**LEDBAT for Windows Validation Guide**

**Requirements:** 2 -- WS2016 server SKU Azure VMs

**FYI:** LEDBAT is currently not supported on WS 2016. It is available for experimentation and you can use it for validation, but, not in production. You can also run this validation guide on hosts with WS 2019 insider builds.

**Steps:**

1. Bring both servers up to date using WU and install (This is only necessary for WS2016)

<https://www.catalog.update.microsoft.com/Search.aspx?q=KB4132216>

<https://www.catalog.update.microsoft.com/Search.aspx?q=KB4284833>

1. Install IIS on one of the VMs.
2. Use ipconfig to obtain the **private** ip address of the IIS VM.
3. Open a browser on the other VM and navigate to the default page on the IIS server.
4. Download the [CtsTraffic](https://github.com/Microsoft/ctsTraffic/tree/master/Releases/2.0.1.4) tool to a directory on each server. CtsTraffic is an open source, Microsoft written, high performance tool for generating all kinds of test traffic.
5. Open a powershell window and run a ping command on each VM. This will provide an RTT sample giving us a good estimate of the latency.

PS>ping -t 10.3.1.8

Pinging 10.3.1.8 with 32 bytes of data:

Reply from 10.3.1.8: bytes=32 time<1ms TTL=128

Reply from 10.3.1.8: bytes=32 time=2ms TTL=128

1. On the VM with the IIS start CtsTraffic in server mode:

PS C:\Users\dahavey\Documents\Tools\CtsTraffic> .\ctsTraffic.exe -listen:0.0.0.0 -transfer:1000000000000

Configured Settings

-----------------------

Protocol: TCP

Options: InlineIOCP

IO function: iocp (WSASend/WSARecv using IOCP)

IoPattern: Push <TCP client send/server recv>

PrePostRecvs: 1

PrePostSends: 1

Level of verification: Connections & Data

**Port: 4444**

Buffer used for each IO request: 65536 [0x10000] bytes

Total transfer per connection: 1000000000000 bytes

Accepting connections on addresses:

0.0.0.0:4444

Accepting function: AcceptEx

Server-accepted connections before exit : 0xffffffffffffffff

1. On the VM with the browser run CtsTraffic in client mode

PS>.\ctstraffic.exe -target: 10.3.1.7 -transfer:1000000000000 -connections:1000 -consoleverbosity:1 -statusupdate:1000 -Port:4444 -Pattern:Push -congestionalg:default

Configured Settings

-----------------------

Protocol: TCP

Options: InlineIOCP TCP\_CONGESTION\_ALGORITHM(0)

IO function: iocp (WSASend/WSARecv using IOCP)

IoPattern: Push <TCP client send/server recv>

PrePostRecvs: 1

PrePostSends: 1

Level of verification: Connections & Data

**Port: 4444**

Buffer used for each IO request: 65536 [0x10000] bytes

Total transfer per connection: 1000000000000 bytes

Connecting out to addresses:

13.66.216.239:4444

Binding to local addresses for outgoing connections:

0.0.0.0

Connection function: ConnectEx

Connection limit (maximum established connections): 1000 [0x3e8]

Connection throttling rate (maximum pended connection attempts): 1000 [0x3e8]

Total outgoing connections before exit (iterations \* concurrent connections) : 0xffffffffffffffff

Legend:

\* TimeSlice - (seconds) cumulative runtime

\* Send & Recv Rates - bytes/sec that were transferred within the TimeSlice period

\* In-Flight - count of established connections transmitting IO pattern data

\* Completed - cumulative count of successfully completed IO patterns

\* Network Errors - cumulative count of failed IO patterns due to Winsock errors

\* Data Errors - cumulative count of failed IO patterns due to data errors

TimeSlice **SendBps** RecvBps In-Flight Completed NetError DataError

0.036 0 0 0 0 0 0

1.042 250092151 29423 970 0 0 0

2.047 196542790 22089 970 0 0 0

3.052 164329074 19880 970 0 0 0

4.043 140594890 0 970 0 0 0

1. Make sure that the sendBps rate has settled at its maximum.
2. On the VM with the IIS observe the ping time. It should look something like this:

PS>ping -t 10.3.1.8

Pinging 10.3.1.8 with 32 bytes of data:

Reply from 10.3.1.8: bytes=32 time<1ms TTL=128

Reply from 10.3.1.8: bytes=32 time<1ms TTL=128

Reply from 10.3.1.8: bytes=32 time<1ms TTL=128

Reply from 10.3.1.8: bytes=32 time<1ms TTL=128

Reply from 10.3.1.8: bytes=32 time<1ms TTL=128

Reply from 10.3.1.8: bytes=32 time<1ms TTL=128

Reply from 10.3.1.8: bytes=32 time<1ms TTL=128

Reply from 10.3.1.8: bytes=32 time<1ms TTL=128

Reply from 10.3.1.8: bytes=32 time<1ms TTL=128

Reply from 10.3.1.8: bytes=32 time=30ms TTL=128

Reply from 10.3.1.8: bytes=32 time=88ms TTL=128

Reply from 10.3.1.8: bytes=32 time=173ms TTL=128

Reply from 10.3.1.8: bytes=32 time=297ms TTL=128

Reply from 10.3.1.8: bytes=32 time=320ms TTL=128

Reply from 10.3.1.8: bytes=32 time=363ms TTL=128

Reply from 10.3.1.8: bytes=32 time=598ms TTL=128

Reply from 10.3.1.8: bytes=32 time=747ms TTL=128

Reply from 10.3.1.8: bytes=32 time=574ms TTL=128

Reply from 10.3.1.8: bytes=32 time=365ms TTL=128

Reply from 10.3.1.8: bytes=32 time=448ms TTL=128

Reply from 10.3.1.8: bytes=32 time=722ms TTL=128

Reply from 10.3.1.8: bytes=32 time=842ms TTL=128

Reply from 10.3.1.8: bytes=32 time=830ms TTL=128

Reply from 10.3.1.8: bytes=32 time=828ms TTL=128

Reply from 10.3.1.8: bytes=32 time=899ms TTL=128

Reply from 10.3.1.8: bytes=32 time=837ms TTL=128

Reply from 10.3.1.8: bytes=32 time=787ms TTL=128

Request timed out.

Request timed out.

1. Wait until the latency reaches its highest levels and on the other VM use the browser to navigate to the IIS server default page. (On my system it didn’t work at all. In fact, the VM was completely immobilized.)
2. Use control-C to kill the CtsTraffic server (the one on the IIS VM), wait for the queues to drain and the latency to return to normal (typically <1ms)
3. Use powershell to configure LEDBaT on the VM with the browser (not the one with the IIS).

PS>NetTCPSetting -SettingName DatacenterCustom -CongestionProvider LEDBAT

PS>Get-NetTCPSetting -SettingName DatacenterCustom

SettingName : DatacenterCustom

MinRto(ms) : 20

InitialCongestionWindow(MSS) : 10

CongestionProvider : LEDBAT

CwndRestart : True

DelayedAckTimeout(ms) : 10

DelayedAckFrequency : 2

MemoryPressureProtection : Enabled

AutoTuningLevelLocal : Normal

AutoTuningLevelGroupPolicy : NotConfigured

AutoTuningLevelEffective : Local

EcnCapability : Enabled

Timestamps : Disabled

InitialRto(ms) : 3000

ScalingHeuristics : Disabled

DynamicPortRangeStartPort : 49152

DynamicPortRangeNumberOfPorts : 16384

AutomaticUseCustom : Disabled

NonSackRttResiliency : Disabled

ForceWS : Disabled

MaxSynRetransmissions : 2

AutoReusePortRangeStartPort : 0

AutoReusePortRangeNumberOfPorts : 0

1. Use powershell to configure LEDBAT on port 4444 (CtsTraffic default):

### Next configure TCP to use the DatacenterCustom template for all connectons with a localport of 4444

PS>New-NetTransportFilter -SettingName DatacenterCustom -Protocol TCP -LocalPortStart 0 -LocalPortEnd 65535 -RemotePortStart 4444 -RemotePortEnd 4444

SettingName       : DatacenterCustom

Protocol          : TCP

LocalPortStart    : 0

LocalPortEnd      : 65535

RemotePortStart   : 4444

RemotePortEnd     : 4444

DestinationPrefix : \*

1. Observe the latency. It should be much lower. Something like this:

Reply from 10.3.1.8: bytes=32 time=1ms TTL=128

Reply from 10.3.1.8: bytes=32 time<1ms TTL=128

Reply from 10.3.1.8: bytes=32 time<1ms TTL=128

Reply from 10.3.1.8: bytes=32 time<1ms TTL=128

Reply from 10.3.1.8: bytes=32 time=10ms TTL=128

Reply from 10.3.1.8: bytes=32 time=18ms TTL=128

Reply from 10.3.1.8: bytes=32 time=26ms TTL=128

Reply from 10.3.1.8: bytes=32 time=32ms TTL=128

Reply from 10.3.1.8: bytes=32 time=47ms TTL=128

Reply from 10.3.1.8: bytes=32 time=50ms TTL=128

Reply from 10.3.1.8: bytes=32 time=49ms TTL=128

Reply from 10.3.1.8: bytes=32 time=57ms TTL=128

Reply from 10.3.1.8: bytes=32 time=54ms TTL=128

1. Now use the browser to navigate to the IIS default page. It should appear quickly. That is the power of LEDBaT to alleviate user frustration.

Thank you for evaluating LEDBaT. Please leave a comment on the blog.

Daniel ‘low-latency’ Havey.