

Project 4: HTTP Server & CGI

NP TA 孝全

Important Dates

- **Deadline: 5/11 (Sun.) 23:59**
- **Demo: 5/12 (Mon.) 17:30 - 21:20**
 - We will announce demo slots 2 - 3 days before
 - Tasks:
 - Correct format and compile.
 - QA.
 - Pass test cases.
 - Implement 1 extra function with limit time.

Info

- You are **HIGHLY** encouraged to publish your questions on E3 Project4 forum
 - Check the spec and other questions first.
- You can contact the TAs by E3.
 - Mails sent to other addresses will **NOT** be replied
 - Please ensure that you email **ALL** the TAs.
- TA hours: **4/29, 5/6**(Tue.) 15:00 - 17:00
 - You **MUST** make a reservation by email in advance.
 - TAs will **NOT** debug for you.

Introduction

- The goal of this project is to build a **Remote Batch System** using **Boost.Asio**.
- This project is divided into 2 parts.
 - Part 1 should run on **NP Linux Workstation**.
 - Part 2 should run on **Windows 10/11**.
- The requirements in both parts are roughly the same with slightly different conditions.

Introduction - User view

來賓 NP Project 3 Panel

← 不安全 | nplinux12.cs.nycu.edu.tw:12345/panel.cgi

#	Host	Port	Input File
Session 1	nplinux12 ▾ .cs.nycu.edu.tw	12344	t1.txt ▾
Session 2	nplinux12 ▾ .cs.nycu.edu.tw	12343	t2.txt ▾
Session 3	▾ .cs.nycu.edu.tw		▾
Session 4	▾ .cs.nycu.edu.tw		▾
Session 5	▾ .cs.nycu.edu.tw		▾

Run

panel.cgi

Introduction - User view

```
nplinux12.cs.nycu.edu.tw:12344
*****
** Welcome to the information server. **
*****
*** User '(no name)' entered from 140.113.17.72:43978. ***
% printenv PATH
bin:.
% removetag test.html | cat

Test
This is a test program
for ras.

% removetag test.html

Test
This is a test program
for ras.
```

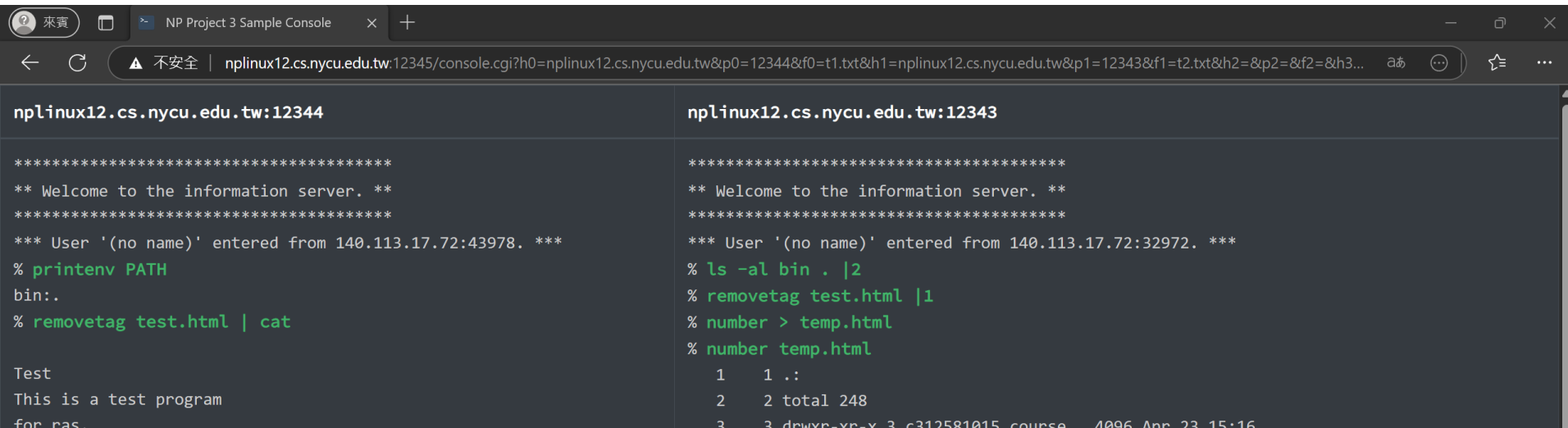
```
nplinux12.cs.nycu.edu.tw:12343
*****
** Welcome to the information server. **
*****
*** User '(no name)' entered from 140.113.17.72:32972. ***
% ls -al bin . |2
% removetag test.html |1
% number > temp.html
% number temp.html
1 1.:
2 2 total 248
3 3 drwxr-xr-x 3 c312581015 course 4096 Apr 23 15:16 .
4 4 drwxr-xr-x 5 c312581015 course 4096 Apr 23 15:00 ..
5 5 drwxr-xr-x 2 c312581015 course 4096 May 14 2024 bin
6 6 -rwxr-xr-x 1 c312581015 course 229752 May 13 2024 np_
7 7 -rw-r--r-- 1 c312581015 course 0 Apr 23 15:16 tem
8 8 -rw-r--r-- 1 c312581015 course 86 May 13 2024 tes
9 9
10 10 bin:
```

console.cgi

Part 1: Linux Version

Requirements for Part 1

- In part 1, you are asked to implement the following on the NP Linux Workstation
 - A simple HTTP server called **http_server**
 - A CGI program called **console.cgi**



```
nplinux12.cs.nycu.edu.tw:12344
*****
** Welcome to the information server. **
*****
*** User '(no name)' entered from 140.113.17.72:43978. ***
% printenv PATH
bin:.
% rmovetag test.html | cat

Test
This is a test program
for ras.

nplinux12.cs.nycu.edu.tw:12343
*****
** Welcome to the information server. **
*****
*** User '(no name)' entered from 140.113.17.72:32972. ***
% ls -al bin . | 2
% rmovetag test.html | 1
% number > temp.html
% number temp.html
1 1 .:
2 2 total 248
3 3 drwxr-xr-x 3 c312581015 course 4096 Apr 23 15:16
```


Run HTTP Server

client



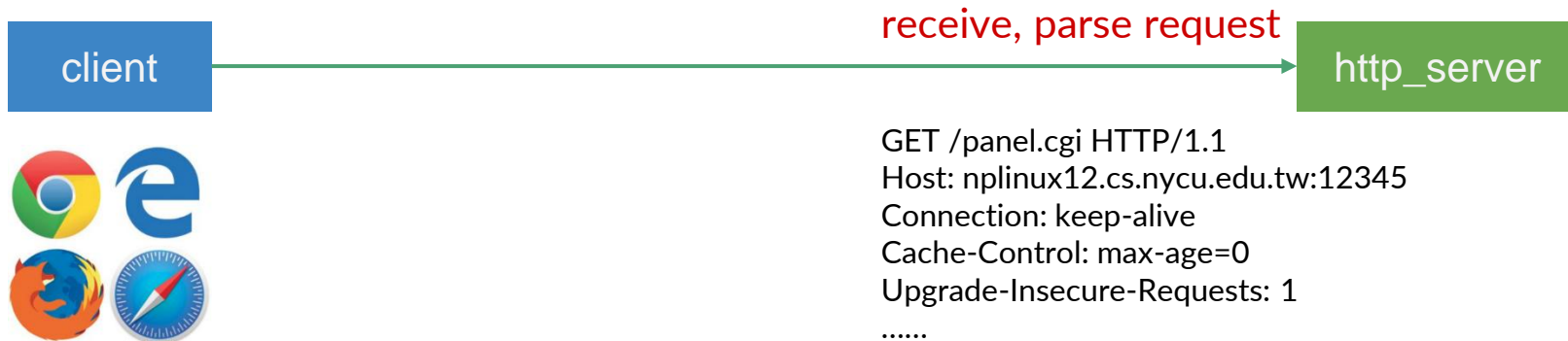
http_server

nplinux12
port: 12345

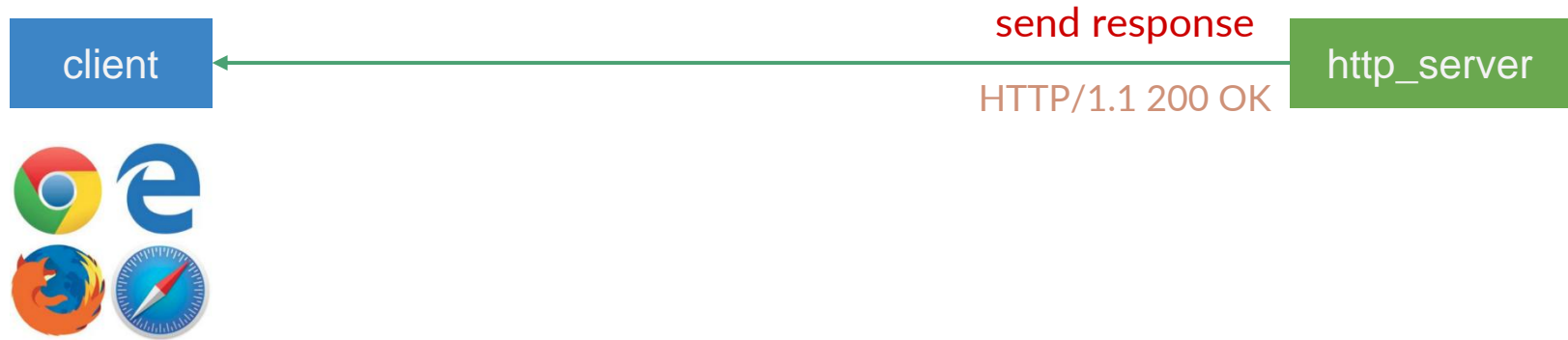
Request 1 - Send



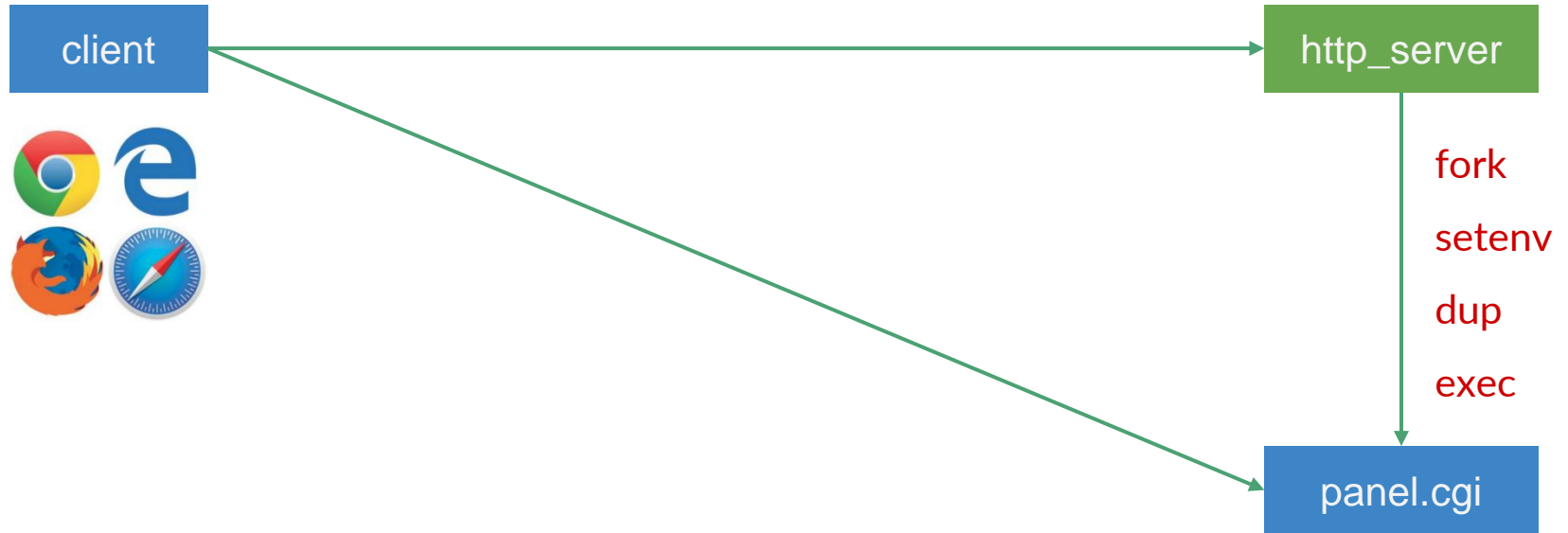
Request 1 - Receive



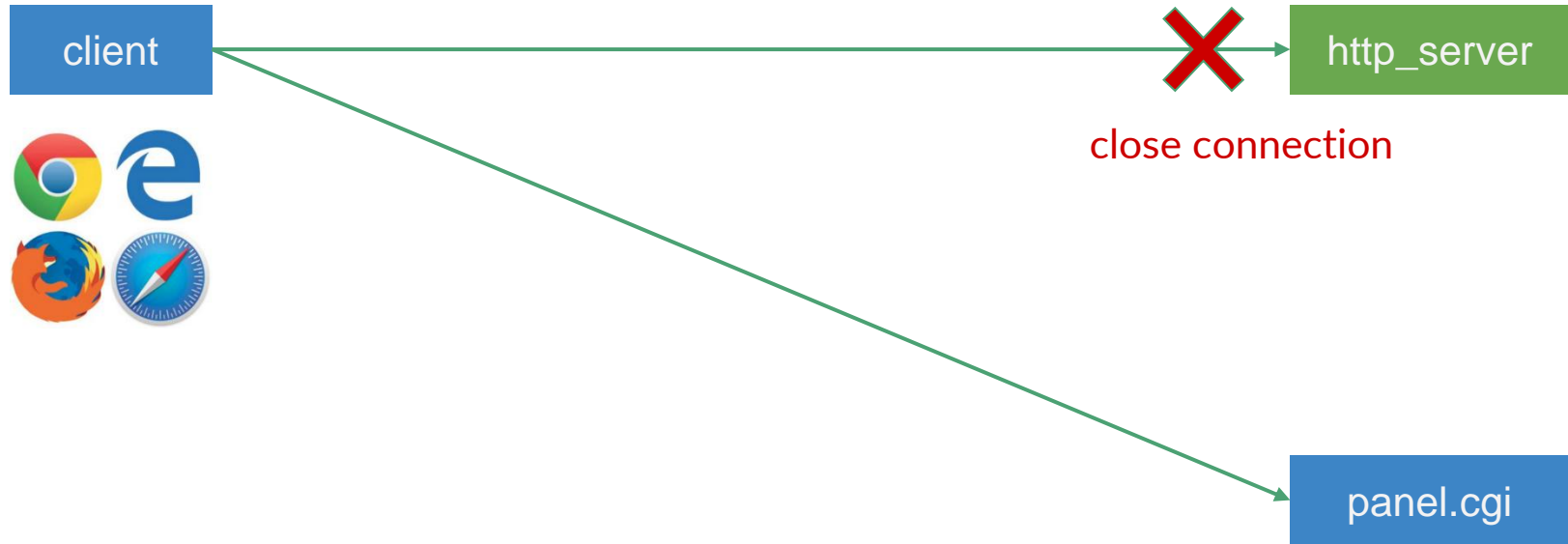
Request 2 - Close Connection



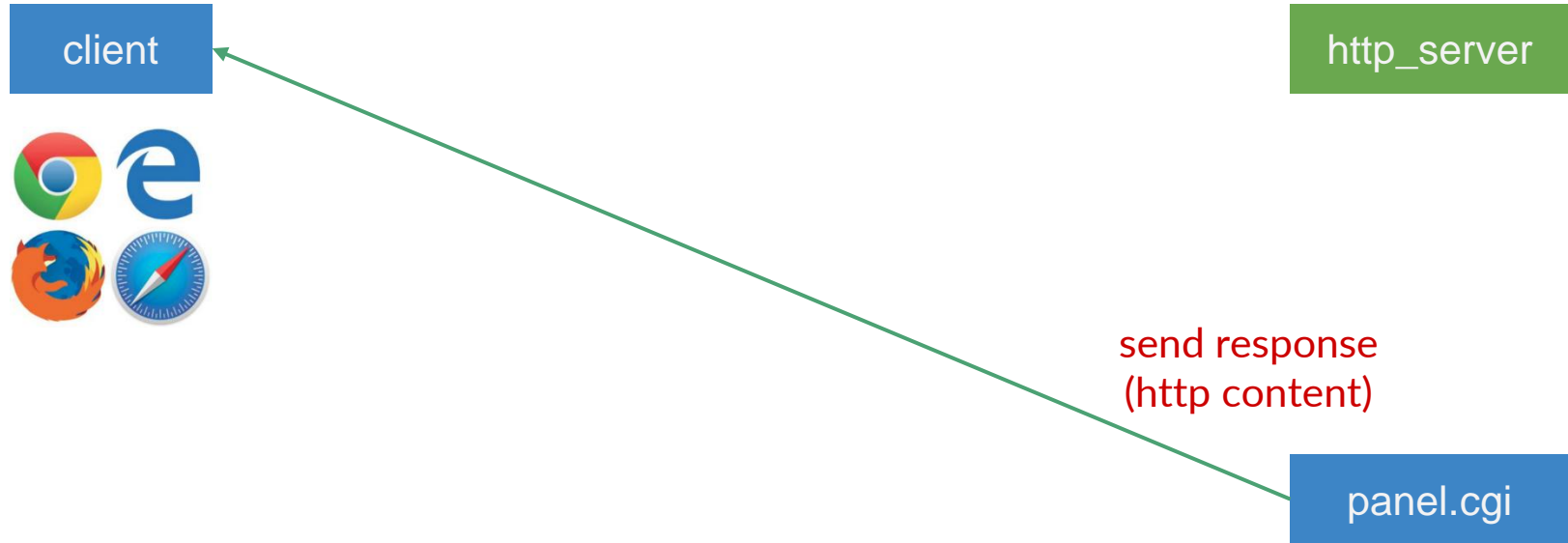
Request 1 - Execute CGI Program



Request 1 - Close Connection



Request 1 - Response



Request 1 - Terminate

client













http_server

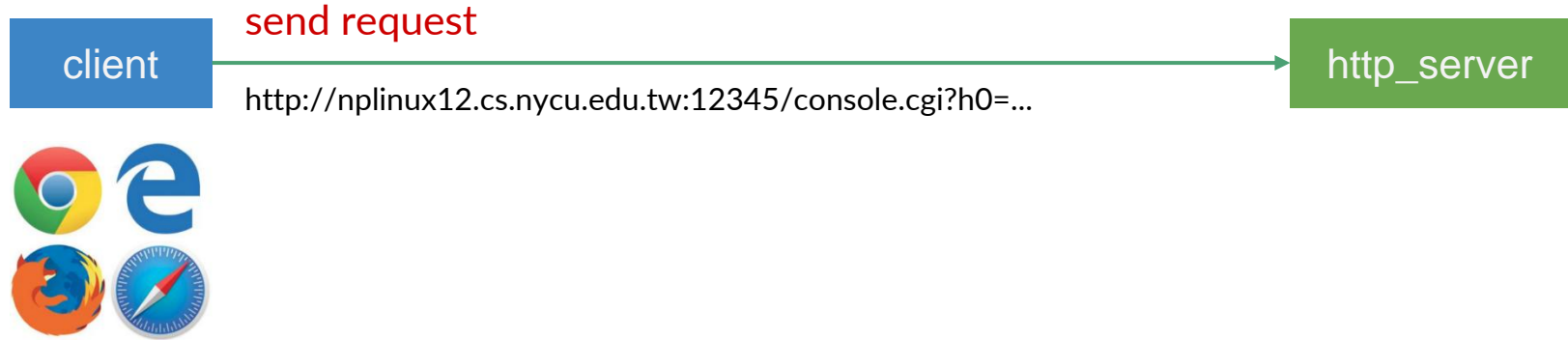
child exit

panel.cgi

Page Generated by panel.cgi

#	Host	Port	Input File	
Session 1	<input type="text" value="nplinux12"/> 	<input type="text" value=".cs.nycu.edu.tw"/>	<input type="text" value="12344"/>	<input type="text" value="t1.txt"/> 
Session 2	<input type="text" value="nplinux12"/> 	<input type="text" value=".cs.nycu.edu.tw"/>	<input type="text" value="12343"/>	<input type="text" value="t2.txt"/> 
Session 3	<input type="text"/> 	<input type="text" value=".cs.nycu.edu.tw"/>	<input type="text"/>	<input type="text"/> 
Session 4	<input type="text"/> 	<input type="text" value=".cs.nycu.edu.tw"/>	<input type="text"/>	<input type="text"/> 
Session 5	<input type="text"/> 	<input type="text" value=".cs.nycu.edu.tw"/>	<input type="text"/>	<input type="text"/> 
				<input type="button" value="Run"/>

Request 2 - Send



Request 2 - Receive

client



receive, parse request

http_server

GET

/console.cgi?h0=nplinux12.cs.nycu.edu.
tw&p0=12344&f0=t1.txt&h1=nplinux1
2.cs.nycu.edu.tw&p1=12343&f1=t2.txt
&h2=&p2=&f2=&h3=&p3=&f3=&h4=&
p4=&f4= HTTP/1.1

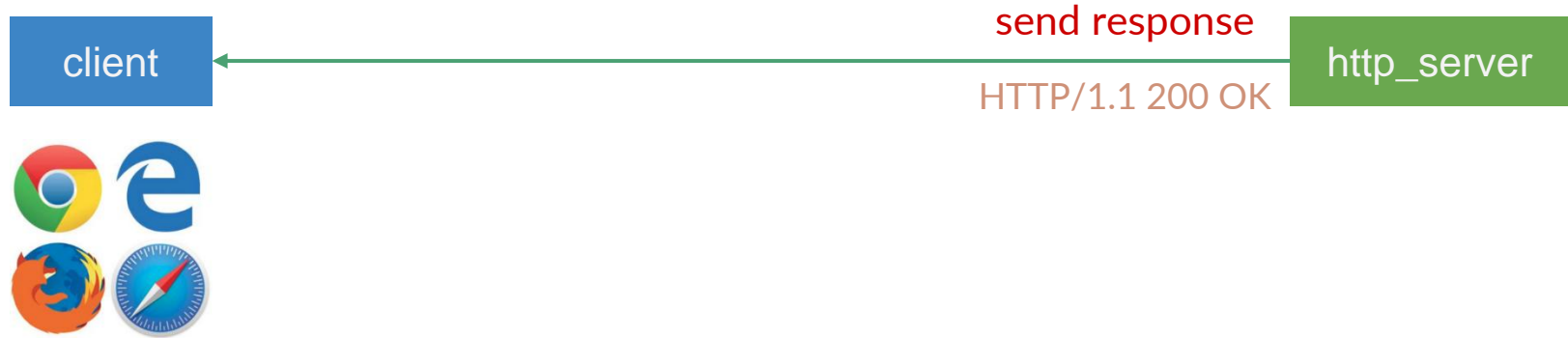
Host: nplinux12.cs.nycu.edu.tw:12345

Connection: keep-alive

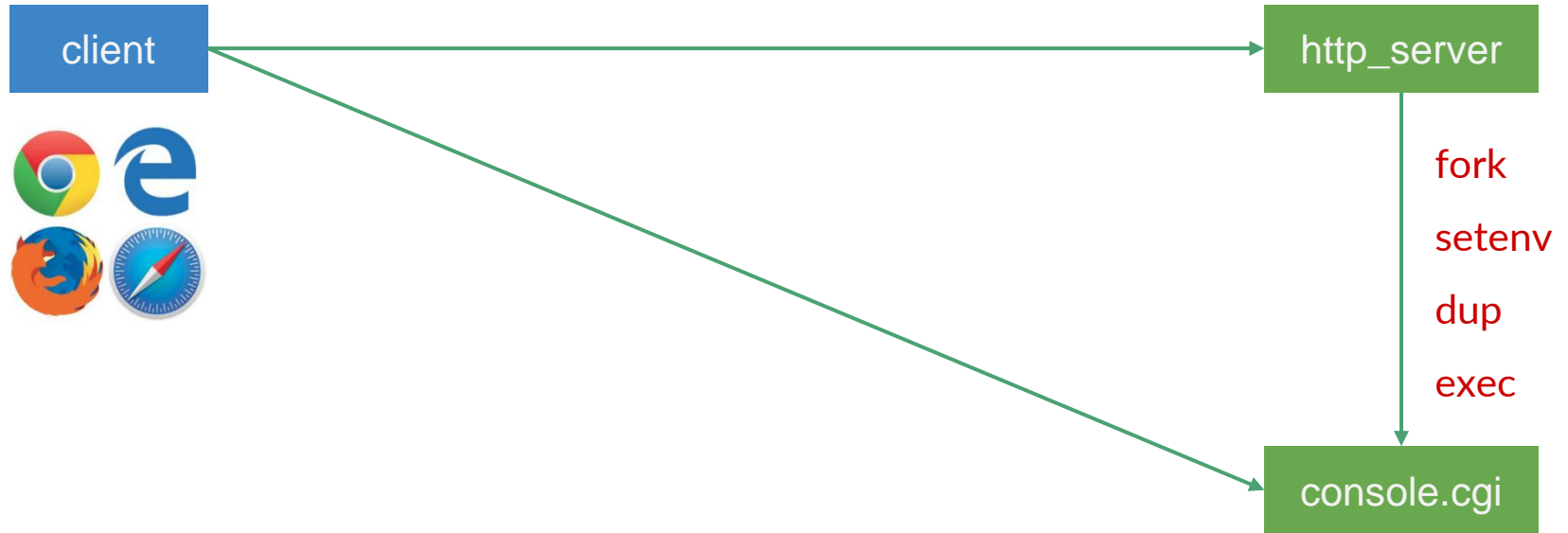
Upgrade-Insecure-Requests: 1

.....

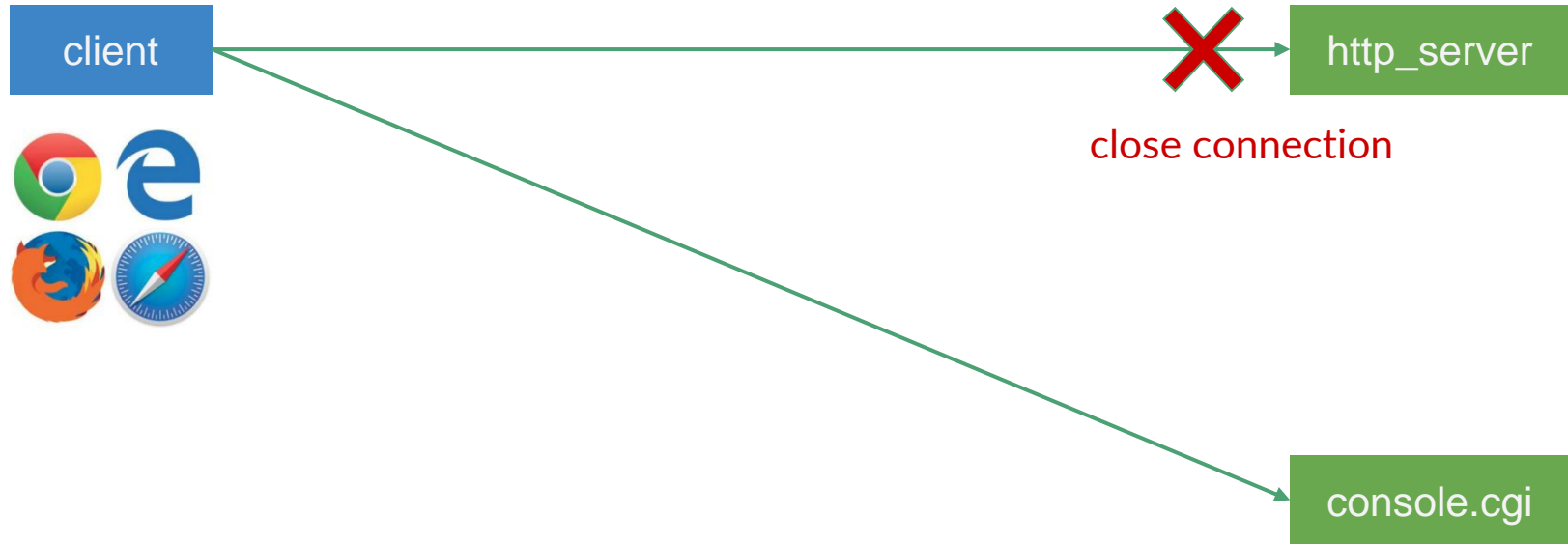
Request 2 - Close Connection



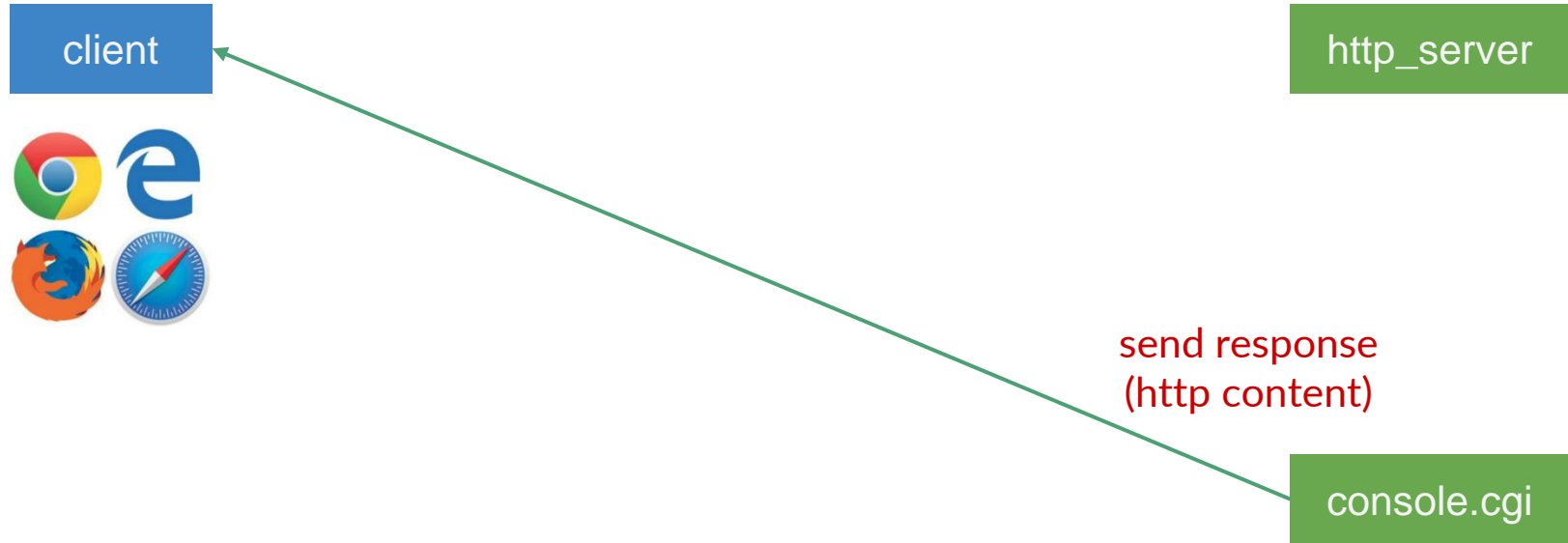
Request 2 - Execute CGI Program



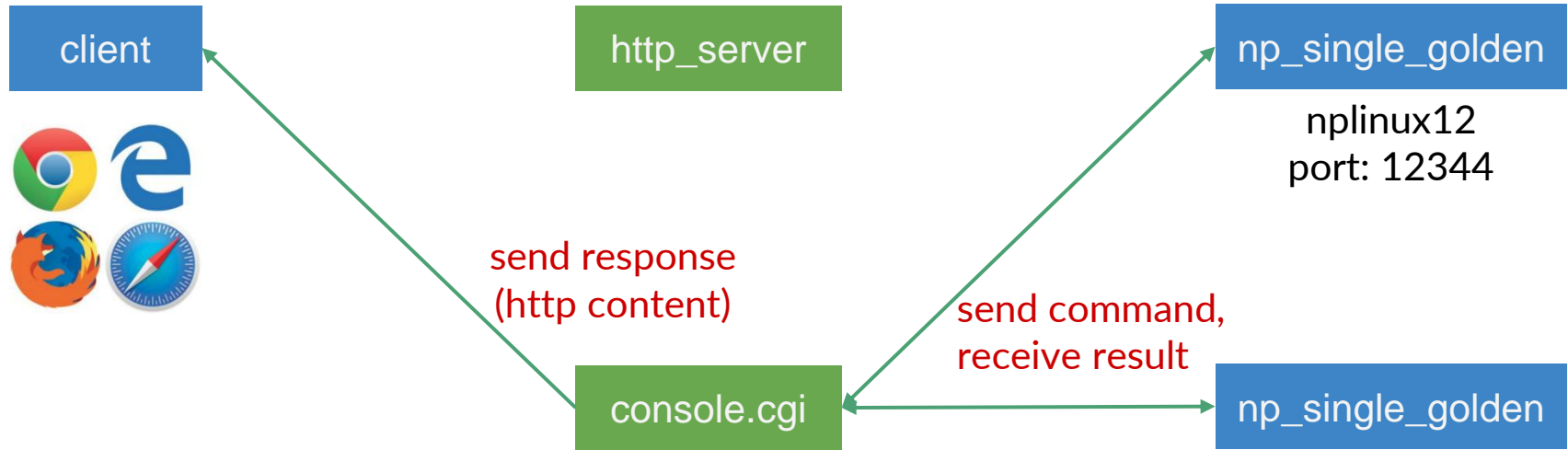
Request 2 - Close Connection



Request 2 - Response



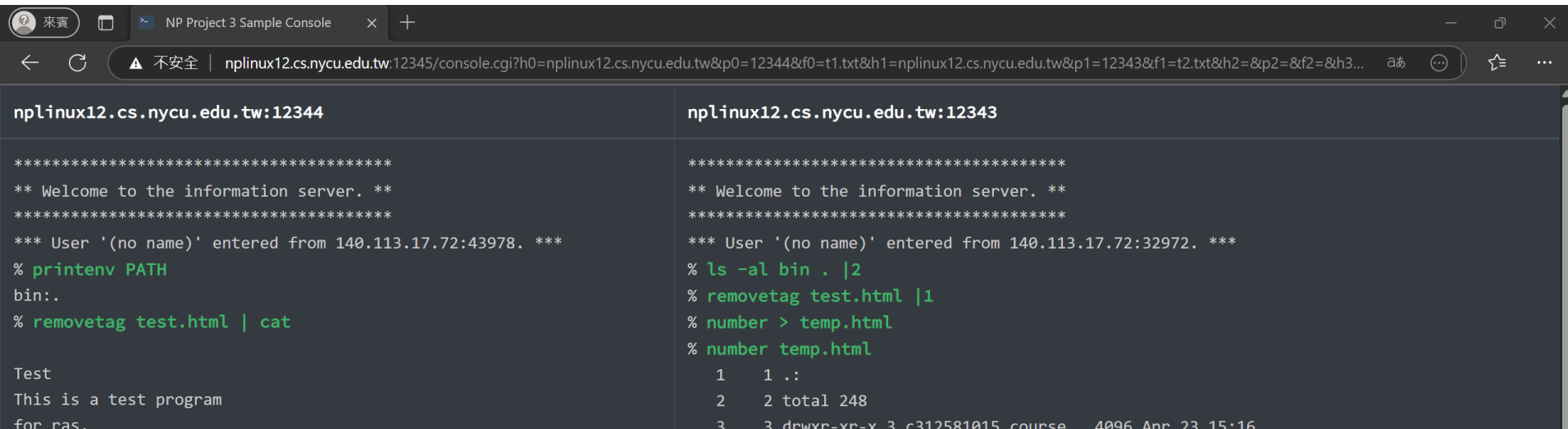
Request 2 - Connect Servers



QUERY_STRING=
h0=nplinux12.cs.nycu.edu.tw&p0=12344&f0=t1.txt&
h1=nplinux12.cs.nycu.edu.tw&p1=12343&f1=t2.txt&
h2=&p2=&f2=&h3=&p3=&f3=&h4=&p4=&f4=

Page Generated by console.cgi

- Display hostname and port of the remote server in each session
- Make sure the commands are displayed in the right order, at the right time, and can be easily distinguished from the outputs



```
nplinux12.cs.nycu.edu.tw:12344
*****
** Welcome to the information server. **
*****
*** User '(no name)' entered from 140.113.17.72:43978. ***
% printenv PATH
bin:.
% rmovetag test.html | cat

Test
This is a test program
for ras.

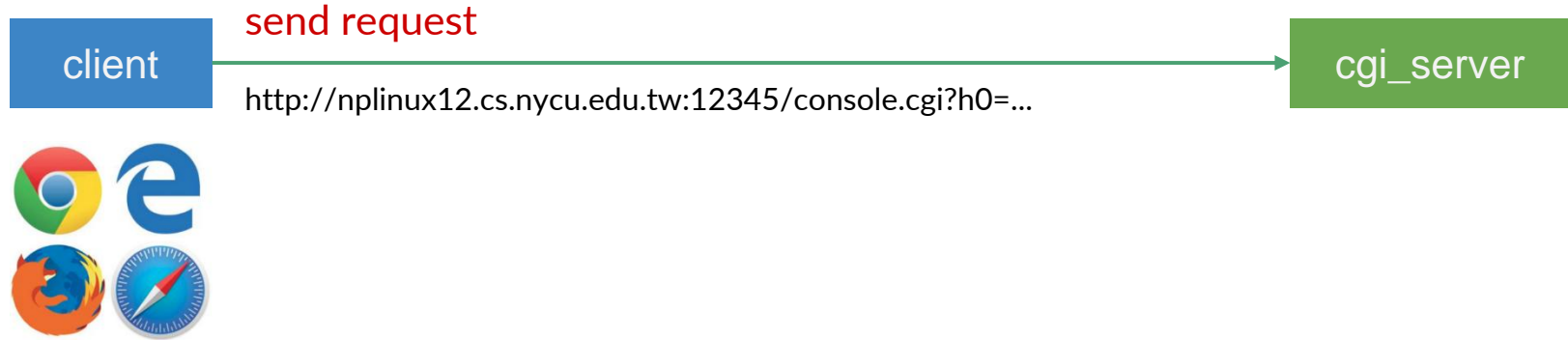
nplinux12.cs.nycu.edu.tw:12343
*****
** Welcome to the information server. **
*****
*** User '(no name)' entered from 140.113.17.72:32972. ***
% ls -al bin . | 2
% rmovetag test.html | 1
% number > temp.html
% number temp.html
1 1 .:
2 2 total 248
3 3 drwxr-xr-x 3 c312581015 course 4096 Apr 23 15:16
```

Part 2: Windows Version

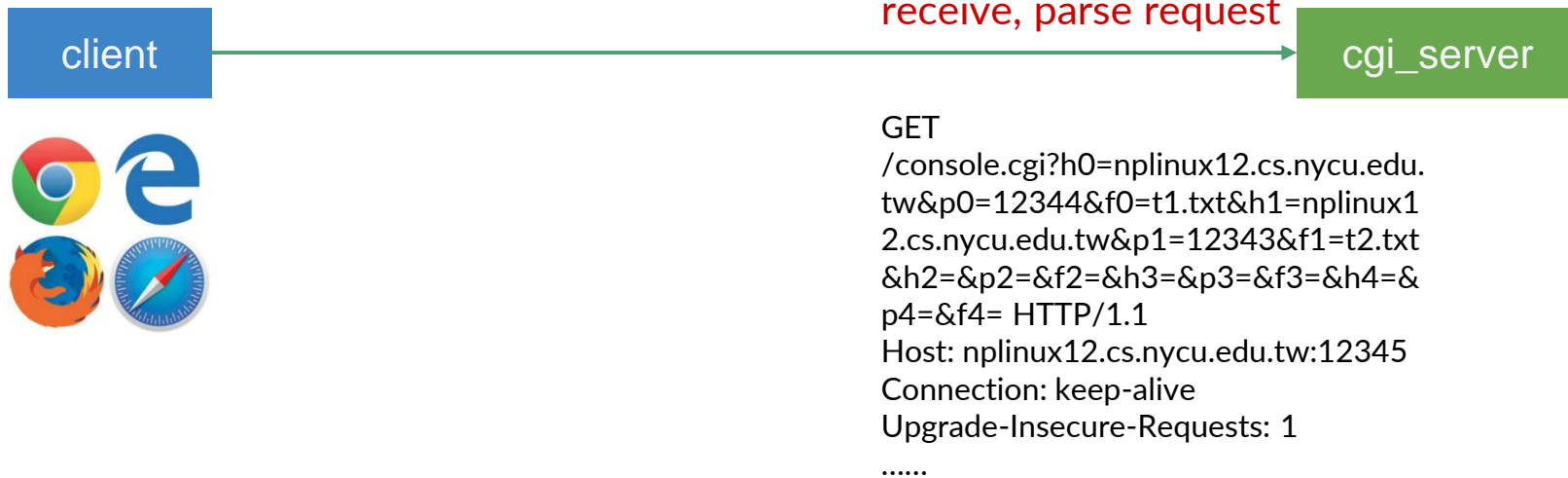
Requirements for Part 2

- In part 2, you are asked to implement a single executable on Windows 10/11, which combines all the functionalities from Part 1 into:
 - A special HTTP server called **cgi_server.exe**
 - It combines the roles of *http_server*, *panel.cgi*, and *console.cgi*

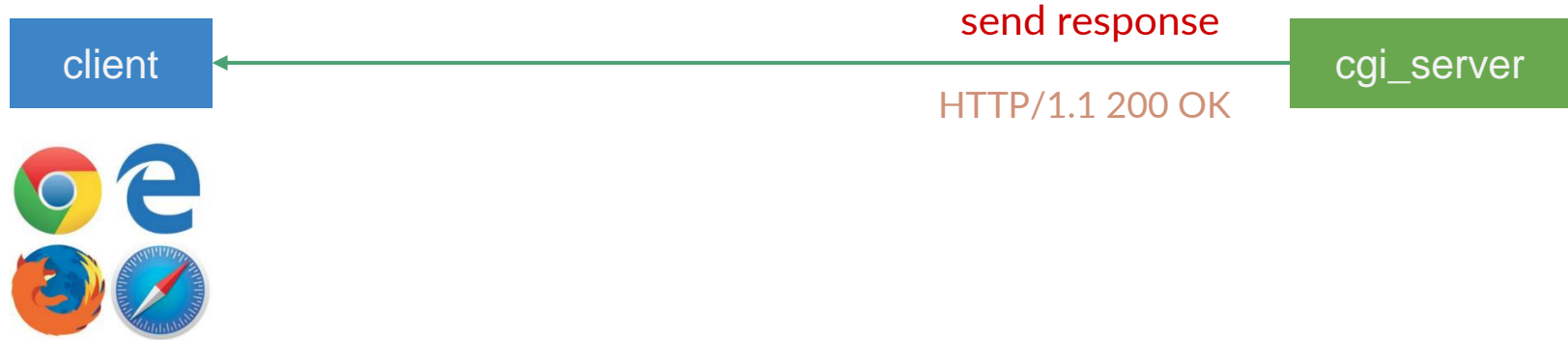
Request 2 - Send



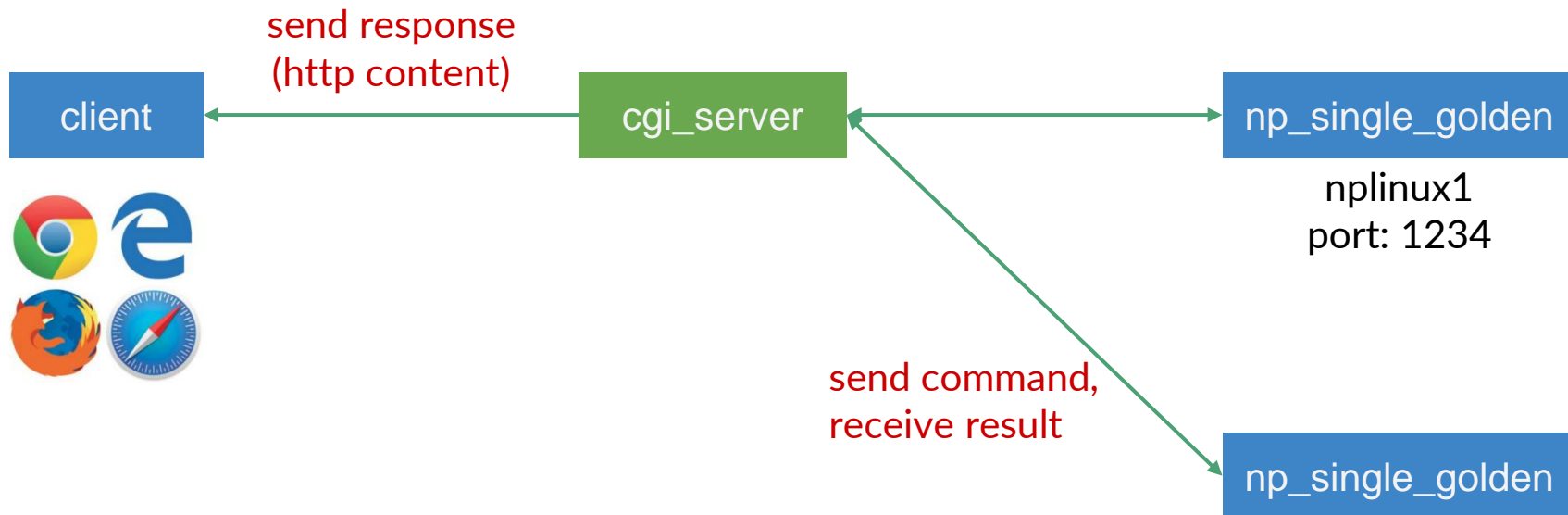
Request 2 - Receive



Request 2 - Response



Request 2 - Connect Servers



QUERY_STRING=
h0=nplinux1.cs.nycu.edu.tw&p0=1234&f0=t1.txt&
h1=nplinux2.cs.nycu.edu.tw&p1=5678&f1=t2.txt&
h2=&p2=&f2=&h3=&p3=&f3=&h4=&p4=&f4=

Reminder

Extra Files

- You should check the files in extra_files.zip first
 - Boost.Asio Example
 - CGI programs
 - np_single_golden
 - etc.

Supplementary

Outline

- Lambda Expressions
- Auto Specifier
- Shared Pointer
- Move
- Boost.Asio Example
- Printing HTML Content in C++

Lambda Expressions (since C++11)

- An unnamed function object capable of capturing variables in scope
- A lambda expression consists of three parts

`[] () { }`

- captures
- params
- body

```
/* without capture */
```

```
function<int(int)> square = [](int x) { return x * x; };
```

```
cout << square(5) << endl; /* output: 25 */
```

Lambda Expressions (since C++11)

- An unnamed function object capable of capturing variables in scope

```
/* capture by reference */
```

```
int x = 0;
```

```
function<int(int)> add = [&x](int y) { x = 1; return x + y; };
```

```
cout << add(3) << endl; /* output: 4 */
```

```
cout << x << endl;      /* output: 1 */
```

```
/* capture by value */
```

```
int x = 0;
```

```
function<int(int)> add = [x](int y) mutable { x = 1; return x + y; };
```

```
cout << add(3) << endl; /* output: 4 */
```

```
cout << x << endl;      /* output: 0 */
```

Lambda Expressions (since C++11)

- Without lambda expression

```
bool by_name(Person a, Person b) {  
    return a.name < b.name;  
}  
bool by_age(Person a, Person b) {  
    return a.age < b.age;  
}  
vector<Person> employees;  
  
/* sort employees ordered by name */  
sort(employees.begin(), employees.end(), by_name);  
  
/* sort employees ordered by age */  
sort(employees.begin(), employees.end(), by_age);
```

Lambda Expressions (since C++11)

- With lambda expression

```
vector<Person> employees;
```

```
/* sort employees ordered by name */
```

```
sort(employees.begin(), employees.end(), [](Person a, Person b) {  
    return a.name < b.name;  
});
```

```
/* sort employees ordered by age */
```

```
sort(employees.begin(), employees.end(), [](Person a, Person b) {  
    return a.age < b.age;  
});
```

Auto Specifier (since C++11)

- Let compiler automatically deduce types

```
// int
```

```
auto a = 1 + 2;
```

```
// int
```

```
auto b = a;
```

```
/* function<int(int)> */
```

```
auto square = [](int x) { return x * x; };
```

```
vector<int> arr;
```

```
/* vector<int>::iterator */
```

```
auto begin_it = arr.begin();
```


Shared Pointer (since C++11)

- **std::shared_ptr** is a smart pointer that retains shared ownership of an object through a pointer
- You do not have to **free** and **delete** manually

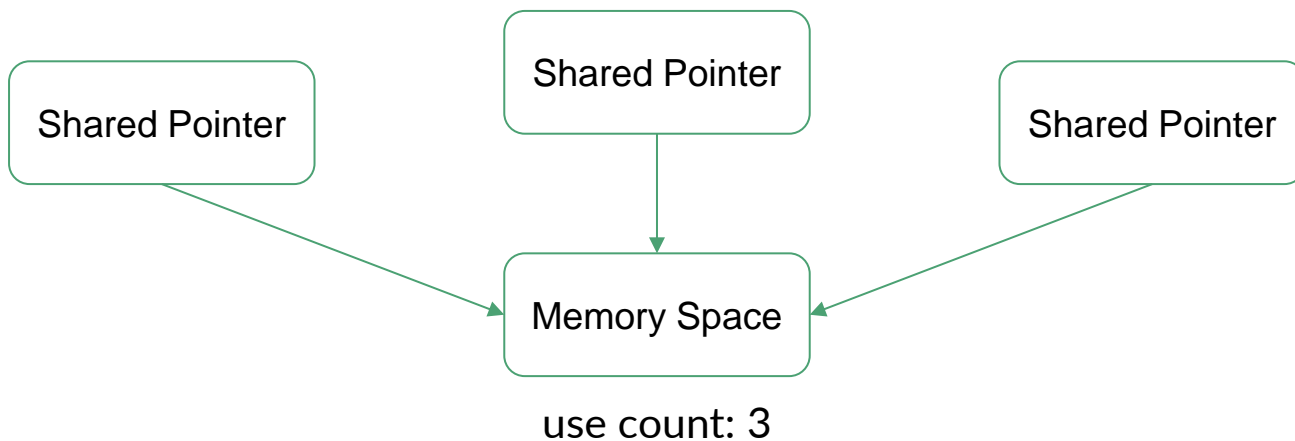
```
std::shared_ptr<int> sp(new int);  
*sp = 5;  
cout << *sp;    /* output: 5 */  
  
auto sp2 = std::make_shared<int>(10);  
cout << *sp2;    /* output: 10 */
```

C++ smart pointers

1. std::unique_ptr
2. std::shared_ptr
3. std::weak_ptr

Shared Pointer (since C++11)

- When will the allocated resource be destroyed?
 - When the last remaining `shared_ptr` owning the object is destroyed (when use count is 0)



Shared Pointer (since C++11)

- When will the allocated resource be destroyed?
 - When the last remaining `shared_ptr` owning the object is destroyed (when use count is 0)

```
{
    std::shared_ptr<int> sp(new int);
    {
        std::shared_ptr<int> sp2(sp);
        cout << sp.use_count();  /* output: 2 */
    }
    cout << sp.use_count();      /* output: 1 */
} /* free the space */
```

enable_shared_from_this

- Allows an object that is currently managed by a shared_ptr **safely** generate additional shared_ptr instances

```
class MyClass : public std::enable_shared_from_this<MyClass>{
public:
    std::shared_ptr<MyClass> get_ptr() {
        return shared_from_this(); // Good
        return this;                // Bad
    }
};
```

Move (since C++11)

- **std::move** is used to indicate that an object may be "moved from", i.e. allowing the efficient transfer of resources from one object to another.

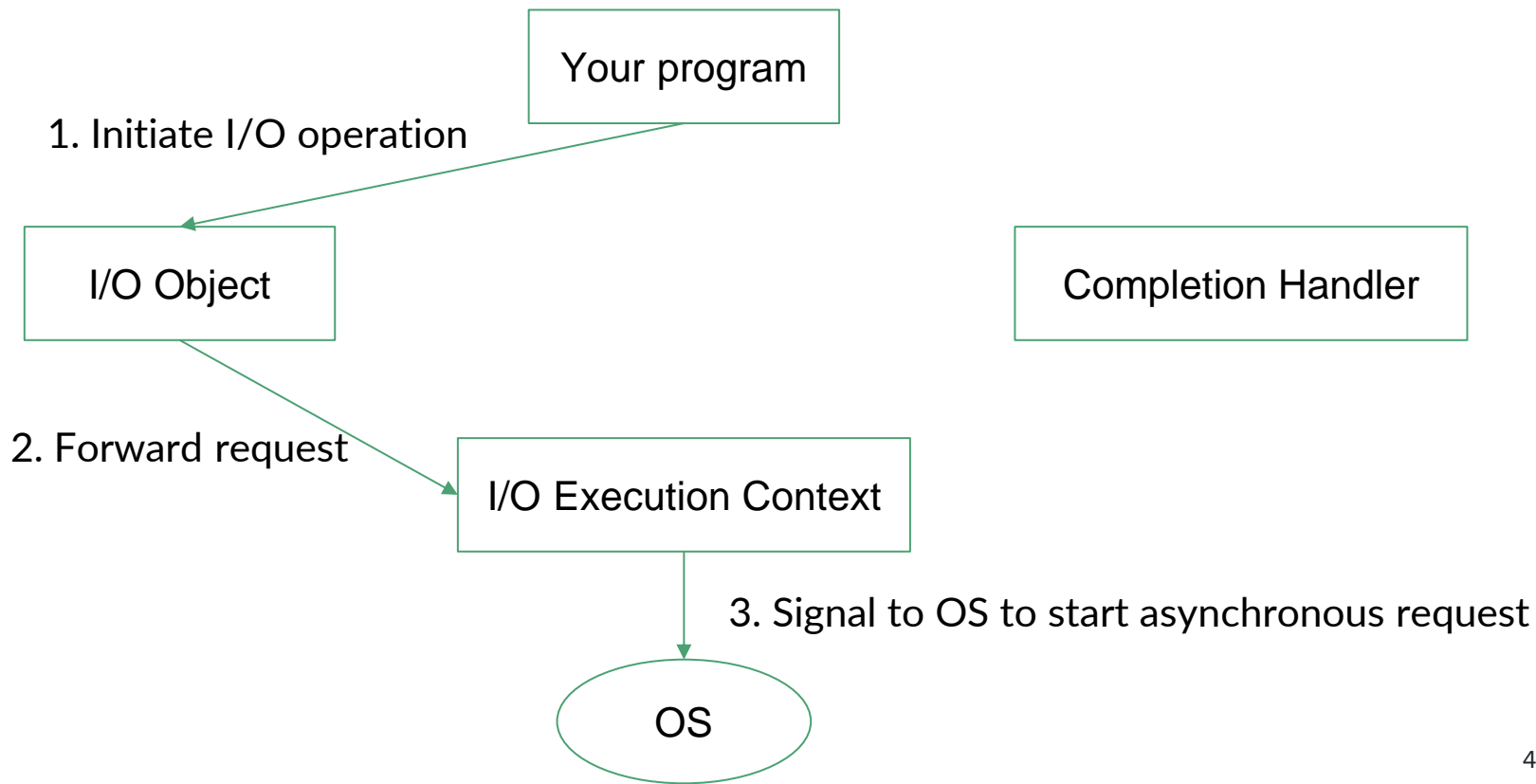
```
string a = "Hello";
```

```
/* extra cost of copying string a */  
string b = a;
```

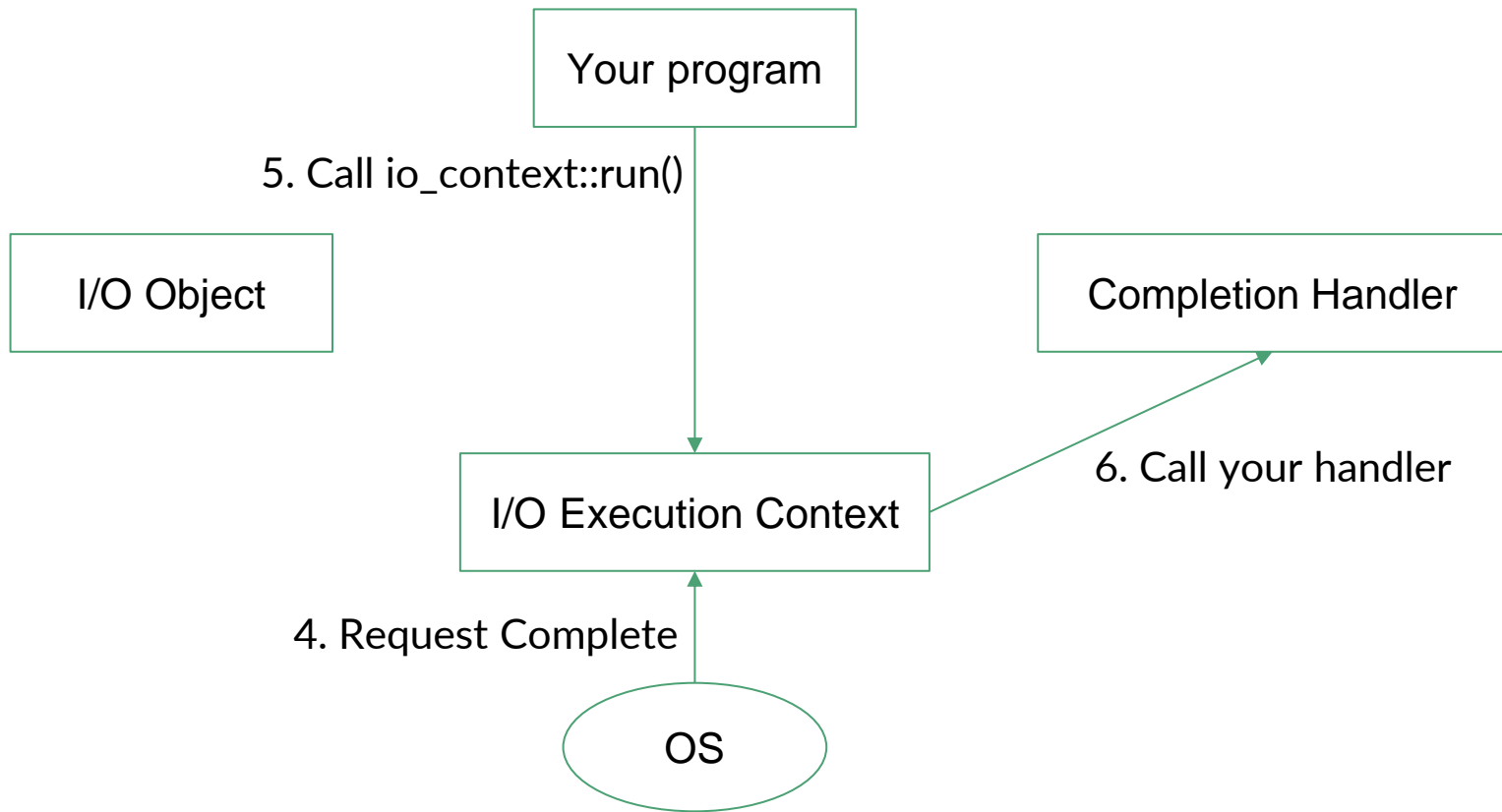
```
/* the content of string a will be moved into string c */  
string c = move(a);
```

```
cout << "a" << a << " " << endl; // output: "a"  
cout << "b" << b << " " << endl; // output: "Hello"  
cout << "c" << c << " " << endl; // output: "Hello"
```

Boost.Asio Overview



Boost.Asio Overview



Boost.Asio Example

- A **c++11** asynchronous echo server example from Boost.Asio documentation
- The following codes are simplified because of space limitation

Boost.Asio Example

- Call `io_context::run()`

```
int main(int argc, char* argv[])
{
    boost::asio::io_context io_context;
    server s(io_context, std::atoi(argv[1]));

    /* VERY IMPORTANT! */
    io_context.run();

    return 0;
}
```

Boost.Asio Example

```
class server {  
private:  
    tcp::acceptor acceptor_;  
  
public:  
    server(boost::asio::io_context& io_context, short port)  
        : acceptor_(io_context, tcp::endpoint(tcp::v4(), port)) {  
        do_accept();  
    }  
  
    .  
    .  
    .  
  
};
```


Boost.Asio Example

```
void do_accept() {  
    acceptor_.async_accept(  
        [this](error_code ec, tcp::socket socket) {  
            if (!ec) {  
                std::make_shared<session>(std::move(socket))->start();  
            }  
            do_accept();  
        });  
}
```

forward request to io_context
provide handler for I/O completion

io_context

socket cannot be copied, so move is used here



Boost.Asio Example

```
class session : public std::enable_shared_from_this<session> {  
private:
```

```
    tcp::socket socket_;  
    enum { max_length = 1024 };  
    char data_[max_length];
```

```
public:
```

```
    session(tcp::socket socket) : socket_(std::move(socket)) {}  
    void start() { do_read(); }
```

socket cannot be copied, so move is used here

- .
- .
- .

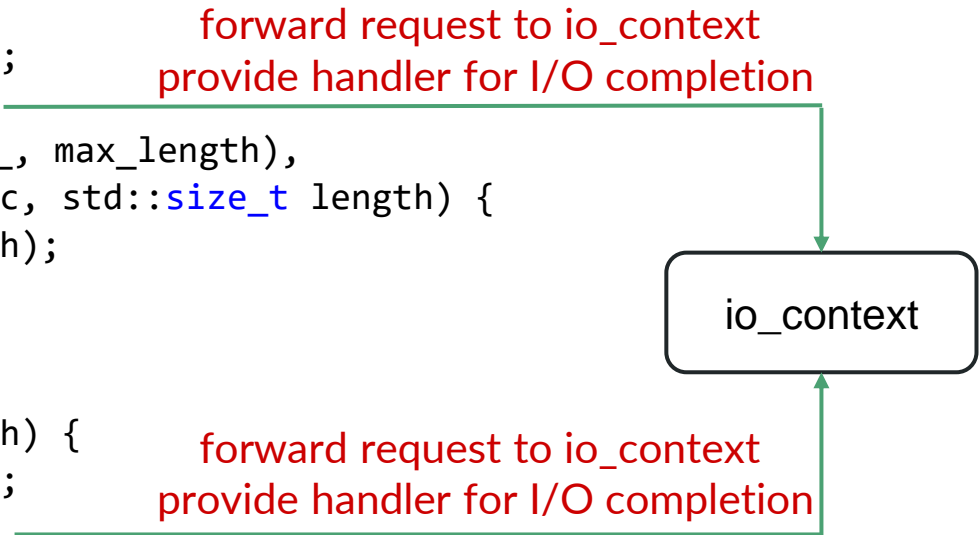
Boost.Asio Example

```
void do_read() {  
    auto self(shared_from_this());  
    socket_.async_read_some(  
        boost::asio::buffer(data_, max_length),  
        [this, self](error_code ec, std::size_t length) {  
            if (!ec) do_write(length);  
        });  
}  
  
void do_write(std::size_t length) {  
    auto self(shared_from_this());  
    boost::asio::async_write(  
        socket_, boost::asio::buffer(data_, length),  
        [this, self](error_code ec, std::size_t length) {  
            if (!ec) do_read();  
        });  
}
```

forward request to io_context
provide handler for I/O completion

io_context

forward request to io_context
provide handler for I/O completion



Printing HTML Content in C++

console.cpp

```
std::string html_content =
    "<!DOCTYPE html>\n"
    "<html lang='en'>\n"
    "  <head>\n"
    "    <meta charset='UTF-8' />\n"
    "    <title>NP Project 3 Sample Console</title>\n"
    "...
    </head>\n"
    "  <body>\n"
    "...
    </body>\n"
    "</html>";
std::cout << html_content << std::flush;
```

Printing HTML Content in C++

console.cpp (Raw String Literal)

```
std::string html_content = R"(
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>NP Project 3 Sample Console</title>
    ...
  </head>
  <body>
    ...
  </body>
</html>
)";
std::cout << html_content << std::flush;
```

Reference

- <https://en.cppreference.com/w/cpp/language/lambda>
- <https://en.cppreference.com/w/cpp/language/auto>
- https://en.cppreference.com/w/cpp/memory/shared_ptr
- https://en.cppreference.com/w/cpp/memory/enable_shared_from_this
- <https://en.cppreference.com/w/cpp/utility/move>
- https://www.boost.org/doc/libs/1_77_0/doc/html/boost_asio.html