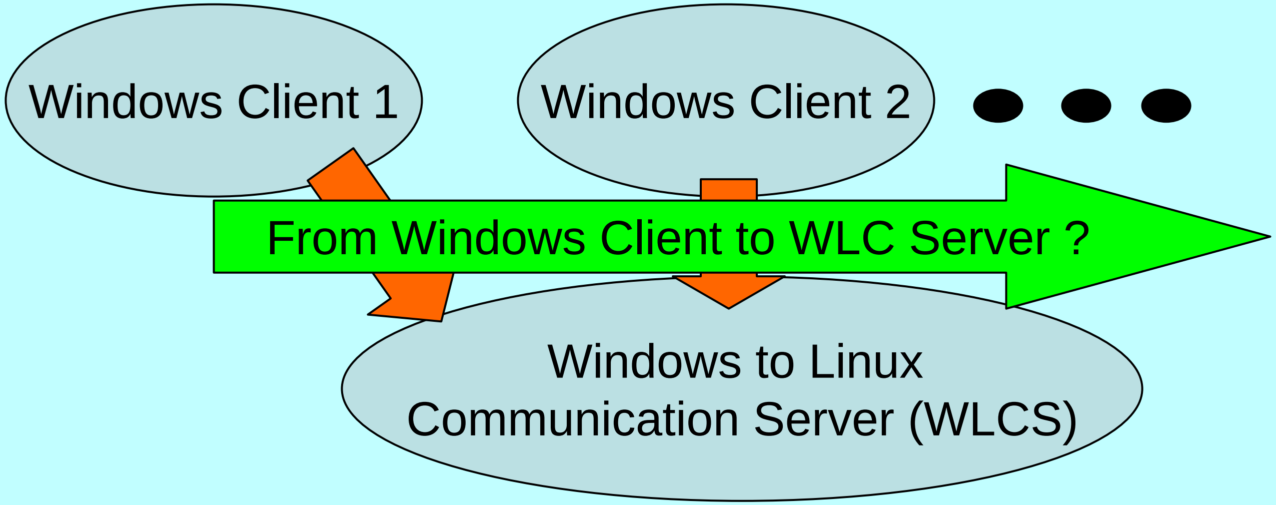
Scheme that allows to use DOOCS from Windows



Scheme that allows to use DOOCS from Windows



Steps of communicating with LWCS


Windows Client 1
Windows Client 2
● ● ●
From Windows Client to WLC Server ?
Windows to Linux
Communication Server (WLCS)

1. ~~Creating socket for communicating with LWC Server~~
2. ~~Sending an address~~ and in the case of setting set value to LWC Server
3. Receiving Error and in the case of Reading Value from LWC Server

Functions for reading from and writing into DOOCS address in Windows

```
int SetDOOCSFloat( int TimeOutPerByte, const char* Server_IP,
    const char* DOOCS_Address, float Value );
```

```
float GetDOOCSFloat( int TimeOutPerByte, const char* Server_IP,
    const char* DOOCS_Address, int*const Error );
```

- Error = -1 means that the connection to the server failed.
- Error with other codes means, that this error acquired during the setting or reading value in low levels.

These 2 functions are also in libraries DOOCS_Funcs_FW32.dll for 32 bit Windows and DOOCS_Funcs_FW64.dll for 64 bit Windows

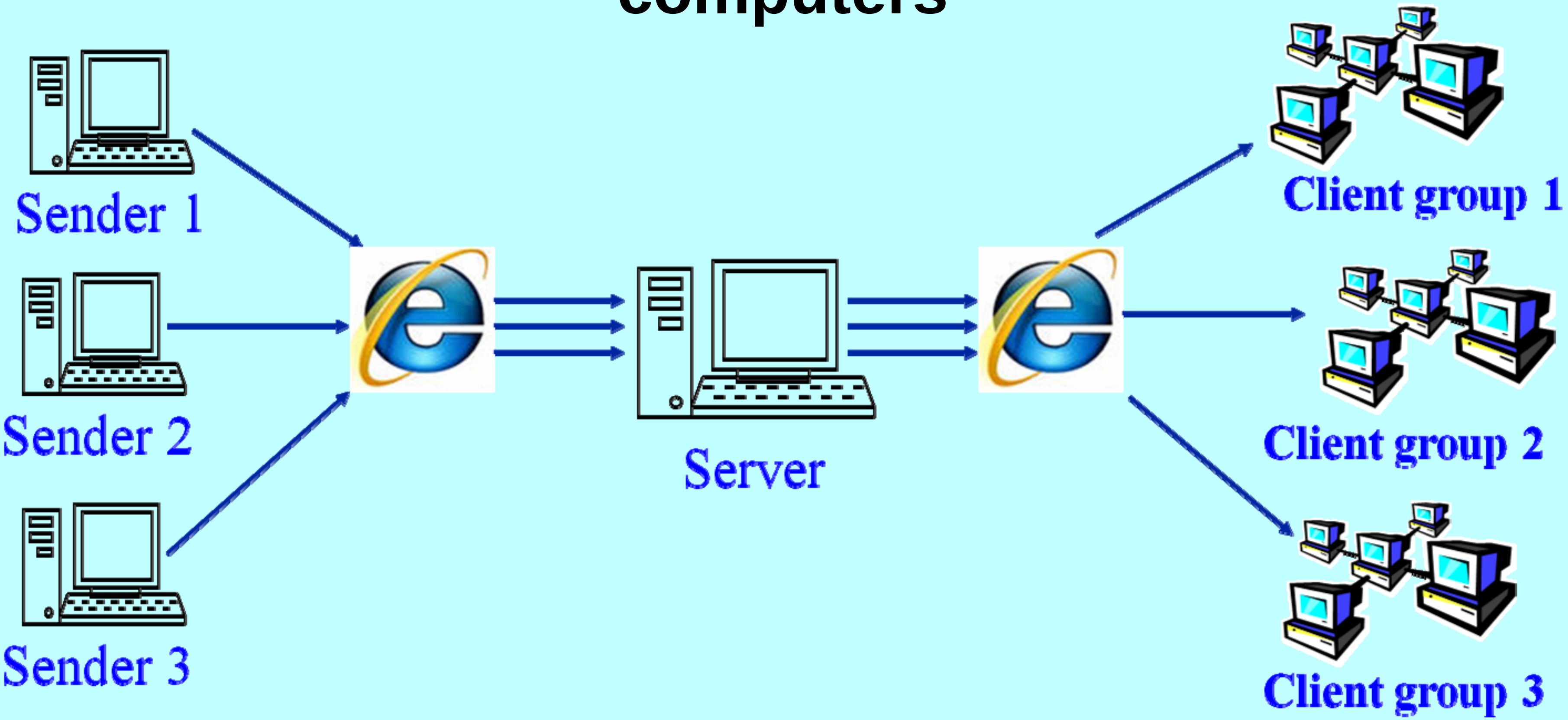
Functions after optimizing

1. ~~Creating socket for communicating with SLW Server~~
2. ~~Sending an address~~ and in the case of setting set value to WLC Server
3. Receiving Error and in the case of Reading Value from WLC Server

- int InitWindowsDOOCS(int TimeOutPerByte, const char* Server_IP);
- int SetDOOCSFloat(const char* DOOCS_Address, float Value);
- float GetDOOCSFloat(const char* DOOCS_Address, int& Error);
- int SetDOOCSFloat(int DOOCS_Address_Code, float Value);
- float GetDOOCSFloat(int DOOCS_Address_Code, int& Error);

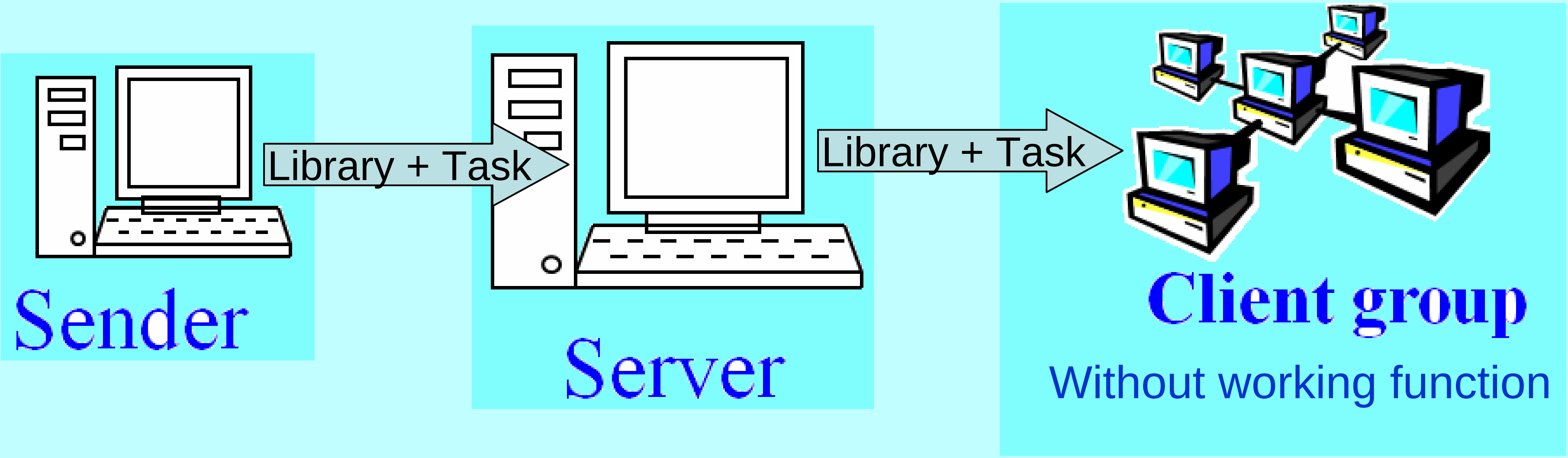
Libs: DOOCS_Funcs_FW_New32.dll and DOOCS_Funcs_FW_New64.dll

Diagram of a system for doing job using multiple computers



This system can be used from several programs and programming languages. For example this system can be used from MATLAB, also from C and C++

The ways to set working function in client side



Sender

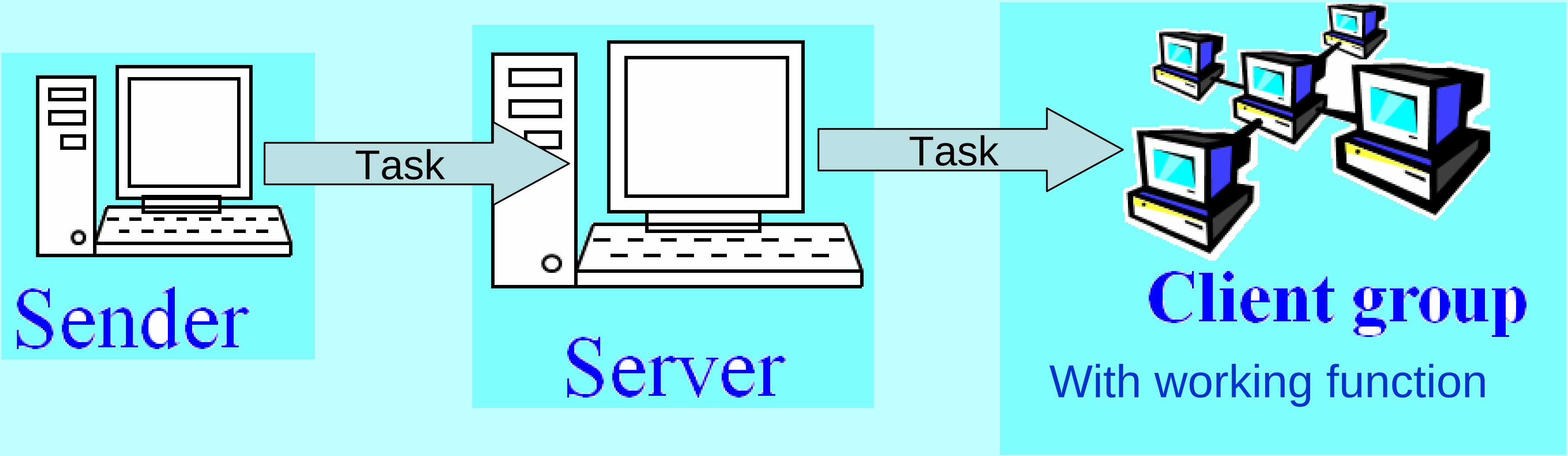
Library + Task

Server

Library + Task

Client group

Without working function



Sender

Task

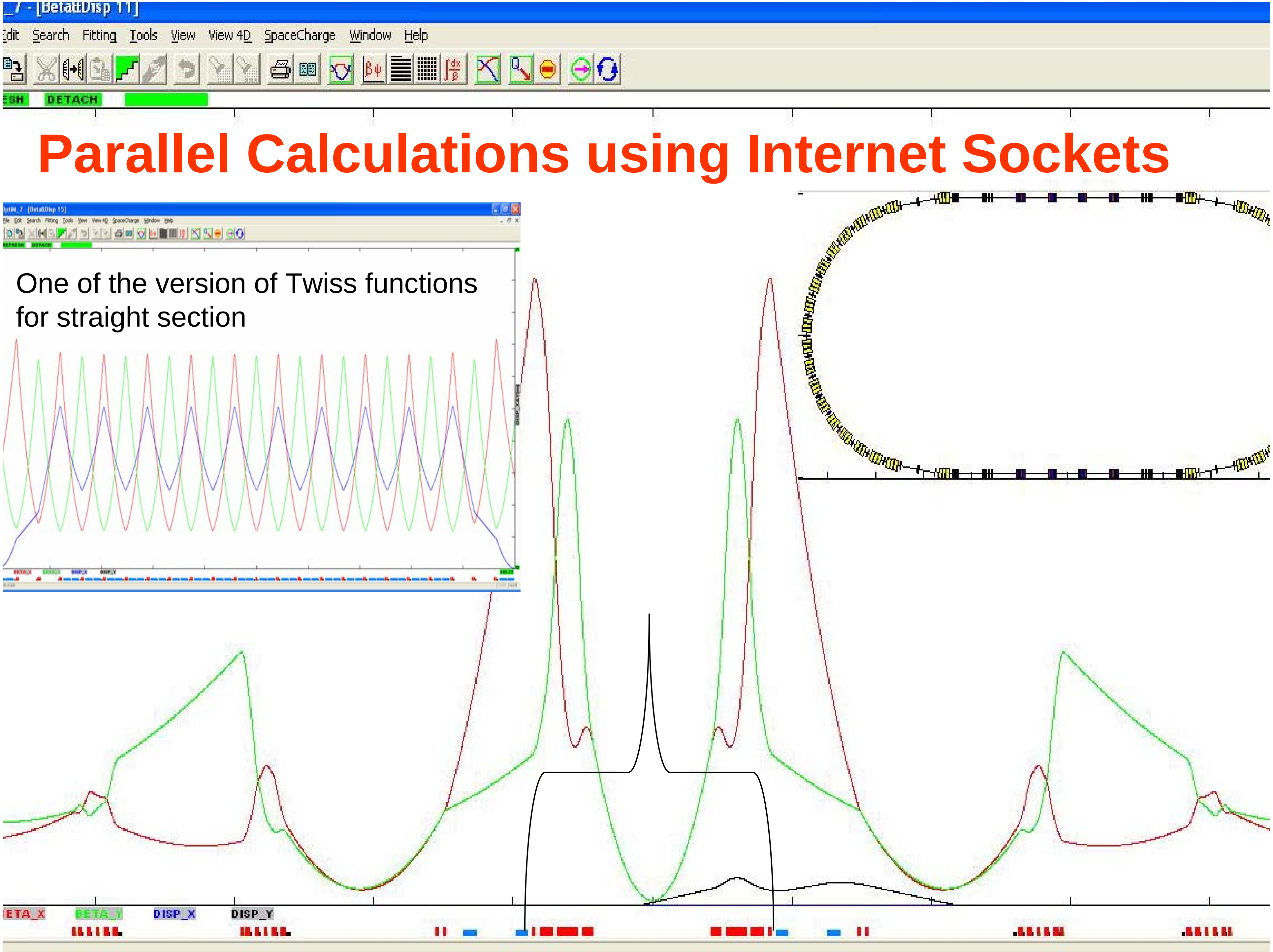
Server

Task

Client group

With working function

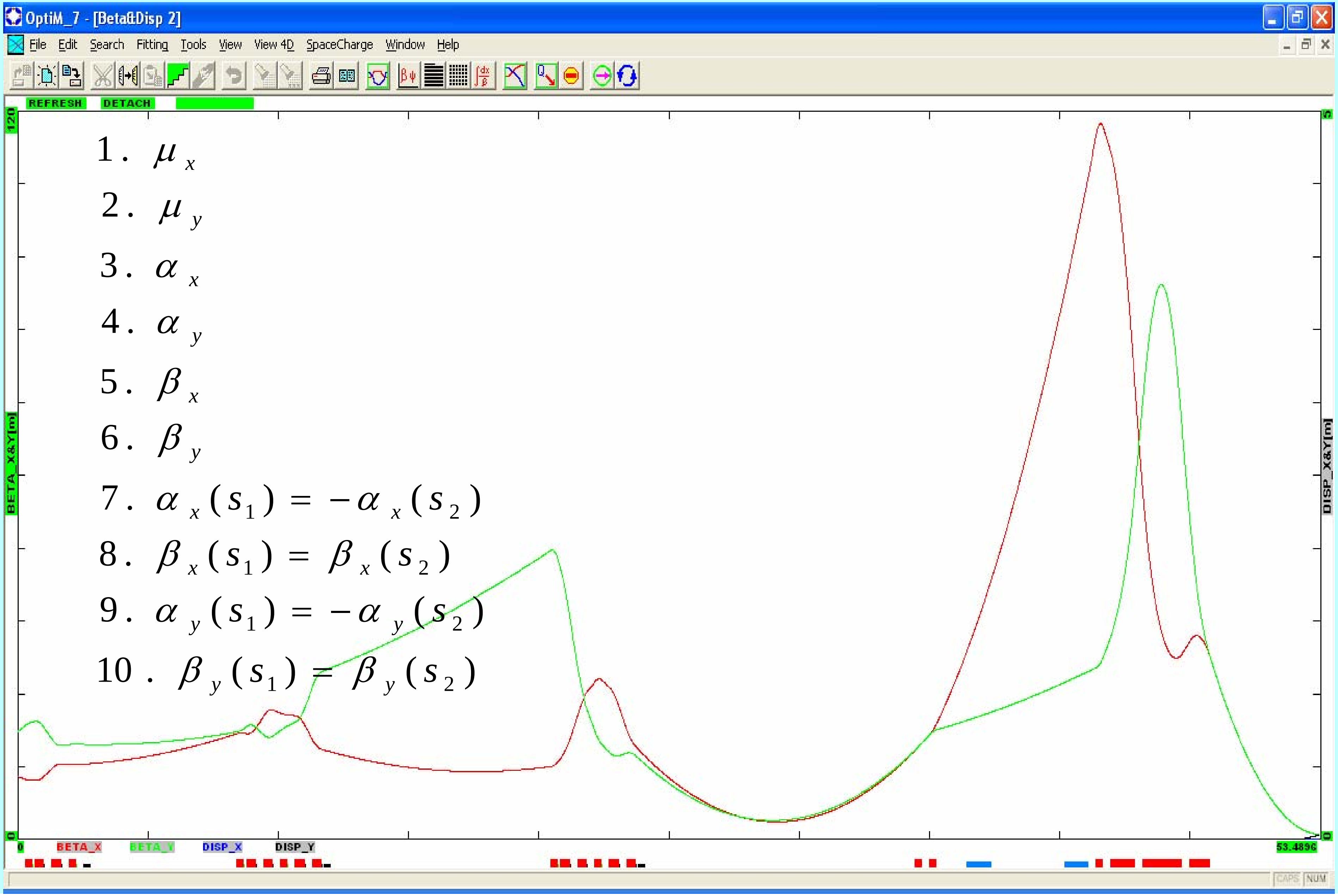
Parallel Calculations using Internet Sockets



One of the version of Twiss functions for straight section

_7 - [Beta&Disp 11]
Edit Search Fitting Tools View View 4D SpaceCharge Window Help
ESH
DETACH
ETA_X
BETA_Y
DISP_X
DISP_Y
OptiM_7 - [Beta&Disp 15]
File Edit Search Fitting Tools View View 4D SpaceCharge Window Help
REFRESH DETACH
BETA_X BETA_Y DISP_X DISP_Y
DISP_X&Y[m]

Parameters those need to reconcile



1. μ_x
2. μ_y
3. α_x
4. α_y
5. β_x
6. β_y
7. $\alpha_x(s_1) = -\alpha_x(s_2)$
8. $\beta_x(s_1) = \beta_x(s_2)$
9. $\alpha_y(s_1) = -\alpha_y(s_2)$
10. $\beta_y(s_1) = \beta_y(s_2)$

Thanks to Bagrat and Levon for helping to get familiar with DOOCS system

Thanks to Marek for helping to get involved to the problem

Thank you for your attention !