#include <IPHlpApi.h>

#pragma comment(lib, "IPHlpApi.lib")

bool GetStatisticOfSysNetworkFlow(DWORD &dwbitTotalRecv, DWORD &dwbitTotalSend)

{

/\*获取MIB-II接口表大小\*/

DWORD dwBufferLen = 0;

GetIfTable(NULL, &dwBufferLen, 0);

/\*创建MIB-II接口表\*/

PMIB\_IFTABLE pMibIfTable = (MIB\_IFTABLE\*)malloc(dwBufferLen);

/\*获取MIB-II接口表\*/

DWORD dwRet = GetIfTable(pMibIfTable, &dwBufferLen, 0);

if (NO\_ERROR != dwRet)

{

std::cout << "GetIfTable != NO\_ERROR, ErrorCode=" << dwRet << std::endl;

free(pMibIfTable);

return false;

}

dwbitTotalRecv = dwbitTotalSend = 0;

/\*多网卡\*/

for (int i = 0; i != pMibIfTable->dwNumEntries; ++i)//端口数

{

// if (pMibIfTable->table[i].dwType <= 23 && pMibIfTable->table[i].dwType != MIB\_IF\_TYPE\_LOOPBACK)

if (pMibIfTable->table[i].dwType <= 23)

// if(pMibIfTable->table[i].dwType == MIB\_IF\_TYPE\_ETHERNET)

{

dwbitTotalRecv += pMibIfTable->table[i].dwInOctets;

dwbitTotalSend += pMibIfTable->table[i].dwOutOctets;

}

}

/\*Byte转bit\*/

//dwbitTotalRecv \*= 8;

//dwbitTotalSend \*= 8;

// cout << "dwbitTotalRecv = " << dwbitTotalRecv << "dwbitTotalSend == " << dwbitTotalSend << endl;

free(pMibIfTable);

return true;

}

bool GetSysNetworkFlowByMidTable(DWORD &dwbpsRecv, DWORD &dwbpsSend)

{

/\*首次获取\*/

DWORD dwTotalRecv1 = 0, dwTotalSend1 = 0;

if (!GetStatisticOfSysNetworkFlow(dwTotalRecv1, dwTotalSend1))

{

printf("GetStatisticOfSysNetworkFlow == false\n");

return false;

}

Sleep(1000);

/\*再取\*/

DWORD dwTotalRecv2 = 0, dwTotalSend2 = 0;

if (!GetStatisticOfSysNetworkFlow(dwTotalRecv2, dwTotalSend2))

{

printf("GetStatisticOfSysNetworkFlow == false\n");

return false;

}

/\*计算\*/

//dwbpsRecv = dwTotalRecv2 - dwTotalRecv1;

//dwbpsSend = dwTotalSend2 - dwTotalSend1;

dwbpsRecv = dwTotalRecv2;// -dwTotalRecv1;

dwbpsSend = dwTotalSend2;// -dwTotalSend1;

// cout << "dwbpsRecv" << dwbpsRecv << "dwbpsSend " << dwbpsSend << endl;

return true;

}