# TB640 Streamserver hardware guide

**Document number**
9010-00021-1D

**March 2012**

This page in intentionally left blank

# Table of contents

This page in intentionally left blank

# How to choose appropriate stream server hardware

While developing the TMedia hardware and the tbstreamserver software, we have performed several performance tests. According to the results of these tests, we have been able to determine guide-lines for choosing appropriate stream server hardware according to operating condition requirements.

The current document presents the results of our tests. However, we have not performed tests with a wide variety of hardware, thus we encourage our customers to follow the guide-lines presented in this document. We also encourage customers that wish to try other hardware solutions to run performance tests and share with us the results of those tests[1].

# *Typical performance bottle-necks*

## Available RAM

The tbstreamserver application requires some amount of **physical** RAM available on the host to cache audio data before it is being streamed.

- Playing files: The tbstreamserver uses double-buffering when playing files from disk. It plays one piece of the file while it loads the subsequent from disk. The amount of RAM will thus depend on the number of simultaneous playing files. Each playing file uses two buffers of 64KB, 128KB or 256KB (according to configuration).
- Transcoding files: When playing or recording audio files require transcoding (from input format to aLaw/uLaw packets), tbstreamserver uses a transcoding buffer. The default transcoding buffer size is 16KB per channel.
- Recording files: The tbstreamserver uses double-buffering when recording files to disk. It accumulates packets in a RAM buffer, while the other buffer is written to disk. The amount of RAM will thus depend on the number of simultaneous recording files. Each recording file uses two buffers of 64KB, 128KB or 256KB (according to configuration).
- RAM Cache: The tbstreamserver will use any remaining RAM to implement RAM caching of the most recently played files. This reduces the number of disk access for frequently accessed files. The more RAM is available, the longer the file will remain in the RAM cache.

## Available disk space

The tbstreamserver application saves audio files in aLaw or uLaw format on the host’s hard drive. This uses 8KB of disk space per second of audio stored. This allows about 4 years worth of audio on a single 1TB hard drive. The maximum number of hard drive per host depends on the server hardware.

The tbstreamserver application can also play/record compressed audio files to save disk space, and perform live compression/decompression. However this increases the CPU usage on the host. The only codec that offers reasonable performance is the VOX codec, which offers compression ratio of 2:1, but uses little CPU so that a Quad core Intel Core i7 CPU (for example) can play or record more than 16,000 channels with live VOX transcoding.

## Disk access performance

When playing audio files from disk, the tbstreamserver needs to regularly read data to disk for each playing channel. As each hard drive can provide a limited number of read access per second, this will limit either the number of simultaneously playing files, or the number of “calls per second”.

This bottle-neck can be solved by distributing files to play over multiple hard drives, using a good disk controller.

Using a SSD drive also solves any disk access performance problem.

## Note about using RAID

Please note that RAID should be used with caution with the tbstreamserver, as inappropriate configuration may cause much lower performance.

We recommend using RAID-Level-1 (mirroring) for redundancy, at the cost of 2x disk storage requirements.

If using RAID-Level-0 (striping), the size of the RAID stripes MUST exactly matche the size of the audio “chunks” used with the tbstreamserver (64KB, 128KB or 256KB), otherwise each disk access from tbstreamserver will require to move the two drives, thus reducing performance by 2x (no more faster than one drive). We don’t recommend using RAID-Level-0, as it provides no disk redundancy (on the opposite, if one of the drive fails, all data is lost!).

If using RAID-Level-5 (parity redundancy), the size of the RAID stripes MUST exactly matche the size of the audio “chunks” used with the tbstreamserver (64KB, 128KB or 256KB), otherwise each disk access from tbstreamserver will require to move multiple drives, thus reducing performance by 2x, 3x or even more. ).

See below in this document for more information about RAID performance with tbstreamserver.

## tbstreamserver audio database

The tbstreamserver application can use its own private database (stored on disk as one big file) to store audio files. This audio database has been optimized to avoid disk fragmentation, and provide much better read performance (about 1.5x) and write performance (more than 2x) than using separate audio files on the file system.

However, managing files inside that audio database is less convenient than managing standard files on the file system. We thus recommend using SSD drives, which performance is so high that usage of tbstreamserver’s audio database becomes irrelevant.

## Ethernet card performance

We recommend running the tbstreamserver application on a host with a Gigabit Ethernet network card (and Ethernet switches). In fact, 100Mbps network cards will limit the number of simultaneous channel to a low number (around 1000 channels).

Multiple Gigabit Ethernet cards can be used in the same host for load sharing, since the performance of the tbstreamserver software allows playing much more than one Gigabit worth of audio packets.

## Network performance

It is mandatory to design the Ethernet network architecture to make sure there exist no bottleneck between the tbstreamserver hosts and the TMedia units. In fact, all Ethernet switches used must be Gigabit. Uplinks between Ethernet switches must be avoided (tbstreamserver hosts should be connected to the same Ethernet switches as the TMedia units). When this is impossible, the uplinks throughput should be high enough for the usage (possibly using link aggregation).

We recommend building a network redundant setup by doubling the number of Ethernet cards in the host server, and building two separate Ethernet networks to reach the two ports of the TMedia units to play/record audio with.

## Host CPU

The host CPU is generally not the bottleneck for the tbstreamserver. Other bottlenecks will be reached before CPU capacity is exhausted (Ethernet bandwidth, Disk performance, PCI-express bus bandwidth, etc).

CPU usage will be higher when playing non-aLaw (or uLaw) audio files, and thus requiring audio transcoding. It will also be higher when mixing audio from multiple files into one output channel.

## *Parameters that impact the bottlenecks*

There are a few parameters that will affect the various performance bottlenecks of the tbstreamserver application.

### tbstreamserver audio “chunk” size

The tbstreamserver reads files from disks in “chunks” of customizable size. The size of these “chunks” has an impact both on the RAM requirements, and on the disk access performance requirements.

Smaller audio “chunk” size will decrease the RAM requirements for playing files from disk, but will increase the number of disk access, thus increase the number of hard drives required to reach the required performance.

Bigger audio “chunk” size will do the opposite (increase RAM requirements; decrease the disk access performance requirements).

The tbstreamserver application allows the following audio “chunk” sizes: 64KB (8 seconds), 128KB (16 seconds) or 256KB (32 seconds). The ideal value will vary according to the host hardware and target application. For example, RAM vs. Hard drive cost may be taken in consideration. The maximum RAM and maximum number of hard drives in one system can also have an impact on this setting.

When using SSD drives, a “chunk” size of 64KB is ideal.

### Packets duration

The tbstreamserver application supports packets duration of 20ms, 40ms, 80ms or 160ms.

However, 160ms packets should always be used. This will not increase the playback delay if the jitter-buffer on the TMedia unit is configured to an appropriate value, such as 50ms.

All the performance measurements in this document are based on 160ms packets.

## *Identifying the bottlenecks*

The requirement on the various bottlenecks of the tbstreamserver application should be calculated as follow.

### Available RAM

Total RAM requirements for the host should be calculated as follow:

- Each playing audio file requires **2 x “chunk size” KB** of RAM
- The tbstreamserver application requires about **32MB** for it’s own usage
- The OS generally requires about **1GB** for its own usage
- tbstreamserver is available in 64 bits version for Linux, if more than 2GB of RAM are required.

### Available disk space

Total disk space requirements for the host should be calculated as follow:

- Each audio file requires **8KB per second**

### Disk access performance

If you are using SSD drive, please skip this section, it’s irrelevant due to ultra-high performance of SSD drives.

Total disk access per second requirement for the host should be calculated as follow:

- Each file play request requires tbstreamserver to open the file path on the file system, which can require a couple of disk access (up to 10), though the file system may cache that in to RAM for frequently played files.
- While they are playing or recording, each channel requires one disk access every “chunk” duration (64KB chunks contain 8 seconds of audio, for example)[2]
- Here are typical numbers:
    - With 64KB audio chunks, each 7200 RPM hard drive is capable of about **150 accesses per second**, while high-end 15K RPM disks can provide more than 250 accesses per second.
    - With 256KB audio chunks, 7200 RPM drives reach about 100 accesses per second, while 15K RPM drives reach about 150 accesses per second.
- SSD drives:
    - SSD drives offer a huge performance gain over hard drives, with performance of often more than 40,000 access per second.

[2] If recording directly to disk (not using tbstreamserver’s internal database), each chunk write actually requires two disk access, one for the chunk data write, one for the OS to update it’s FAT.

- Thus this limits the number of play or record channels per drive to “**disk access per second” multiplied by “chunk duration in second”.**

## Disk access performance with RAID-Level-1

If using RAID-Level-1 (mirroring), the read performance will be calculated using the total number of drives (as each of the two mirrored drives can simultaneously read a different file). However the write **performance will be calculated using the total number of drive pairs** (as each write must be done simultaneously on the two drives).

With RAID-Level-10 (combining multiple pairs of RAID-Level-1 drives), performance is multiplied by the number of drive pairs.

## Disk access performance with RAID-Level-5

If using RAID-Level-5 (parity redundancy), the **read performance will be calculated using the total number of** drives (as each of the drives can simultaneously read a different data chunk). However the **write performance will be calculated using the total number of drive divided by two** (as parity must be updated for each chunk written).

## Ethernet card performance

One typical Gigabit Ethernet card is capable of sending packets for about 8,000 audio channels (this includes a good comfort margin compared to maximum potential bandwidth).

## Network performance

We recommend that all the TMedia Units which one tbstreamserver host is playing to are connected to the same Gigabit Ethernet switch. This avoids causing traffic on the uplink.

The network bandwidth requirements are about 67Kbps per channel, overhead included.

## *Method to identify the bottlenecks*

As we just explained above in this document, there are multiple bottlenecks that can limit the tbstreamserver performance, and it is not trivial to choose the appropriate hardware configuration of a given application.

The current section of this document attempts to describe simple steps that allow identifying the main bottlenecks in order to choose the most appropriate hardware. Please not that these calculations are not exact and are somewhat based on statistical averages. We thus recommend to choose hardware that offers some performance margin above the numbers calculated with the method below (25-50% margin should be appropriate).

Do not forget to also take into account that performance will not scale infinitely. In fact, though two hard drives will probably be twice faster as one, 20 hard drives will definitely not be twice faster than 10, due to other bottle-necks in the system. This also holds for the number of Gigabit Ethernet ports.

### The parameters you need to know

1. **TotalFilesSize**: The total size of audio files required on disk. This value may change a lot according to the application (ring-back-tone, music playback, voice mail, prompts playback).
2. **SimultPlayFiles**, **SimultRecordFiles**: The total number of simultaneous channels required (files play, and files record).
   The total being **SimultChannels**.
3. Disk RAID mode.

### The bottlenecks calculation

1. Calculate the number of Gigabit Ethernet cards required, assuming one Ethernet card allows about 8,000 channels (play and record combined):
   **NbEthernetCards** = **SimultChannels** / 8,000 channels per card.
   Make sure that the host hardware can support that number of network cards (including PCI bottlenecks). Otherwise multiple hosts will be required.
2. Calculate the number of hard drives required to store these audio files, assuming 500GB drives:
   **NbDrivesForStorage** = (**TotalFilesSize** / 500GB)
   x 2 for RAID-1 setup
   x (N-1) for a N-drives RAID-5 setup.
3. Calculate the number of hard drives required for the required number of simultaneous channels (do the calculation for 64KB, 128KB and 256KB audio “chunks” size):
   a. **NbDiskAccessPerSecPlay** =
      (**SimultPlayFiles/ chunkDuration**).
   b. **NbDiskAccessPerSecRecord** =
      (**SimultRecordFiles/chunkDuration**).

   c. **NbDisks** =
      **(NbDiskAccessPerSecPlay** + (**NbDiskAccessPerSecRecord** x 2 if RAID) ) / nb access per second per disk.

4. Calculate RAM requirements (for 64KB, 128KB and 256KB audio “chunks” size), assuming 256MB reserved for the operating system and 32MB for the tbstreamserver:
   **RamUsage** = 256MB (OS) + 32MB (tbstreamserver) + (2 x **(SimultPlayFiles+SimultRecordFiles)** x chunk size)
5. Determine the appropriate audio “chunk” size by choosing the most appropriate balance between RAM usage and number or disks required. Make sure that the host hardware can support that number of disks, or that amount of RAM.

# Hardware recommendations

## RAM:

Make sure the host has plenty of RAM (for the operating system, for all running applications, and for tbstreamserver calculated with the number of simultaneous file play to reach).

Don't hesitate adding more RAM than necessary, this is a cheap resource that can greatly help with performance. More RAM also means less disk access (due to RAM caching) which will reduce hard drive failures (no effect on SSD drives however).

## Hard drives:

We recommend using SSD drives on a fast bus (SATA 3Gbps, 6Gbps or faster).

If you need more storage capacity than what SSD drives can offer, we recommend using server grade 10,000 RPM drives that are rated to last for a long time under heavy load.

We do not recommend using RAID-level-0 due to lack of redundancy.

We recommend using RAID-level-5 for redundancy, **as long as the RAID stripe size matches the tbstreamserver's audio chunk size**.

## Ethernet cards:

We recommend using **Gigabit Ethernet cards** (two if network redundancy is required, one otherwise) that are connected to the CPU through **PCI-Express** or faster BUS.

We DO NOT recommend using 100MBps network adapters.

We highly recommend using **1Gbps network switches** (two switches on separate subnets for network redundancy). Try using switches that have large amount of internal buffer memory for packet queues.

NEVER USE NETWORK HUBS, NEVER MIX DIFFERENT SWITCH SPEEDS (1Gbps connected to 100Mbps, for example).

If stream traffic has to be routed between separate Gigabit switches, the uplink between these switches MUST support the total aggregated bandwidth of all ports of the switches used for streaming.

## CPU:

We recommend using Intel Core2 CPUs or faster (like Intel core i5, i7). Large number of CPU cores won't be required unless live transcoding is used.

***IMPORTANT***: On Windows, DO NOT CHANGE THE PRIORITY OF THE STREAM SERVER using Windows Task Manager (or other tool). When the priority is changed, the relative priority of the threads inside the application seems to be lost. The stream server will work properly only if relative priority of threads (relatively to each other) inside the stream server application are not modified.

***IMPORTANT***: On Solaris and Linux, it is mandatory that the user launching the application has the privileges to change threads scheduling to real-time at system-scope. Only the root user has these privileges by default.

## Load balancing:

The stream server API easily allows an application to perform load balancing. Using the stream server API, any application can simultaneously query multiple stream servers for the current load, or availability of a file.

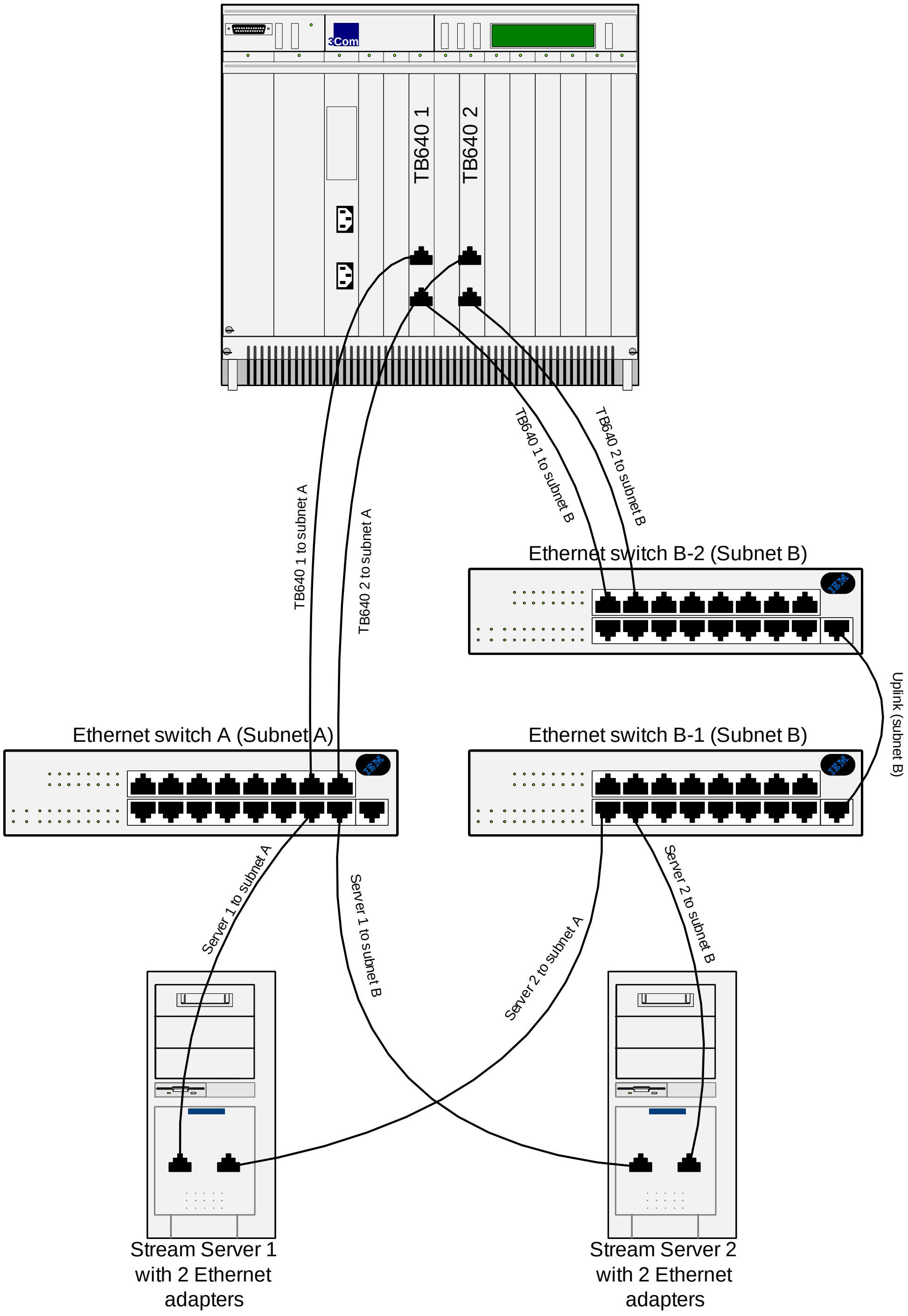
Using the information returned by every available stream server, the application can choose the stream server that has the least load to perform the next playback. The “amount of load” can be determined by looking at the number of active channels and the number of disk access pending, and comparing those values with the capacity of the server.

Because all the servers run on the same network and are all connected to every TMedia unit through the Gigabit Ethernet switches, a properly designed network will allow any stream server to handle a play on any port of any TMedia unit, thus allowing the controlling applications to easily implement load balancing.

The tbtoolpack software will automatically handle load balancing to multiple tbstreamservers.

# Example network organization

When designing the network for a streaming application between stream servers and TMedia unit, it is important to make sure the network has no bottleneck.
The following diagram shows an example network that allows two stream servers to feed two TMedia units with network redundancy.



End of the document