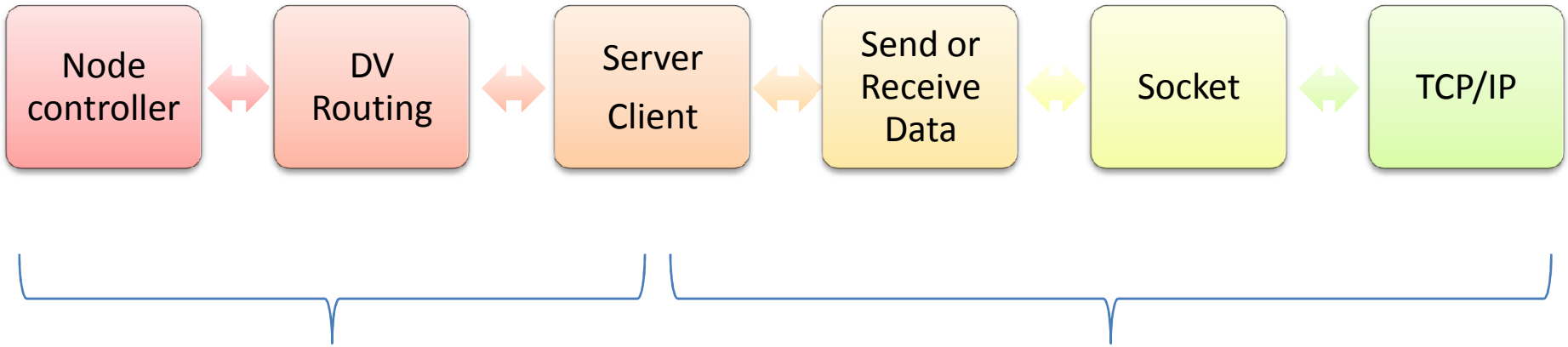
The Implement of Application layer Distance Vector Routing with winsock

By Tius

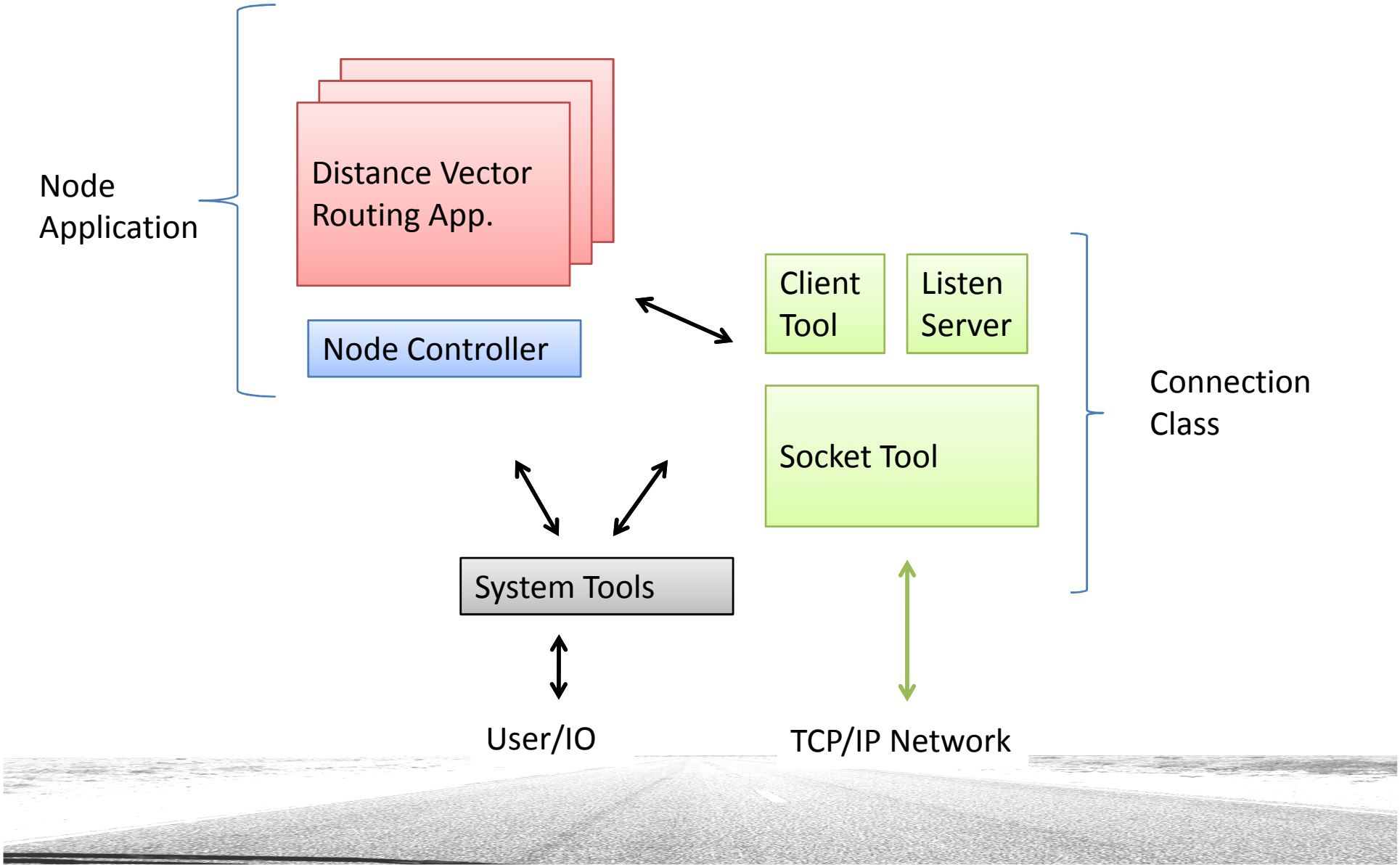
Outline

- Module Design
- C++ implement skills
- Debug improve
- Discuss

Control Flow



Class Hierarchy



Example code from book

```
/*compute new shortest paths*/
for (i=0; i<4; i++) {
    tmp[i] = dt3.costs[i][0];
    for (j=1; j<4; j++) {
        if (tmp[i] > dt3.costs[i][j])
            tmp[i] = dt3.costs[i][j];
    }
    if (tmp[i] != spath3[i] ) {
        /*update shortest path*/
        spath3[i] = tmp[i];
        flag =1;
    }
}
return flag;
```

Mix data structure and algorithm implement will lead to weak maintainability (High dependency)

```
/*update distance table*/
for (i=0; i<4; i++) {
    dt3.costs[i][linkid] = dt3.costs[i][linkid] - oldcost + newcost;
    if (dt3.costs[i][linkid] > INFINITY)
        dt3.costs[i][linkid] = INFINITY;
}
```

DV algorithm

Loop

wait new cost or update

for each neighbors Y
Dx(Y) = min{ c(x,v) + Dv(Y) }

if our Dx(y) changed
send DV to all neighbors

forever

DV algorithm

Loop

wait new cost or update

for each neighbors Y
Dx(Y) = min{ c(x,v) + Dv(Y) }

if our Dx(y) changed
send DV to all neighbors

forever

Old Cost
= new minimum cost

DV Class Hierarchy

Table 提供存取/修改功能

Distance Vector Table

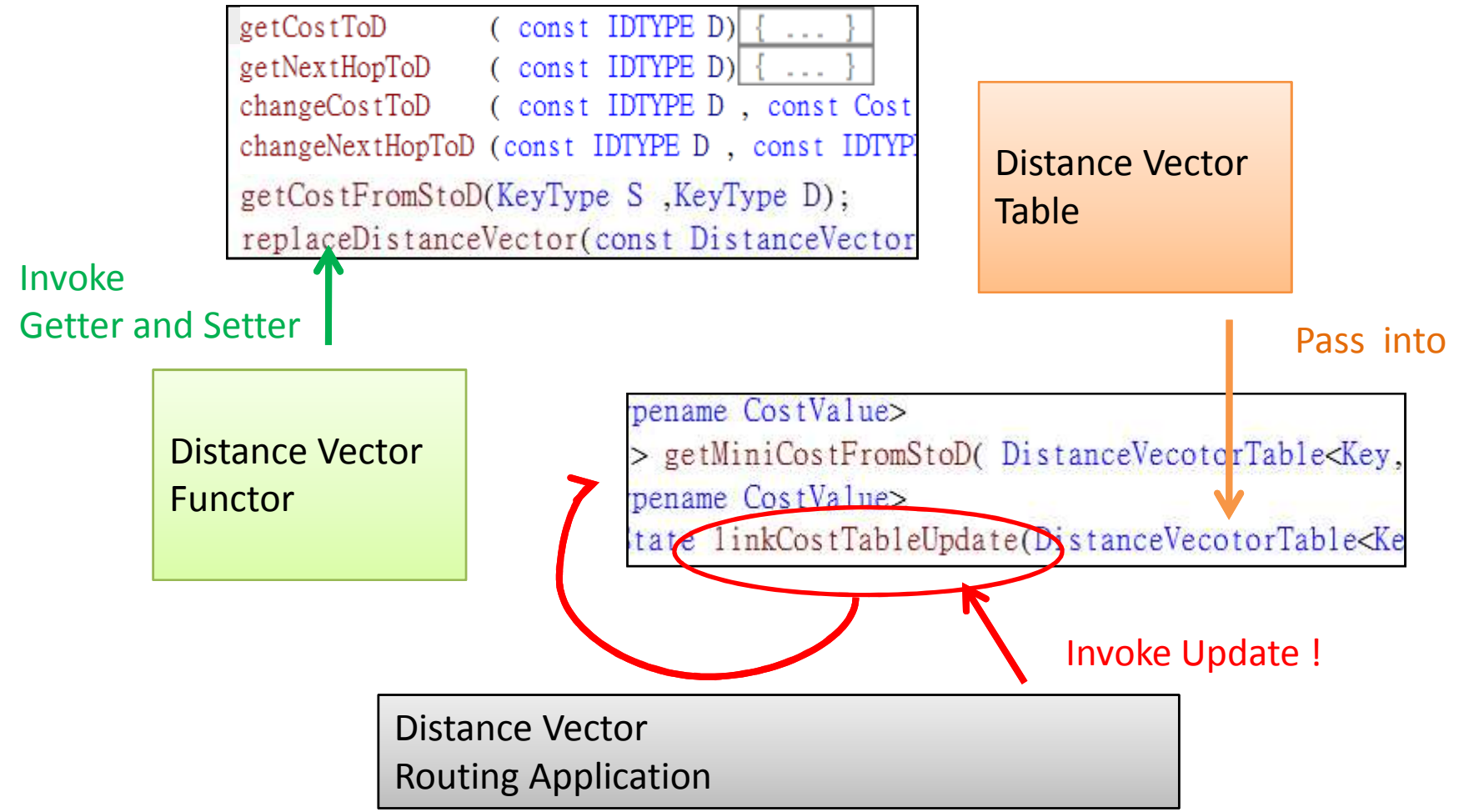
Distance Vector Functor

Functor 提供演算法策略(Policy)

Distance Vector Routing Application

App. 負責呼叫執行

Class Hierarchy



More clear code

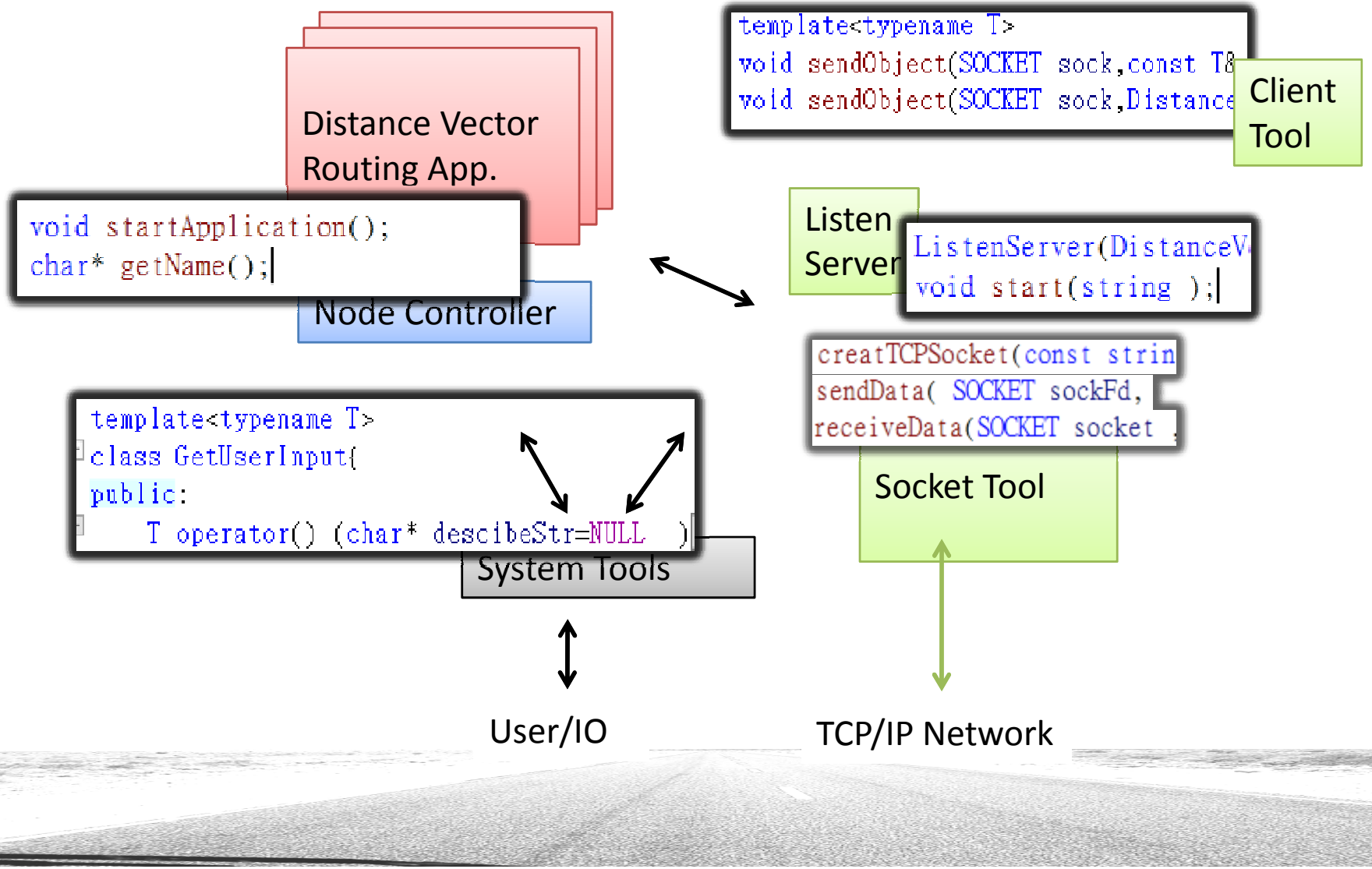
```
// travel all neighbors
for(unsigned int ix=0; ix < NEIGHBORS_AMOUNT ;++ix){
    if( isReachable(dvTable, SOURCE, ix, DEST) )
    {
        calResult = dvTable.getCostFromStoD(SOURCE,ix)+ dvTable.getCostFromStoD(ix,DEST);

        if(miniCost > calResult ){
            miniCost = calResult;
            nextHop  = ix;
        }
    }
}
return pair<Key , CostValue>(nextHop,miniCost);
```

```
/*compute new shortest paths*/
for (i=0; i<4; i++) {
    tmp[i] = dt3.costs[i][0];
    for (j=1; j<4; j++) {
        if (tmp[i] > dt3.costs[i][j])
            tmp[i] = dt3.costs[i][j];
    }
    if (tmp[i] != spath3[i] ) {
        /*update shortest path*/
        spath3[i] = tmp[i];
        flag =1;
    }
}
return flag;
```

Principles of Module Design

- High/low level Modules depend on abstract interface
  - Dependency inverse principle
- Use “Composition” as possible as you can
- Encapsulation common part
- Decrease complexity of a class
- Create a clear and simple class

Depend on Abstract interfaces



Outline

- Module Design
- C++ implement skills
- Debug improve
- Discuss

C++ implement skills

- When we using socket to send data …..

```
buf[LINELEN] = '\0';     /* ensure line null-termination */
outchars = strlen(buf);
(void) send(socket, buf, outchars, 0);
 cc = recv(socket, &buf[inchars], outchars-inchars, 0);
```

C++ implement skills

- When we using socket to send data …..

```
buf[LINELEN] = '\0';     /* ensure line null-termination */
outchars = strlen(buf);
(void) send(socket, buf, outchars, 0);
 cc = recv(socket, &buf[incha
```

1. Need to do type cast transform obj. into string type
2. Maintain a buffer
3. Count size
4. ….

String Stream

- Provide a interface to manipulate string as stream

1. No need to do type cast
2. No need to maintain buffer
3. No need to count str.
4. Support any functions of cin/cout/cerr (include mix type transform)

```
//transfer T data
stringstream sstr;
sstr<<data;

int result = send(sockFd, sstr.str().c_str() , sstr.str().size()+1 , 0);
```

```
sstr<<"test"<<123.5465<<"dafsdfsd"<<endl;
```

Generic send()

- It can send any type support output operator !!

```
template<typename T>
void sendData( SOCKET sockFd, T data )
{
    //transfer T data
    stringstream sstr;
    sstr<<data;

    int result = send(sockFd, sstr.str().c_str() , sstr.str().size()+1 , 0);
```

```
std::ostream& operator<< (std::ostream & ostr ,T &msh ) { ... }
```

Partial Specialize send()

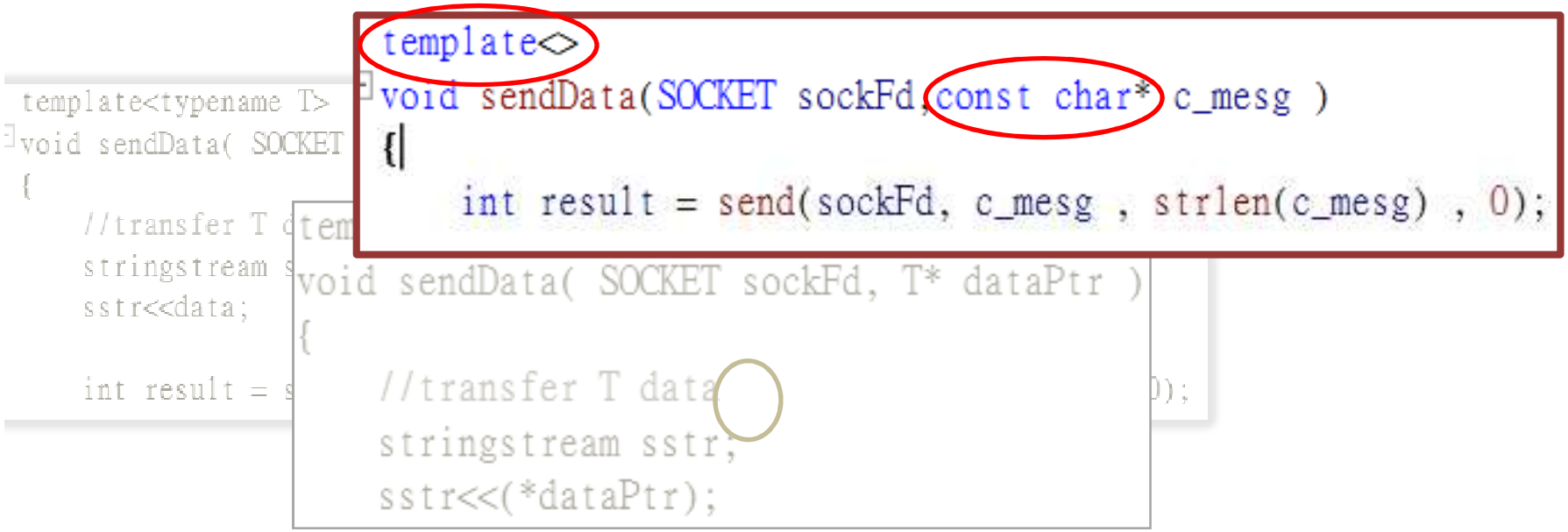
- Now it can distinguish ANY Pointer type !!!

```
template<typename T>
void sendData( SOCKET sockFd, T data )
{
    //transfer T data
    stringstream sstr;
    sstr<<data;

    int result = send(sc
```

```
template<typename T>
void sendData( SOCKET sockFd, T* dataPtr )
{
    //transfer T data
    stringstream sstr;
    sstr<<(*dataPtr);
```

Full Specialize send()

- Now it can get better performance at char* type !!



- Now, you can send anything, and encapsulate common part
- Better Maintainability
- Easily Module Test
- Also you can add ERROR handle to throw error to upper layer

Server and Client Mapping

- We need to map the operation take between client and server
- Two-Step EX
  - See Header
  - Decided action take

Enum & Client Server mapping

```
public:
    static enum MesgHeader{
        MH_Ini,
        MH_Stable,
        MH_AsynLinkCostUpdate,
        MH_SynLinkCostUpdate,
        MH_VectorUpdate,
        MH_TopoIni,
        MH_IniSucc,
        MH_IniFail,
        MH_ConnecReady,
        MH_EXIT
    };
```

```
MesgHeader create_MH_ConnecReady_Header()[
MesgHeader create_AsynLinkCostChange_Heade
MesgHeader create_SynLinkCostChange_Header
MesgHeader create_StableSignal_Header() {
MesgHeader create_VectorUpdate_Header() {
MesgHeader create_TopoIni_Header() { ... }
MesgHeader create_IniFail_Header() { ... }
MesgHeader create_IniSuccess_Header() { ..
MesgHeader create_EXIT_Header() { ... }
```

↑ Factory class

Enum & Client Server mapping

```
switch(newInputHeader){

case MesgHeaderFactory::MH_ConnecReady:
    dvrApp.connectReady=1;
    break;
case MesgHeaderFactory::MH_IniFail :
    this->dvrApp.iniFailCount++;
    break;
case MesgHeaderFactory::MH_IniSucc :
    this->dvrApp.iniSuccCount++;
    break;
case MesgHeaderFactory::MH_AsynLinkCostUpdate:
    //should now do update
    receUpdateVector(cliSocket);
    dvrApp.passiveUpdate();

    break;
```

More about generic

- Generic Input getter with plug-in error handle function(functor is a better choice)

```
userChoose = -1;
userChoose = GetUserInput<int>()("Please choose services:");
```

```
template<typename T>
class GetUserInput{
public:
    typedef bool (*CheckFunc)(T);

    T operator() (char* descibeStr=NULL  ) { ... }
    T operator() ( CheckFunc checkF, char* descibeStr = NULL  );
```

Outline

- Module Design
- C++ implement skills
- Debug improve
- Discuss

Debug Print

- Don’t waste your debug message
- Enable/Disable it as you want

Debug Print

- Don’t waste your debug message
- Enable/Disable it as you want

```
12  #define DEBUG_COUT
13  #ifdef DEBUG_COUT
14
15  #define  DE_PRINT(X); std::cout<<X<<std::endl
16  #endif
17
18  #ifndef DEBUG_COUT
19  #define DE_PRINT(X);
20  #endif
```

Debug Print

```
12  #define DEBUG_COUT
13  #ifdef DEBUG_COUT
14
15  #define  DE_PRINT(X); std::cout<<X<<std::endl;
16  #endif
17
18  #ifndef DEBUG_COUT
19  #define DE_PRINT(X);
20  #endif
```

If it's defined, it act as cout

If no define, define it as no-effect macro

Debug Print

```
DE_PRINT("I sent Stable Signal");

DE_PRINT("Send INI FAIL");
clientTool.sendObject(neisSocks.find(iter->first)-

DE_PRINT("CHV"<<changeCount<<"INVISIBLE CONNECTION :sent Stable Signal FORCE IGNORE");
```

```
12  #define DEBUG_COUT
13  #ifdef DEBUG_COUT
14
15  #define  DE_PRINT(X); std::cout<<X<<std::endl;
16  #endif
17
18  #ifndef DEBUG_COUT
19  #define DE_PRINT(X);
20  #endif
21
```

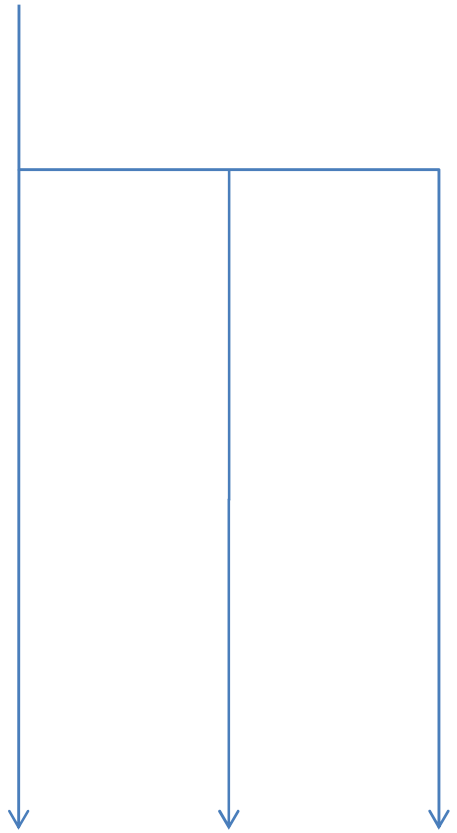
Multi-Level Debug

```
#ifdef ALL_DEG
#define SEND_COUT(X)
#define UPDATE_COUT(X)


....

#endif
#ifndef ALL_DEG

#ifdef SEND_COUT(X)
#define SEND_COUT(X)
#endif


....
#endif
```

ALL Debug on

If ALL Debug is OFF
Then
  check other Debug Setting
….

Race and Thread…



```
CRITICAL_SECTION criticalSig ;

CRITICAL_SECTION updateSig ;

InitializeCriticalSection (&criticalSig) ;
InitializeCriticalSection (&updateSig)  ;

EnterCriticalSection (&updateSig) ;

    …
    Data modified area
    …

LeaveCriticalSection (&updateSig) ;
```

Discuss

- Class type
  - Value-like ,Policy classes , Pointer class
- Tradeoff
  - Performance and Maintainability
- Readability

Thanks for listening!