**به نام خدا**

**افزایش تحمل پذیری خطا و قابلیت دسترسی در مراکز داده (Data Center) مبتنی بر مجازی سازی سرور**

پیاده سازی شده در NS-2

نسخه Ns-2.28

دو فایل ارسالی pdf شما اطلاعات کاملی از نحوه شبیه سازی ارائه می دهد و توضیحات کلی را دربردارد.

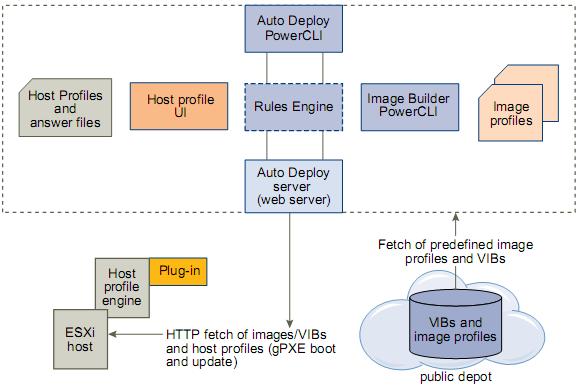
لازم است که شما حتما با عملکرد VMWare vSpace ESXi آشنا باشید. به همین دلیل بهتر است که آن را نصب کنید. به صورت خلاصه و ساده روش نصب آن را با تمام مواردی که نیاز دارد، ابتدا می نویسیم. سپس به توضیح خطوط برنامه در شبیه ساز NS-2 می پردازیم.

دقت شود برای بعضی از بخش ها در کد یک توضیح در مورد تابع اصلی نوشته شده است زیرا خطوط به هم مرتبط هستند و نمی توان به صورت تکه تکه توضیحات آن را نوشت.

**VMWare vSpace ESXi 5.0**

ابتدا توضیحات خودتان در فایل pdf دوم را مطالعه نمایید که توضیحات کاملی بودند. سپس این بخش را نیز مطالعه نمایید.

هنگامی که از حالت خود مستقر (Auto Deploy) سرور استفاده می شود، یک هاست از DHCP به منظور گرفتن آدرس IP و گرفتن بوت PXE سرور استفاده می کند. حالت بوت PXE دانلود شده است ومحیط پیش بوت در هاست اجرا می شود. حال قسمت ذکر شده با حالت خود مستقر سرور ارتباط برقرار می کند که می تواند قوانین موتور راه اندازه (شبکه اصلی) را چک کند. بر اساس قواعد موجود در شبکه اصلی، حالت بوت PXE به هاست پیشنهاد می شود و پروفایل هاست به هاست ارجاع داده می شود.



برای اینکه موارد ذکر شده در محیط اتفاق بیفتد به موارد زیر نیاز است:

* ویندوز سرور برای اجرای حالت خود مستقر یا vSphere vCenter
* بوت سرور TFPT
* DNS یا DCP سرور
* vSphere PowerCL
* هاست هایی که توانایی اجرای ESXi5 را داشته باشند.

راهنمای کار vCenter خود مستقر به شما چگونگی کاربرد vSphere vCenter را برای حالت خود مستقر نمایش می دهد. برای اینکار نیاز به ویندوز سرور 2008 نسخه 64 بیتی است. ما لینک برایتان گذاشته ایم. اما می توانید با جستجو در نت، موارد را پیدا کنید.

* ابتدا نسخه 5 vCenter را دریافت کنید:

<http://192.168.95.242/isos/vmware/esxi5/VMware-VIMSetup-all-5.0.0-456005.iso>

* سپس ESXi.0 را دانلود کنید:

<http://192.168.95.242/isos/vmware/esxi5/VMware-ESXi-5.0.0-469512-depot.zip>

* حال برای نصب سرور vCenter 5 می توانید ویدئوی زیر را مشاهده نمایید.

<http://192.168.95.242/isos/vmware/mp4/0001.mp4>

* حال باید Powershell نسخه ترجیحا 2.0 را نصب کنید (دقت شود اگر نسخه vCenter5.0 را نصب کنید، این قسمت اتوماتیک نصب می گردد):

[http://192.168.95.242/isos/vmware/esxi5/WindowsServer2003-KB968930-x64-ENG.exe](http://10.224.95.242/isos/vmware/esxi5/WindowsServer2003-KB968930-x64-ENG.exe)

مشاهده ویدئوی نحوه نصب Powershell 2.0

<http://192.168.95.242/isos/vmware/mp4/0002.mp4>

* حال باید PowerCLI نسخه ترجیحا 2.0 را نصب کنید (دقت شود اگر نسخه vCenter5.0 را نصب کنید، این قسمت اتوماتیک نصب می گردد):

[http://192.168.95.242/isos/vmware/esxi5/VMware-PowerCLI-5.0.0-435426.exe](http://10.224.95.242/isos/vmware/esxi5/VMware-PowerCLI-5.0.0-435426.exe)

مشاهده ویدئوی نحوه نصب PowerCLI

[http://192.168.95.242/isos/vmware/mp4/0003.mp4](http://10.224.95.242/isos/vmware/mp4/0003.mp4)

اگر در زمان نصب به همچین مشکل یا خطایی برخورد کردید:

*File C:\Program Files (X86)\VMware\Infrastructure\vSphere PowerCLI\Scripts\Initialize-VIToolkitEnvironment.ps1 cannot be loaded because the execution of scripts is disabled on this system. Please see "get-help about\_signing" for more details.At line:1 char:2+*

عملیات زیر را به منظور رفع آن انجام دهید:

* گام اول: مسیر را طی و فایل .exe را اجرا کنید:

c:\windows\syswow64\windowspowershell\v1.0\powershell.exe

مقادیر را ست کنید به صورت

set-executionpolicy remotesigned –force

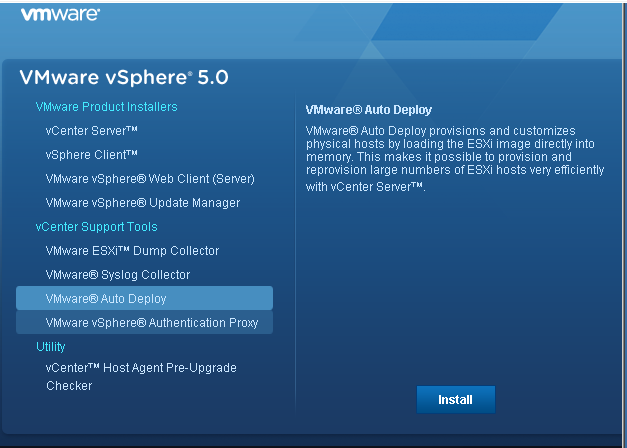
* گام دوم: برنامه PowerCLI را باز کنید.

* گام سوم: دستور زیر را اجرا کنید:

set-executionpolicy remotesigned

یک لینک برای ویدئو که می تواند در همه مراحل شما را یاری دهد:

<http://192.168.95.242/isos/vmware/mp4/0004.mp4>



**توضیح برنامه**

دو فایل servicemgr.h و StdAfx.h فایل های سرآیند (Header Files) هستند که در شبیه ساز NS-2 موجود نیستند. لذا باید در کنار پروژه قرار بگیرند و جز فایل های سرآیند C++ هستند.

**فایل servicemgr.cpp**

مدیریت خدمات در شبیه ساز را به عهده دارد.

#include <stdafx.h>

#include <windows.h>

#include <stdio.h>

#include "servicemgr.h"

فایل های سرآیند مورد نیاز

extern void Server();

اعلان تبع به صورتی که متغیری تعریف خواهیم کرد که درون تابعی باشد و استاتیک نباشد (به آن ها متغیرهای خودکار (Automatic Variables) می گویند).

SERVICE\_STATUS ssStatus;

SERVICE\_STATUS\_HANDLE sshStatusHandle;

تعیین وضعیت سرویس و در دسترس پذیر (هندل) بودن آن.

HANDLE hServiceEvent = NULL;

رویدادها در سرویس برای کاربر قابل دسترس نباشد. (قابل تغییر نباشد زیرا امنیت را به خطر می اندازد (مربوط به مباحث امنیت دیتاسنتر).

در بخش زیر تعیین شده است که همه عملیاتی که صورت میگیرد شامل اجرا، توقف، نقاط بررسی مجدد و نقاطی که هندل به آن اشاره دارد در قالب یک فایل گزارش باشد و ثبت گردد.

BOOL ReportStatusToSCMgr(DWORD dwCurrentState, DWORD dwWin32ExitCode, DWORD dwCheckPoint, DWORD dwWaitHint)

{

BOOL fResult;

if (dwCurrentState==SERVICE\_START\_PENDING)

ssStatus.dwControlsAccepted=0;

else

ssStatus.dwControlsAccepted=SERVICE\_ACCEPT\_STOP; // | SERVICE\_ACCEPT\_PAUSE\_CONTINUE;

ssStatus.dwCurrentState=dwCurrentState;

ssStatus.dwWin32ExitCode=dwWin32ExitCode;

ssStatus.dwCheckPoint=dwCheckPoint;

ssStatus.dwWaitHint=dwWaitHint;

if (!(fResult = SetServiceStatus(sshStatusHandle, &ssStatus))) {

SetEvent(hServiceEvent);

}

return fResult;

}

در بخش زیر از یک اشاره گر 32 بیتی به یک رشته متناهی ثابت 8 بیتی در ویندوز جهت دودویی کردن مسیر و وابستگی های سرویس استفاده شده است. (دستور LPCSTR این اشاره گر ذکر شده می باشد)

void RegisterService(LPCSTR lpszBinaryPathName, LPCSTR lpszDependencies)

{

printf("Create %s Service for executable %s\n", GetServiceName(), lpszBinaryPathName);

SC\_HANDLE schService, schSCManager;

schSCManager=OpenSCManager(NULL, NULL, SC\_MANAGER\_ALL\_ACCESS);

if (schSCManager!=NULL) {

schService=CreateService(

schSCManager,

GetServiceName(FALSE),

GetServiceName(TRUE),

SERVICE\_ALL\_ACCESS,

SERVICE\_WIN32\_OWN\_PROCESS,

SERVICE\_DEMAND\_START,

SERVICE\_ERROR\_NORMAL,

(char\*)lpszBinaryPathName,

NULL,

در این Null عملیات تعادل بارگذاری شبکه اتفاق نمی افتد.

NULL,

حالت Null بالا باعث برچسب نگذاشتن به شناساگرها در زمان تعادل کردن شبکه و بسته می شود.

lpszDependencies,

NULL,

حالت Null بالا محساب محلی سیستم است (Local System Account)

NULL);

حالت Null بالا عمل PWD را خنثی می کند. PWD مخفف Print Working Directory است و دستوری در سیستم عامل های شبکه و یونیکس برای چاپ کردن مسیر جاری دایرکتوری استفاده می گردد. (نبود این مسیر در زمان اجرای سرویس به امنیت کاری کنک می کند)

if (schService==NULL)

printf("CreateService Error: %ld", GetLastError());

else

printf("CreateService SUCCESS\n");

}

CloseServiceHandle(schService);

CloseServiceHandle(schSCManager);

}

در بخش زیر خدماتی که ثبت نمی شوند و در گزارش نیستند و نیازی به آن ها نیست را آورده ایم. (در واقع بخش زیر معرفی Failover Clustering می باشد که نیازی به ثبت وقایع آن نداریم)

void UnregisterService() {

SC\_HANDLE schService, schSCManager;

schSCManager=OpenSCManager(NULL, NULL, SC\_MANAGER\_ALL\_ACCESS);

schService=OpenService(schSCManager,

GetServiceName(FALSE),

SERVICE\_ALL\_ACCESS);

if (DeleteService(schService))

printf("Service %s successfully unregistered.\n", GetServiceName());

else

printf("Error while unregistering service %s: %ld\n", GetServiceName(), GetLastError());

CloseServiceHandle(schService);

CloseServiceHandle(schSCManager);

}

کنترل سرویس ها در بخش زیر صورت می گیرد. این کنترل ها شامل حالت سرویس دهی، گزارش از حالت های سرویس دهی، ثبت گزارش کلی عملکرد ها و رویداد ها، سرویسی که در حال اجراست یا توقف کرده است، می باشد.

VOID service\_ctrl(DWORD dwCtrlCode) {

DWORD dwState = SERVICE\_RUNNING;

switch(dwCtrlCode) {

case SERVICE\_CONTROL\_STOP:

dwState = SERVICE\_STOP\_PENDING;

ReportStatusToSCMgr(

SERVICE\_STOP\_PENDING,

NO\_ERROR,

1,

3000);

SetEvent(hServiceEvent);

return;

case SERVICE\_CONTROL\_INTERROGATE:

break;

default:

break;

}

ReportStatusToSCMgr(dwState, NO\_ERROR, 0, 0);

}

در بخش زیر اصل کار سرویس دهی صورت می پذیرد.

VOID service\_main(DWORD dwArgs, LPTSTR \*lpszArgv)

{

DWORD dwWait;

sshStatusHandle=RegisterServiceCtrlHandler(GetServiceName(),

(LPHANDLER\_FUNCTION)service\_ctrl);

if (!sshStatusHandle) goto cleanup;

ssStatus.dwServiceType=SERVICE\_WIN32\_OWN\_PROCESS;

ssStatus.dwServiceSpecificExitCode=0;

if(!ReportStatusToSCMgr(SERVICE\_START\_PENDING,

NO\_ERROR,

1,

3000))

goto cleanup;

در این بخش حالت سرویس مشخص می شود. سپس از کدینگ خارج شده و به یک نقطه بررسی مجدد می رسد و سپس نقاطی که هندل به آن نیاز دارد می رسد و زمان آن را تعیین می کند (برحسب ثانیه)

hServiceEvent = CreateEvent(NULL, TRUE, FALSE, NULL);

if (hServiceEvent == (HANDLE)NULL) goto cleanup;

if (!ReportStatusToSCMgr(SERVICE\_START\_PENDING, NO\_ERROR, 2, 3000)) goto cleanup;

سرویس اصلی از این جا شروع می شود. (سرویس های اصلی که در ابتدا ذکر شد)

if (CreateThread(NULL, 4000, LPTHREAD\_START\_ROUTINE(Server), NULL, 0, NULL)==NULL) goto cleanup;

if (!ReportStatusToSCMgr(SERVICE\_RUNNING, NO\_ERROR, 0, 0)) goto cleanup;

dwWait = WaitForSingleObject( hServiceEvent, INFINITE);

یک زمان توقف برای داده های تکی که در شبکه هستند در نظر گرفته شده است. (هدف در دسترس پذیری داده روی دیتاسنتر می باشد)

cleanup:

CloseHandle(hServiceEvent);

سپس در دسترس پذیری داده (Data Handling) با توجه به رویدادی که در سرویس دهی برای آن اتفاق می افتد از بافر پاک می شود.

if (sshStatusHandle!=NULL)

ReportStatusToSCMgr(SERVICE\_STOPPED, 0, 0, 0);

return;

}

شرط بالا نیز برسی می کند که اگر وضعیت دسترس پذیری داده ها دیگر در شرایط Null نیست، سرویس دهی را متوقف می کند و گزارش عملکرد در مدیریت سرویس ثبت می گردد.

فایل vmwaregateway.cpp

در این فایل تنظیمات کلی برای VMWare و درگاه آن و همینطور بخش تعادل باگذاری شبکه انجام میگیرد.

#include "stdafx.h"

#define WINVER 0x401

#include "winsock2.h"

#include <stdio.h>

#include "servicemgr.h"

فایل های سرآیند مورد نیاز

#define SERVICE\_NAME "VMWareGateway"

#define DISPLAY\_NAME "VMWareGateway"

نام سرویس و نام نمایش دهنده در زمان اجر را مشخص کرده ایم.

#define SERVER\_PORT 567

#define BUFFER\_SIZE 1024

تعیین پورت یا درگاه و اندازه بافر مورد نیاز (توصیه می شود بافر 1024 یا 2048 باشد. بیشتر از آن توصیه نمی گردد (مباحث امنیت دیتا سنتر و شبیه سازها)

#define PIPE\_NAME “\\\\.\\pipe\\vmwaredebug"

نام خط لوله (Pipeline) زمان debug کردن.

Const char\* errServerBusy = “Server is currently connected\n";

ثابت errserverbusy مشخص می کند که سرور در حال استفاده و متصل است (حالت پرمشغله بودن و سپس در صف قرار گیری داده ها برای ورود)

در بخش زیر شماره رویداد ها و شاخص ها در آرایه رویداد مشخص می گردد. به شماره ها و تعیین هر پارامتر دقت شود زیرا پرکاربرد هستند.

#define LISTEN\_EVENT 0

#define IO\_EVENT 1

#define PIPE\_CONNECT 2

#define PIPE\_READ 3

#define SERVICE\_EVENT 4

#define MAX\_EVENTS 5

BOOL verbose = FALSE;

برای خروجی دادن زمان نباید خیلی طولانی شود (بسته به اندازه بافرینگ)

در بخش زیر چرخه سرور مرکزی مشخص شده است.

void Server()

{

WORD wVersionRequested;

WSADATA wsaData;

wVersionRequested = MAKEWORD( 2, 2 );

در این قسمت سوکت ها و در دسترس پذیری خط لوله ها نوشته شده است. سوکت ها باید به هم متصل باشند (مراجعه شود به برنامه نویسی سوکت در شبکه های کامپیوتری)

HANDLE pipe;

SOCKET s, conn;

WSAEVENT hEvents[MAX\_EVENTS];

OVERLAPPED ovRead, ovConnect;

char buffersocket[BUFFER\_SIZE];

char bufferpipe[BUFFER\_SIZE];

DWORD dwBytesSocket, dwBytesPipe;

کپی کردن اطلاعات موجود در بافر

BOOL pipeConnected = FALSE;

BOOL socketConnected = FALSE;

وضعیت بخش های ارتباطی (خط لوله و سوکت ها)

در بخش زیر مسائل سوکت بررسی شده است

sockaddr\_in sockaddr;

int err;

int addrlen;

شرط زیر شبکه اولیه را در زمان بارگذاری بررسی می کند. شرط این است که اگر سوکت ها ارتباط نداشته باشند و خط لوله ای نباشد پیغام خطایی چاپ گردد.

if (err = WSAStartup(wVersionRequested, &wsaData)) {

printf("Error initializing network\n");

return;

}

ساخت خط لوله و نامگذاری آن

pipe = CreateNamedPipe(PIPE\_NAME,

PIPE\_ACCESS\_DUPLEX | FILE\_FLAG\_OVERLAPPED,

0,

1,

100,

100,

NMPWAIT\_WAIT\_FOREVER,

NULL);

شرط زیر مسیر خط لوله زمانیکه فاقد اعتبار و هندل باشد را بررسی می کند. (خط لوه ای نیست باید ایجاد بشود)

if (pipe == INVALID\_HANDLE\_VALUE) {

printf("Could not create named pipe (%d)\n", GetLastError());

exit(0);

}

در این بخش سوکتی برای دریافت درخواست های ورودی در نظر گرفته می شود.

hEvents[SERVICE\_EVENT] = hServiceEvent;

hEvents[LISTEN\_EVENT] = WSACreateEvent();

hEvents[IO\_EVENT] = WSACreateEvent();

ovConnect.hEvent = hEvents[PIPE\_CONNECT] = CreateEvent(NULL, TRUE, FALSE, NULL);

ovRead.hEvent = hEvents[PIPE\_READ] = CreateEvent(NULL, TRUE, FALSE, NULL);

if (hEvents[LISTEN\_EVENT] == WSA\_INVALID\_EVENT ||

hEvents[IO\_EVENT] == WSA\_INVALID\_EVENT ||

hEvents[PIPE\_CONNECT] == NULL ||

hEvents[PIPE\_READ] == NULL)

{

printf("Error creating net event\n");

return;

}

err = ConnectNamedPipe(pipe, &ovConnect);

ارتباط آسنکرون به خط لوله نامدار (دارای مسیر)

در زیر سوکت منتظر دریافت بررسی شده است.

s = socket(AF\_INET, SOCK\_STREAM, 0);

if (s == INVALID\_SOCKET) {

printf("Could not initialize socket (%d)\n", GetLastError());

return;

}

sockaddr.sin\_family=AF\_INET;

sockaddr.sin\_port=htons(SERVER\_PORT);

sockaddr.sin\_addr.s\_addr = ADDR\_ANY;

err = bind(s, (struct sockaddr\*)&sockaddr, sizeof(sockaddr));

if (err) {

printf("Could not bind socket (%d)\n", GetLastError());

return;

}

err = listen(s, 5);

WSAEventSelect(s, hEvents[LISTEN\_EVENT], FD\_ACCEPT);

حلقه اصلی برای ماشین مجازی ساز – طبق قواعد دریافت در ماشین شبیه ساز – رجوع به توضیحات فایل پی دی اف خودتان

while(1)

{

DWORD numEvent = WSAWaitForMultipleEvents(MAX\_EVENTS, hEvents,

FALSE, WSA\_INFINITE, TRUE);

//printf("Event: %d\n", numEvent);

switch(numEvent) {

case LISTEN\_EVENT:

printf("incoming VMWare request ");

if (!socketConnected) {

addrlen = sizeof(sockaddr);

conn = accept(s, (struct sockaddr\*)&sockaddr, &addrlen);

if (conn == INVALID\_SOCKET) {

printf("error connecting\n");

}

else {

//err = listen(s, 5); relisten???

hEvents[IO\_EVENT] = WSACreateEvent();

WSAEventSelect(conn, hEvents[IO\_EVENT], FD\_READ|FD\_CLOSE);

socketConnected = TRUE;

printf("accepted\n");

}

}

else {

addrlen = sizeof(sockaddr);

SOCKET tmp = accept(s, (struct sockaddr\*)&sockaddr, &addrlen);

send(tmp, errServerBusy, strlen(errServerBusy), 0);

closesocket(tmp);

printf("denied\n");

}

WSAResetEvent(hEvents[LISTEN\_EVENT]);

break;

case IO\_EVENT:

{

WSANETWORKEVENTS NetEvents;

WSAEnumNetworkEvents(conn, hEvents[IO\_EVENT], &NetEvents);

switch(NetEvents.lNetworkEvents)

{

case FD\_READ:

dwBytesSocket = recv(conn, buffersocket, sizeof(buffersocket), 0);

if (verbose) {

buffersocket[dwBytesSocket]=0;

printf("%s", buffersocket);

}

if (pipeConnected) {

DWORD dwWritten;

WriteFile(pipe, buffersocket, dwBytesSocket, &dwWritten, NULL);

}

break;

case FD\_CLOSE:

closesocket(conn);

socketConnected = FALSE;

listen(s, 5);

printf("VMWare session closed\n");

break;

}

} break;

case PIPE\_CONNECT:

if (!GetOverlappedResult(pipe, &ovConnect, &dwBytesPipe, FALSE))

{

printf("error pipe connect (%d)\n", GetLastError());

break;

}

printf("pipe connected\n");

ResetEvent(hEvents[PIPE\_CONNECT]);

ResetEvent(hEvents[PIPE\_READ]);

pipeConnected = TRUE;

ovRead.Offset = 0;

ovRead.OffsetHigh = 0;

ReadFile(pipe, bufferpipe, sizeof(bufferpipe), &dwBytesPipe, &ovRead);

break;

case PIPE\_READ:

if (GetOverlappedResult(pipe, &ovRead, &dwBytesPipe, FALSE))

{

if (verbose) {

bufferpipe[dwBytesPipe] = 0;

printf(bufferpipe);

}

if (socketConnected)

{

send(conn, bufferpipe, dwBytesPipe, 0);

}

ResetEvent(hEvents[PIPE\_READ]);

}

else

{

DWORD err = GetLastError();

switch (err) {

case ERROR\_BROKEN\_PIPE: /\* disconnect and reconnect pipe \*/

DisconnectNamedPipe(pipe);

ConnectNamedPipe(pipe, &ovConnect);

ResetEvent(hEvents[PIPE\_CONNECT]);

ResetEvent(hEvents[PIPE\_READ]);

pipeConnected = FALSE;

printf("pipe disconnected\n");

continue;break;

}

}

ovRead.Offset = 0;

ovRead.OffsetHigh = 0;

ReadFile(pipe, bufferpipe, sizeof(bufferpipe), &dwBytesPipe, &ovRead);

break;

}

}

WSACleanup();

}

LPCSTR GetServiceName(BOOL bLongName)

{

if (bLongName)

return DISPLAY\_NAME;

else

return SERVICE\_NAME;

}

منویی برای معرفی خودتان. به جای اسم ما نام خود را بنویسید. همچنین دستورات را نمایش میدهد با مشخصات.

void ShowHelp() {

printf("Named pipe <-> VMWare server for Windows NT/2000, 2008 , 2005 (use on your own risk :)\n"

"Namira (nn4e@aol.com), www.nn4e.com\n\n"

"Commands:\n"

"\t/R register as a service\n"

"\t/U unregister service\n"

"\t/T testmode\n"

"\t/V verbose mode\n"

"\nThe pipe name is \"%s\" and the telnet port is %d.\n", PIPE\_NAME, SERVER\_PORT);

}

BOOL CheckArg(int argc, char \* argv[], char \* arg)

{

for (int i=1; i<argc; i++)

if (\_stricmp(argv[i], arg) == 0) return TRUE;

return FALSE;

}

کد بخش اصلی (main) که تعریف ماژول و ثبت وقایع و رویداد و سرویس دهی می باشد.

int main(int argc, char\* argv[])

{

char modulename[512];

GetModuleFileName(NULL, modulename, sizeof(modulename));

if (argc > 1) {

if (CheckArg(argc, argv, "/v"))

{

verbose = TRUE;

printf("verbose mode\n");

}

if (CheckArg(argc, argv, "/r"))

{

RegisterService(modulename, "\0\0");

return 0;

}

if (CheckArg(argc, argv, "/u"))

{

UnregisterService();

return 0;

}

if (CheckArg(argc, argv, "/t")) {

printf("test mode - press Ctrl+C to stop program\n");

hServiceEvent = CreateEvent(NULL, FALSE, FALSE, NULL);

Server();

CloseHandle(hServiceEvent);

return 0;

}

ShowHelp();

}

else {

SERVICE\_TABLE\_ENTRY dispatchTable[] =

{

{ TEXT(SERVICE\_NAME), (LPSERVICE\_MAIN\_FUNCTION)service\_main},

{ NULL, NULL }

};

StartServiceCtrlDispatcher(dispatchTable);

}

return 0;

}

فایل vm.tcl

فایل اصل که باید در NS-2 اجرا شود. کدهای tcl در زمان کپی کردن کمی به هم ریختگی دارند. شکل صحیح کد را در فایل مربوطه مشاهده نمایید.

#!/usr/bin/tclsh

lappend auto\_path {/root/autotest/lib}

خواندن توابع کتابخانه ای از مسیر نصب شده NS-2 در زیر پوشه های

/root/autotest/lib

set work\_dir [file join [file dirname [info script]]]

خواندن مسیر دایرکتوری – همین پوشه و فایل های الحاقی آن که منظور فایل های C++ و .h می باشد.

package require -exact Expect 5.43.0

package require dataPool

package require cmdline

package require math

نیازمندی های بسته شبیه سازی در زمان اجرا که موارد در نسخه NS-2.28 موجود است. به هیچ وجه از نسخه NS-3 استفاده نشود زیرا فاقد Expect 5.43.0 است و باید آن را جداگانه نصب نمایید.

set options {

{d.arg "" "h3c switch ip address"}

{D.arg "" "server ip address"}

}

تنظیمات مربوط به آدرس IP سوییچ ها و سرور

دستورات زیر طول رشته ها و آدرس IP سوییچ ها و سرور را به شبیه ساز برمیگردانند.

Set usage “$argv0 –d ipaddr –D ipaddr"

array set params [::cmdline::getoptions argv $options $usage]

if {![string length $params(d)] || ![regexp – {^(\d{1,3}\.){3}\d{1,3}$} $params(d) \_NULL]} {

puts $usage

return

}

if {![string length $params(D)] || ![regexp – {^(\d{1,3}\.){3}\d{1,3}$} $params(d) \_NULL]} {

puts $usage

return

}

#exp\_internal –f debug.log 0

log\_user 0

#log\_file [args] [[-a] file]

#log\_file [file join $work\_dir output.log]

وقایع در فایلی به نام debug.log ثبت و ذخیره می شود بعد از اتمام شبیه سازی.

پروسه ای برای خواندن درگاه و اگر در زمان اجرا یک درگاه که برنامه کاربردی ماست از کار بیفتد باید توسط هاست ESXi رست شود.

proc rebootEthernetPort {port} {

global params max\_interval min\_interval

spawn telnet $params(d)

expect {

-re ".+login" {}

timeout {puts stderr "timeout";return 1}

}

ابتدا وارد مرکز می شود

send \r

expect {

-re {<H3C>} {}

timeout {puts stderr "timeout";return 2}

}

send "system-view\r"

سپس دید سیستمی پیدا می کند و موارد موجود را دریافت می کند.

expect {

-re {\[H3C\]} {}

timeout {puts stderr "timeout";return 3}

}

send "interface Ethernet 1/0/${port}\r"

سپس یک رابط به عنوان درگاه یا برنامه کاربردی پیدا می کند.

expect {

-re {\[H3C-Ethernet1/0/[0-9]+\]} {}

timeout {puts stderr "timeout";return 4}

}

send "shutdown\r"

درگاه یا برنامه کاربردی از کار می افتد

expect {

-re {\[H3C-Ethernet1/0/[0-9]+\]} \

{puts "[clock format [clock sec] -for "%h %d %Y %H%M%S"] shutdown ethernet 1/0/${port}..........."}

timeout {puts stderr "timeout";return 5}

}

سپس باید ماشین های مجازی که به صورت Mirroring هستند روی سروری دیگر که سرویس دهی می کند، بدون وقفه جایگزین و شروع به کار می کنند.

set \_interval [::math::random $min\_interval $max\_interval]

puts "Sleep $\_interval minute..........."

after [expr {$\_interval \* 60000}]

send "\r"

expect {

-re {\[H3C-Ethernet1/0/[0-9]+\]} {}

timeout {puts stderr "timeout";return 5}

در صورتیکه دوباره یک درگاه یا برنامه کاربردی متوقف شود، عملیات دوباره انجام می شود. دقت شود در فایل vmwaregateway.cpp ما شماره درگاه را 567 و اندازه بافر را 1024 با یک خط لوله ارسال به صورت active/passive در نظر گرفته ایم. لذا برای بالا بردن تحمل خطای برنامه و در دسترس پذیری این اتفاق نیاز است که اندازه بافر را بیشتر کنید. (2048 یا بالاتر که توصیه نمی شود به دلایل امنیت شبکه و دیتاسنتر)

}

send "undo shutdown\r"

expect {

-re {\[H3C-Ethernet1/0/[0-9]+\]} \

{puts "[clock format [clock sec] -for "%h %d %Y %H%M%S"] undo shutdown ethernet 1/0/${port}..........."}

timeout {puts stderr "timeout";return 5}

}

قابلیت برگشت پذیری برای جایگزین کردن نیز وجود دارد. این عمل تحمل خطای برنامه و در دسترس پذیری را بالاتر می برد.

Close

wait

return 0

در زمانیکه عملیات چه با خرابی چه صحیح انجام شد، داده ارسالی بسته می شود، و منتظر پاسخ بعدی می ماند.

};#end proc rebootEthernetPort

پروسه ای برای ریبوت کردن FTP که پروتکل ارسال فایل روی دیتا سنتر می باشد. زمان شروع و خاتمه ارسال را در سرور در نظر میگیرد. همچنین سبک نوشتن زمان را نیز در این جا در نظر گرفته ایم. این که چند مرتبه این کار را انجام داده ایم زیرا می خواهیم سرورهای مختلفی نیز داشته باشیم.

proc rebootFtpServer {} {

global params max\_interval min\_interval

exec staf $params(D) process start "/etc/init.d/vsftpd stop"

puts "stop $params(D) FTP server..........."

set \_interval [::math::random $min\_interval $max\_interval]

puts "[clock format [clock sec] -for "%h %d %Y %H%M%S"] Sleep $\_interval minute..........."

after [expr {$\_interval \* 60000}]

exec staf $params(D) process start "/etc/init.d/vsftpd start"

puts "[clock format [clock sec] -for "%h %d %Y %H%M%S"] start $params(D) FTP server..........."

return 0

};#end proc rebootFtpServer

proc rebootFtpServer {} {

global params max\_interval min\_interval

exec staf $params(D) process start command /root/bin/stopVsftpd.sh

puts "[clock format [clock sec] -for "%h %d %Y %H%M%S"] stop $params(D) FTP server..........."

set \_interval [::math::random 1 $max\_interval]

puts "Sleep $\_interval minute..........."

after [expr {$\_interval \* 60000}]

exec staf $params(D) process start command /root/bin/startVsftpd.sh

puts "[clock format [clock sec] -for "%h %d %Y %H%M%S"] start $params(D) FTP server..........."

return 0

};#end proc rebootFtpServer

proc rebootTftpServer {} {

global params max\_interval min\_interval

exec staf $params(D) process start command /root/bin/stopTftp.sh

puts "[clock format [clock sec] -for "%h %d %Y %H%M%S"] stop $params(D) TFTP server..........."

set \_interval [::math::random $min\_interval $max\_interval]

puts "Sleep $\_interval minute..........."

after [expr {$\_interval \* 60000}]

exec staf $params(D) process start command /root/bin/startTftp.sh

puts "[clock format [clock sec] -for "%h %d %Y %H%M%S"] start $params(D) TFTP server..........."

return 0

};#end proc rebootTftpServer

پروسه ریبوت FTP روی دیتا سنتر و سرور

proc rebootFtpAndTftpServer {} {

rebootFtpServer

rebootTftpServer

return 0

};#end proc rebootFtpAndTftpServer

set min\_interval 1

set max\_interval 11

set min\_interval\_idle 1

set max\_interval\_idle 11

::dataPool optPool

optPool addItem "rebootEthernetPort 2"

optPool addItem "rebootEthernetPort 4"

optPool addItem "rebootEthernetPort 6"

optPool addItem "rebootEthernetPort 8"

optPool addItem "rebootTftpServer"

optPool addItem "rebootFtpServer"

optPool addItem "rebootFtpAndTftpServer"

optPool reIndex

while 1 {

catch [optPool getItem]

set \_interval [::math::random $min\_interval\_idle $max\_interval\_idle]

puts "Idle Sleep $\_interval minute..........."

after [expr {$\_interval \* 60000}]

};#end while 1

زمان بیکاری سرور و افزایش قابلیت در دسترس پذیری آن.

فایل های energy.tr و vm.tr برای نمایش انرژی مصرفی در هر دو روش یعنی تعادل بارگذاری شبکه و مجازی سازی (فایل های throuphut)، simulated.xg و vm.xg و vms.xg نمودارهای خروجی که البته در زمان اجرا نیز نمایش داده می شود (در xgraph.exe) و vm.nam خروجی انیمیشن که البته با دستور ns run در فایل vm.tcl نیز نمایش داده می شود و در nam.exe این کار صورت می گیرد.