EZEL: a Visual Tool for Performance Assessment of Peer-to-Peer File-Sharing Networks

Lucian Voinea [*]
Technische Universiteit Eindhoven

Alex Telea [†]
Technische Universiteit Eindhoven

Jarke J. van Wijk [‡]
Technische Universiteit Eindhoven

ABSTRACT

In this paper we present EZEL, a visual tool we developed for the performance assessment of peer-to-peer file-sharing networks. We start by identifying the relevant data transferred in this kind of networks and the main performance assessment questions. Then we describe the visualization of data from two different points of view. First we take servers as focal points and we introduce a new technique, faded cushioning, which allows visualizing the same data from different perspectives. Secondly, we present the viewpoint of files, and we expose the correlations with the server stance via a special scatter plot. Finally, we discuss how our tool, based on the described techniques, is effective in the performance assessment of peer-to-peer file-sharing networks.

CR Categories: H.5.2[User Interfaces]: Evaluation/methodology; I.3.2 [Graphic Systems]: Stand-alone systems; J.7 [Computers in Other Systems]: Command and control

Keywords: process visualization, distributed file systems visualization, P2P file-sharing networks visualization, small displays

1. INTRODUCTION

Process visualization is one of the oldest forms of information visualization. It appeared once with the need of gaining insight in the behavior of a system, and it dates back in time to the ancient builders of the Stonehenge, which used the temple as a 'visualization instrument' for the succession of seasons. The appearance of the graphic display computer marked the birth of a plethora of process visualization techniques [1,4,5,7]. These techniques address different domains, from the visualization of application behavior [4] to the visualization of web site accesses [5]. Visualization of distributed systems' performance is, however, one of the less explored domains. Most work in this area is related to the visualization of the structure of such systems [2,3].

We propose a new approach to the visualization of performance of Peer-to-Peer (P2P) file-sharing networks, a branch of distributed processing that has recently gained enormous popularity. We illustrate the proposed visualization techniques by a prototype tool, called EZEL, which we developed for the assessment of performance in the *ED2K* P2P file-sharing network [11]. We first present the issues that are relevant for the assessment of performance in distributed processing systems, with a focus on P2P file-sharing networks (Section 2). In Section 3, we describe the data that is transferred in this kind of systems, and we identify the transactions that are important for performance evaluation.

[*]e-mail: lvoinea@win.tue.nl
[†]e-mail: alext@win.tue.nl
[‡]e-mail: vanwijk@win.tue.nl

IEEE Symposium on Information Visualization 2004
October 10-12, Austin, Texas, USA
0-7803-8779-1/04/$20.00 ©2004 IEEE

Next, we detail the challenges that arise when supporting the assessment with visual tools, and we present our approach to address them.

In Section 4, we describe the visualization of data taking servers as focal points. We show how, via the use of shading and color, multiple aspects can be shown simultaneously in a compact way. We elaborate on the space partitioning power of cushions, and we introduce a novel technique: fading cushions. We demonstrate how this technique allows visualizing the same data from different perspectives. In Section 5, we add the viewpoint of the file, and in Section 6 we expose the correlation between file and servers via a special scatter plot.

Finally, we discuss in Section 7 the suitability of our approach for the assessment of P2P file-sharing networks, and we conclude by outlining future research directions.

2. PROBLEM DESCRIPTION

A distributed processing system is a collection of entities whose purpose is to reduce the overall processing time for a given task by dividing the processing load among its constituent parts. We outline the most important concepts in such a system, with an eye on their implementation in a P2P file-sharing system (see Figure 1 for a conceptual model).

Clients generate requests (e.g., file read requests) and assign them to proxy entities. A *proxy* divides requests in smaller parts (i.e., segments) that are uniquely identifiable and can be independently fulfilled by *server* entities.

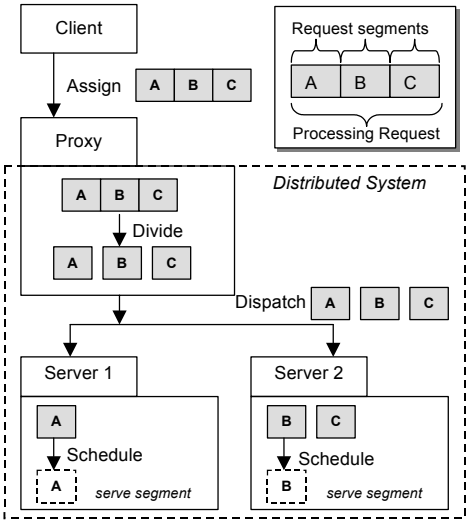

Figure 1: Distributed processing system (conceptual model)

Every proxy has an internal dispatcher algorithm that decides to what servers the requested segments will be sent for processing. Every server has limited processing resources to handle request segments from proxies, and uses a priority based scheduling to manage them. The priorities are internally maintained by the server for each client request.

Visualization of a distributed system's performance aims at helping the user to understand such a system, based on information obtained from transactions between its constituent parts. Both snapshots and history recordings are therefore important [1].

The user can employ this understanding to navigate the transaction data and answer a number of performance related questions. In the case of our distributed file-sharing systems, one is mainly interested in two issues, as follows.

Dispatcher algorithm assessment

When the network of processing servers is large and dynamic (e.g., P2P networks), the segment dispatching algorithm has a strong influence on the request servicing time. The performance visualization should help users to easily assess the dispatcher algorithm, and reveal the factors and the circumstances that might influence it. For example, users should be able to identify the reasons for which a slower server is selected at a certain moment instead of a faster one.

Server assessment

When the dispatcher algorithm on the proxy allows direct selection of the servers, performance visualization should help to determine which server delivers the best value. The interesting case appears when the selection is based on a number of independent performance figures. The most important questions and quantities relevant to P2P networks are:

- *download speed*: how long does it take till one gets a requested file?
- *server popularity*: how long do clients wait in the server-side queue, and how frequently do other clients with higher priority enter that queue?
- *server specialization*: what kind of requests can a server satisfy?

When assessing the performance of a P2P file-sharing network, one has to investigate the evolution of a number of independent parameters. An effective assessment should consider the loosely coupled parameters together, and should be based on tradeoffs that depend on the purpose of the assessment. The very nature of tradeoff making requires the user to divide its focus over more assessment criteria at once. This turns out to be rather difficult when the number of criteria becomes higher than two. A typical download session for a 700 MB movie file contains around 200,000 transactions. If one uses just standard time graphs to visualize the above three quantities, the overall image is quickly lost, and the dispatcher algorithm and server assessment questions remain unanswered. The challenge is to build a unified visualization, in which the user can focus on a particular quantity of interest without losing overview.

For P2P file sharing networks, we use four main criteria to assess a server:

- download speed (higher is better)
- size of segments (larger is better)
- queue evolution (fast advance and less re-queuing after admittance is better)
- segment position (depending on the download purpose, some segments may be more important than others)

The ideal server should be fast, able to provide large contiguous segments, and should have a small waiting time. Additionally, it should not be very popular, to reduce the chance that other clients with a higher priority interrupt the download by acquiring the server. However, such servers usually do not exist. Moreover, the assessment depends on several characteristics of the downloaded file, as explained next. For the fast download of a small file, such as a 3 MB MP3 music file, selecting the fastest server may not be the most appropriate decision. When the waiting time in the queue of the fast server exceeds the time that another slower server requires to perform the task, we prefer the slower server. Another example is the download of an archive, (e.g., a ZIP file). Such a download should not be attempted from a server providing fragmented segments, even if it is fast. A slower server that provides contiguous segments is preferred, as it makes archive recovery simpler when the download cannot be completed.

In the following sections, we walk through the challenges of building a visualization tool for P2P file-sharing networks. We illustrate our solutions with snapshots from EZEL, a visualization tool that we developed for the performance assessment of the popular *ED2K* P2P network. A copy of the tool and example datasets may be downloaded from

http://www.win.tue.nl/~lvoinea/Ezel.htm.

3. DATA MODELING

The first issue we have to consider when building a visualization tool is which data to visualize. P2P file-sharing networks are characterized by a large number of terminals connected via the Internet. Each terminal connected to such a network can act both as a server and as a client in the same time. Clients generate file read requests that proxies break down into segment requests. A segment request is fulfilled by a single server, which provides the client with the related file segment. A file segment consists of file blocks and has a variable size (expressed in blocks).

All terminals in the network exchange transactions based on a specific protocol. These transactions may contain either file blocks, or control information (e.g., download requests, file availability info, queue evolution info). In the case of the *ED2K* network, the exact protocol in use is not disclosed, which makes our assessment task considerably more difficult.

As mentioned in the previous section, server and dispatcher algorithm assessment are central issues for performance evaluation of P2P file-sharing networks. We address these issues by analyzing the transaction data that a client exchanges with the rest of the network.

To study the dynamic behavior of servers, we record two types of transaction events: *file block arrivals* and *queue position reports*. With this information, we build three functional descriptions for a server, from the point of view of a given client. In the following, we consider that a client is serviced by *NS* servers $S_1,...,S_{NS}$, every server S_i being identified by an integer server id. The download time t runs from 0 to the download completion moment T_C. The three server descriptions are:

Queue position: $Q(S_i,t): N \times R \rightarrow N$

Gives the position of the client segment request in the queue of server S_i at time t. If $Q(S_i,t)$ is zero, the client can start downloading from S_i.

Download Speed: $V(S_i,t): N \times R \rightarrow R$

Gives the speed with which the client receives data from the server S_i at time t.

Contribution: $C(S_i,t): N \times R \rightarrow N$

Gives the data downloaded from a server S_i from the beginning till a given time t. In other words:

$$C(S_i,t) = \int_0^t V(S_i,\tau)d\tau$$

The total amount of downloaded data is thus:

$$D = \sum_{i=1}^{NS} C(S_i, T_C)$$

To assess the performance of the dispatcher algorithm, one has to consider both the server assessment and the evolution of the downloaded file itself. For that, we record the *block arrival* events and correlate them with the *file segment requests*. With this information, we construct three functional descriptions of a download:

Provider: $P(p): N \rightarrow N$

Gives the server that provided the block at a position *p*, for all positions *p* in a downloaded file

Time of Arrival: $T(p): N \rightarrow R$

Gives the moment when the client received the block at position *p*, for all positions *p* in a downloaded file.

Segment: $S(p): N \rightarrow N$

Gives the file segment to which the block at position *p* belongs to, for all positions *p* in a downloaded file.

The quantities mentioned above are discrete. For example, a typical movie download consists of around 200,000 time moments *t*, *NS*=150 servers, and a total downloaded value of *D*=700 MB.

All above functional descriptions are equally important for the performance evaluation of our P2P file-sharing network. Consequently, the challenge we face is to build a visualization that facilitates access to all of them and shows how they relate to each other.

We want to assess the dynamic behavior of individual servers, view how a file is downloaded, and see the relation between these processes. Since the functional descriptions to be visualized have several implicit, non-trivial dependencies, we find a straightforward visualization (for instance using separate graphs) not a good solution.

Given that our set of functional descriptions has three main axes (Servers S_i, Time *t*, block Position *p*), a visual representation using a 3D scatter plot may appear to be a direct solution. Figure 2 depicts such an approach. Every dot represents the transmission of a block from a server at a certain moment. However, this visualization would be very hard to interpret, given the large amount of time samples (hundreds of thousands), the inherent 3D occlusion problems, and the data scattering.

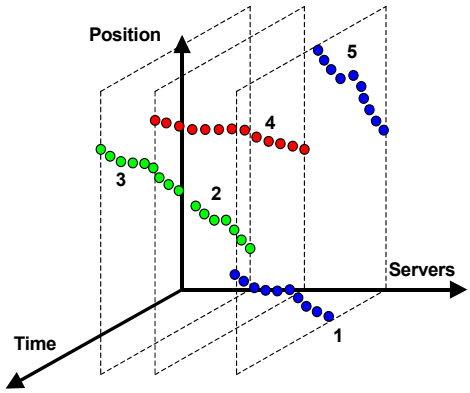

Figure 2: 3D visualization for P2P performance assessment

Therefore, we split the visualization in two parts (one focusing on servers, the other on the downloaded file) and we correlate them using a scatter plot. The server visualization is described next. The downloaded file visualization is described in Section 5. Finally, Section 6 presents the custom made scatter plot.

4. SERVER VISUALIZATION

To support the assessment of servers with a visual representation, we use a horizontal sequence of small diagrams, one per server. This allows the user to easily compare the functional descriptions of different servers (i.e., *Q*, *V* and *C*). Additionally, the representation of each server should offer enough provisions to relate it to the visualization of the downloaded file (Section 5).

There are several alternatives for an individual server representation. The obvious choice is to use the horizontal axis for Time, the vertical axis for Queue (*Q*) and Contribution (*C*) and to display their variation as graphs (Figure 3.a). The Download Speed (*V*) can be estimated in this setup from the slope of *C*.

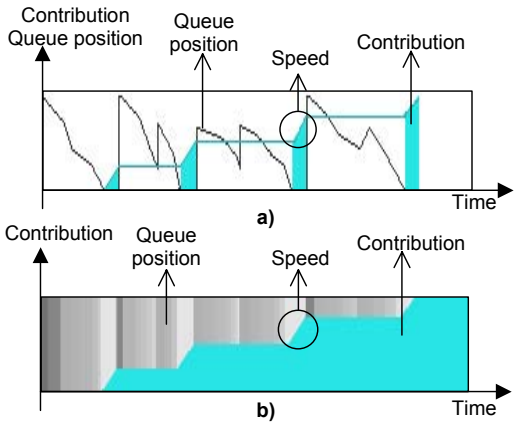

Figure 3: Server diagram, with graphs only a) with graph and luminance strips b)

However this first alternative is quite noisy for real world cases. Due to the mutual exclusion in time of downloading and queuing, the evolution of Queue position and Contribution are not continuous, but interleaved. To remove the noise from the visualization, we replace the spatial encoding of Queue position with a luminance encoding. We use rectangular strips whose gray shade indicates queue position (darker shades indicate lower positions). Although graphs are more precise, grayscale encoding of the queue position is sufficient for our purposes. After all, the user needs only to identify the overall position and to spot general queue trends such as advance or high / low position alternations.

Additionally, we use solid color filling for the area under the *C* graph to enhance the feeling of quantity that Contribution has. Figure 3.b depicts the result of the second approach. Both *C* and *Q* variations appear now continuous, which makes interpretation easier. Moreover, while their representations do no interfere, they still allow users to easily make correlations. The horizontal parts in the variation of *C*, for example, indicate periods in which the Contribution stagnated. The user can easily verify if queuing was the cause of idleness, and can also check the queue evolution of the segment request in that period. Similarly to the first approach, the Download Speed evolution can be estimated from the slope of *C*.

The next visualization design step is to arrange the server images such that they allow easy comparative assessment. For this, we need a way to easily distinguish and identify the diagrams. We use color encoding for that, and in each image we fill the area below the Contribution graph with a server dependent color. Color allows one to easily distinguish the different diagrams and also preserves server identity over changes in the diagram arrangement.

To allow easy comparison of the server diagrams, we need to arrange (sort) them along one of the spatially encoded axes, i.e., the Time axis or the Contribution axis. Using the Time axis for arranging the server images (Figure 4) proves to have two major drawbacks. First, it is hard to compare server quantities (queue position, contribution) at the same given time instant. For example, one could hardly decide if the contribution of a source exceeds that of another, at a given time t_0 (Figure 4).

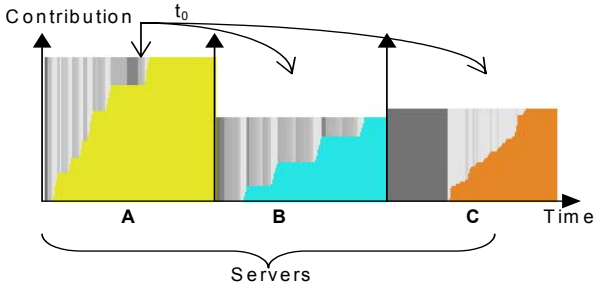

Figure 4: Server diagram arrangement along the Time axis

Secondly, the time interval (width of server diagrams in Figure 4) is identical for all servers, so no meaningful comparison could be made along the Time axis itself.

The second alternative (i.e., arrange on Contribution axis) is better, as it allows easy comparison of servers based on their total contribution (Figure 5).

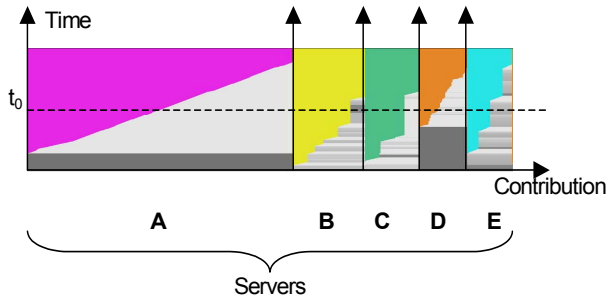

Figure 5: Arranging server diagrams along the Contribution axis

Additionally, for a given time t, this allows comparing the queue position and the cumulated contribution to that moment t.

Figure 6 presents a typical visualization obtained with the method presented so far: a file download served using five servers. We see that the first server (purple) is the most productive one: It gives about 50% of the total amount (half of the horizontal axis), has a stable throughput (constant slope), we are promptly getting on the first queue position, and we maintain this position for the total download duration (purple image slope has no step-like jumps, and its queue area has a constant light shade after we get on the first position). We can also identify in this image the less productive servers, i.e., the slow one (orange) and those exhibiting frequent falls in the queue position (yellow and cyan).

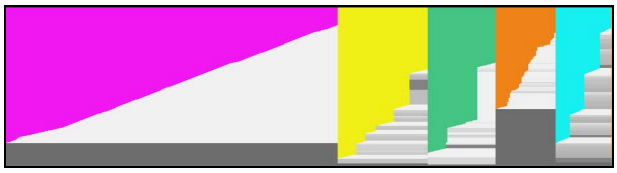

Figure 6: Basic server visualization

However, this visualization is still limited. First, using only color to encode server identity is not a good solution when the server arrangement (horizontal axis sorting) can change. It may happen that two servers with the same, or perceptually similar, colors are arranged one next to the other (Figure 7.a). Indeed, we wish to use only a few (10..16) perceptually different colors, whereas we typically have over 150 servers. Using only luminance (gray value) to encode the queue position causes similar problems. On the other hand, color encoding of server identity keeps visual coherence when rearrangement occurs.

4.1 Spatial partition with bi-level cushions

We solve the above problem using the space partitioning properties of cushions. For a detailed description of cushions, see [6]. As depicted in Figure 7, cushioning makes separation clear between different severs encoded with the same or similar colors, without using extra screen space. It also delineates the borders where the difference in luminance makes distinction hard.

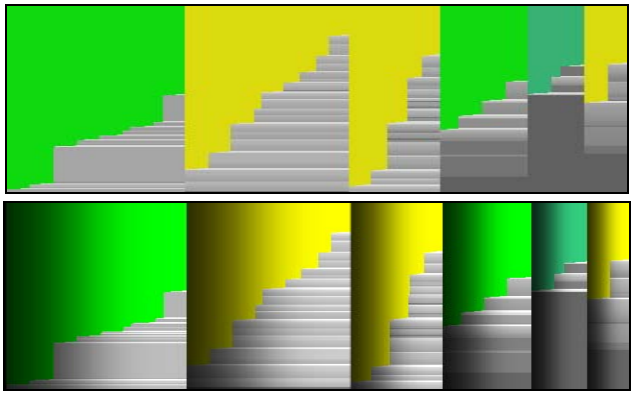

Figure 7: Server arrangement : without cushioning (top) with one level cushioning (down)

In the above server diagrams, the total contribution of a server consists of a set of segments. As the size of the segments varies, we would like to visualize it. That would be also useful later on for making correlations with the download visualization.

With the server visualization presented so far, it is hard to figure out the individual segments, as they are encoded using the same color (i.e., the color of the server). To emphasize the segment partitioning inside the diagram of a server, while maintaining clear separation between servers, we use the bi-level cushioning technique described by van Wijk and van de Wetering in [6]. Figure 8.a depicts the main idea behind this approach. By each server diagram we visualize the illumination of a height-modulated surface. The height assigned to a point in a server diagram is the sum of two parabolas (i.e., cushions), one that describes the server, and one that describes the segment to which the point belongs. The surface is illuminated using a spot light that forms an incidence angle α with the normal on the base plane. Each server diagram depicts the image projected by light reflection on a plane parallel with the base.

Figure 8.b depicts the result of this technique. By using OpenGL texturing, we obtain a much higher performance than the similar software-only implementation of van Wijk and van de Wetering [6]. In detail, we blend the server rectangle image and each of its segment rectangle images, as in Figure 7.(top), with a 1D texture containing the respective server or segment luminance profile in the alpha channel.

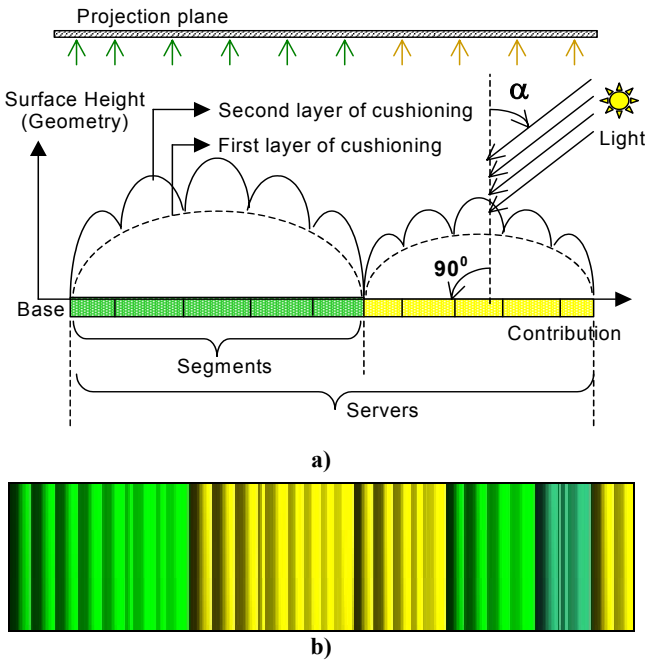

Figure 8: Bi-level cushioning for segment and source partitioning: a) principle; b) result

4.2 Focus migration with faded cushions

Using the bi-level cushioning is very effective for delimiting servers and segments within servers. However, the above method draws cushioned segment information also over the area that displays queue information (gray area in Figure 9) . Segment partitioning is not relevant for that area, and this makes server comparison based on queue evolution, i.e., following horizontal correlations, difficult.

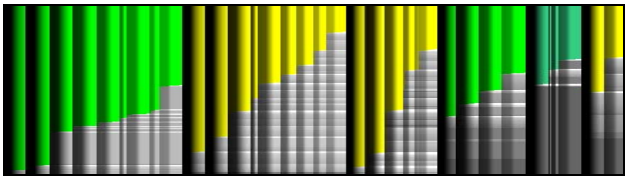

Figure 9: Basic bi-level cushion visualization

In order to maintain the desired segment and source partitioning effect, and, in the same time, remove the undesired influence on the queue evolution visualization, we extend our bi-level cushioning. We change the perceived shape of the segment cushions in the vertical direction from constant curvature to a gradually flattening profile. To achieve this, we introduce a height variation in the vertical direction using a decreasing profile as sketched in Figure 10.a. For this profile, we use an asymptotic function (e.g., the root of order n). The segment cushions are now efficiently implemented as 2D alpha textures and blended atop of the original 1D server cushions.

Eventually, we obtain a visualization that emphasizes both segment and server segregation at the top of the image, and then progressively focuses only on the partition in servers, as the user's focus moves to the bottom of the image. The gradual transition makes focus migration smooth while preserving the server context (Figure 10.b). In other words, the visualization exhibits vertical coherence at the top (segment-server area), which smoothly changes into horizontal coherence at the bottom (queue area). The overall visual effect resembles the draping of a curtain, and nicely scales up for visualizations containing over 100 servers and 1000 segments.

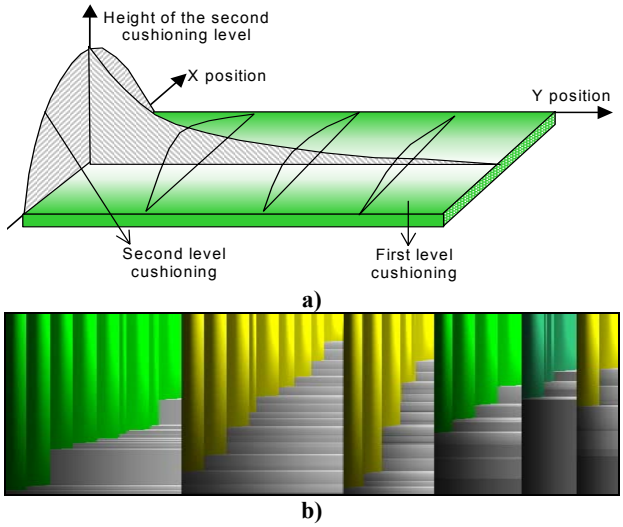

Figure 10: Enhanced bi-level cushioning for smooth focus migration: a) principle b) results

5. Download Visualization

In this section, we address the visualization of the download itself and the creation of correlations with the server visualization described in Section 4.

The only alternative in this part is to use the block Position as one of the main axes in the representation, and report the functional descriptions to it. The challenges are, however, in choosing the right visual encoding for the Provider (P), Time of Arrival (T) and Segment (S) descriptions. To make correlation with the server visualization easy, we use color to encode P, and we choose the same color assignment as for the server visualization.

For Segment encoding (i.e., S), we use a similar approach with the one from the server visualization: we build one-level cushions on top of fixed-width rectangles arranged along the Position axis (Figure 11). We don't need bi-level cushions, as the emphasis is only on segment segregation, and has to be visible along the entire width of the rectangles.

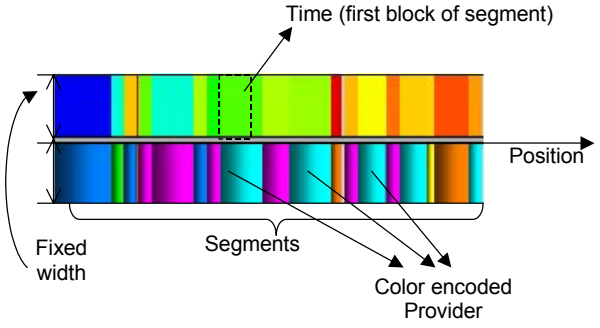

Figure 11: Visual encoding of functional description for a file download

For Time encoding, we may consider a graph-like representation. Neighboring segments on the Position axis,

however, may arrive at non-adjacent time intervals which immediately leads to a very noisy visualization. Therefore, we chose to use a rainbow colormap (t=0 is blue, t=T is red) to encode the time on a per segment basis (Figure 11). While this alternative is visually less accurate for identifying the arrival time of a block, it consumes little space and attenuates the visual noise caused by neighboring segments that arrive at different moments in time. Moreover, the above color scheme highlights discontinuities, i.e., segments that arrive at moments distant in time with respect to their neighbors. To improve the image generation speed, we don't report the time to every single block in a segment, as the *T* description specifies. Instead, we use for all the blocks in a segment the same time description as for the first block, and we try to implement a more accurate representation through server correlations, which we describe next.

6. CORRELATION VISUALIZATION

In this section we present the visualization component that allows making correlations between the server and download visualizations. In the design of the visualization so far, we have already a color-based correlation between the Provider description (i.e., *P*) and the server diagrams. This allows to identify and compare servers that provide some particular blocks in a downloaded file.

Next to this, we also need a correlation that would make the *T* description more accurate. Since the server visualization has a good mapping from Time to Contribution (i.e., *C*), we extend this mapping to the download visualization through a correlation along the block axes (i.e., the Contribution and the Position axes). However, given that the two axes are spatially encoded, a relation at block level would be too fine-grained and hard to visualize. For that reason, we choose to visualize the connections at the (higher abstraction) segment level.

The discrete nature of the block axes favors using a scatter plot representation to visualize the correlation. A simple scatter plot, however, makes visual associations difficult, once the number of segments is greater than 10 (Figure 12.a). A possible workaround is to add lines that make connections explicit. However, this alternative proves to be ineffective too, as it clutters the image, and suffers from aliasing once the distance between lines becomes too small (e.g., the black line in Figure 12.b). These problems are only aggravated by the large number of correlations (hundreds) that must be displayed for a standard download dataset.

In order to make the connections more explicit while keeping the image uncluttered, we replace the solid lines with shades that start from the points of the scatter plot and fade away as they approach the axes (Figure 12.c). This alternative reduces the confusion created by crossing lines, and offers still enough visual clues for recognizing connections. Additionally, it introduces no artifacts and scales very well with the image size. When the distance between the points of the scatter plot becomes to small to observe differences, the shades merge naturally, as if they were addressing the same element. To accomplish this, we draw the shades using OpenGL's GL_MIN blending function, which always keeps the darkest shade element at intersections, regardless of the shade drawing order.

The complete visualization, obtained after linking the server and download visualizations using the correlation methods described in this section, is depicted in Figure 13. For easy navigation, we added interactive selection facilities to allow restricting the download visualization part and the corresponding correlations to:

- specific parts of a file (by individual segment selection on Position axis)
- specific time intervals (using a time cursor on the Time axis).
- specific servers (by individual server selection on Contribution axis)

These selection mechanisms easily allow one to answer questions such as "which are the servers active at a given time moment", "which are the file blocks provided by a given server", and "which are the servers a given file part came from".

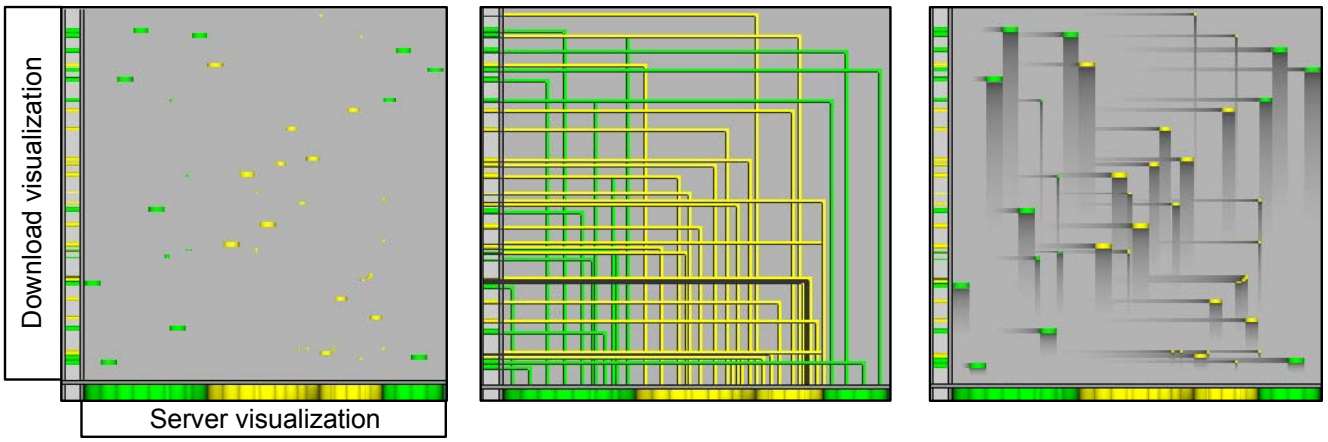

Figure 12: Correlation visualization alternatives a) basic scatter plot; b) adding connecting lines; c) adding shading

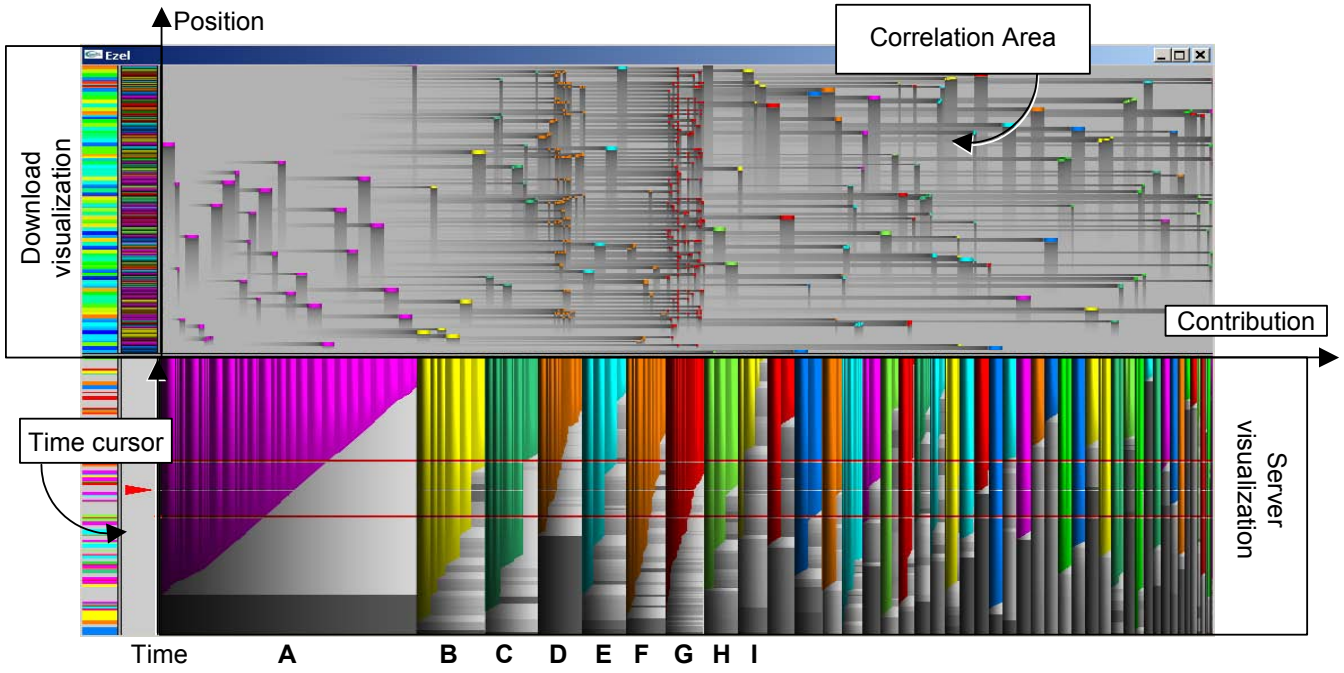

Figure 13: Visual tool for the assessment of performance in P2P file-sharing networks (EZEL snapshot)

7. DISCUSSION

In this section, we discuss how our P2P visualization tool EZEL can be used to answer the main performance related questions in distributed file-sharing networks.

In order to experiment with the tool, one needs real-life information about transactions in P2P file-sharing networks. We obtained such datasets by instrumenting *eMule* [10], an open source download client for the *ED2K* network. The instrumented client provides us with a log file from which the functions *Q,P,C,V,T,* and *S* discussed in Section 3 may be computed.

Figure 13 shows a visualization of the download of a large movie file (702,4MB). The complete download took several hours and contained 201,261 transactions. In this image, the servers are sorted in the decreasing order of their total contribution. The upper half of the image shows the segment fragmentation on a per server basis. We see that the most suitable download sources for archive files are **A**, **B**, **E** and **H**, as they provide large sets of contiguous segments, which makes archive recovery simpler in case of incomplete download. The least preferred in this sense are sources **D**, **F** and **G**, which provide tiny segments scattered along the entire length of the file. Analyzing the slopes in the image (i.e., the server speed) we see that **I** is one of the fastest sources. Unfortunately, it is also a very popular one, as most of the time our request waited in the server queue. A better alternative, especially for the download of a small file, is using servers **B**,**C**, **F** or **G**. Although **F** and **G** are slow and provide fragmented segments, they are unpopular, and thus start satisfying our requests very fast. Finally, if one were asked to single out an overall 'good' download source, **A** would qualify, as it gives us many data, with constant throughput, and little waiting time.

Figure 14 depicts a situation where we spotted a "weakness" of the dispatcher algorithm. For a downloaded file (350MB) we arranged the servers in decreasing speed order. In Figure 14.b, we switched off the display of segment evolution in time. Using a time cursor, we selected those segments that were downloaded at a certain moment t_0 close to the end of the download.

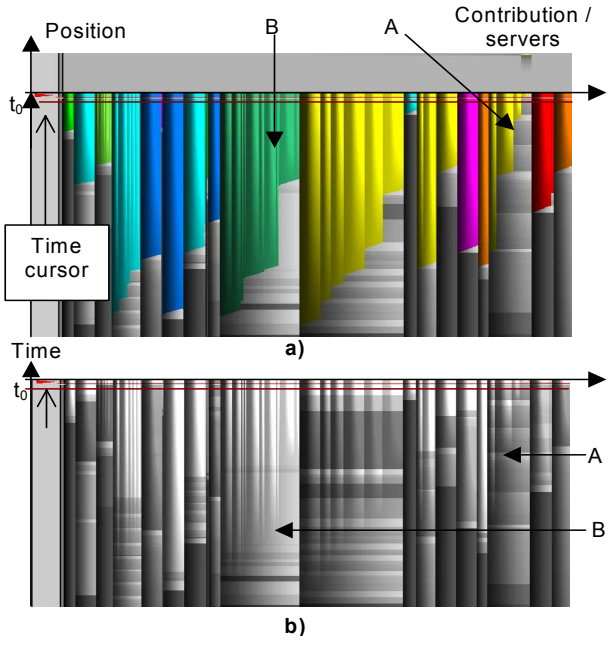

Figure 14: Dispatcher algorithm assessment

Figure 14.a shows that at t_0 the downloaded segment came from server **A**, while Figure 14.b shows that at the same time, the faster source **B** was also available (i.e., we were not in queue but ready to be served). That means the dispatching

algorithm in the *eMule* client is not optimized for the minimization of waiting time.

Finally, Figure 15 illustrates the possibilities that the techniques we described in this paper have for the field of visualization on small displays. The good scaling behavior for the server visualization combined with the efficiency of shading in scatter plots, and the partitioning qualities of cushions create uncluttered images that allow performance assessment of P2P file-sharing networks even on low-resolution displays.

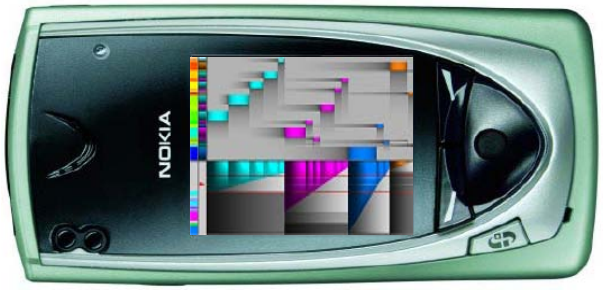

Figure 15: Download visualization of a MP3 song on a Nokia 7650 display using EZEL

8. CONCLUSIONS AND FUTURE WORK

In this paper we presented a new approach for the visual assessment of performance in P2P file-sharing networks, and we validated it using EZEL, a prototype assessment tool for the *ED2K* network.

We started by identifying the data transferred in P2P file-sharing networks, and then we tried to find relevant performance descriptions based on it. Subsequently, we built a custom visualization made of two correlated parts: a server and a download visualization. For each part we visually encoded a number of descriptions and we proposed a number of enhancements and combinations of existing visualization techniques. Notably, we used shaded cushions for virtually all data elements (servers, segments, queue positions, and correlation plot elements). Overall, our visualization gives a compact and scalable way to present a download consisting of thousands of transactions, from over 100 sources, on a single screen. To our knowledge, this is the first attempt to visually explore the data transfer dynamics in the rapidly growing world of P2P file-sharing networks. The other work we are aware of in this field addresses a different task, namely visualizing the topology of a P2P network of a different type [9].

The examples presented in this paper address the visualization of data obtained at the end of a download. However, using the same approach for building a dynamic, "real-time" visualization of the acquired data is in theory possible. The only issue in this case would be the frame rate at which images are produced. In the worst case, more than 200.000 transactions have to be considered for the generation of each frame. This would require an impressive processing power in order to update the image at the arrival of each new transaction (on average every 100 ms). The current implementation of EZEL generates images at 0.5 – 1.0 fps on a Pentium 4 processor running at 2.6 GHz. A scenario in-between would be to generate on demand images with the partial information available during the download. While differences between consecutively generated images could be too large to make meaningful correlations, the individual images may be used to interact with the download process based on intermediate assessments.

In the future, we would like to generalize our visualization and extend it to the larger domain of distributed processing in general. The challenges we foresee there relate to the process visualization of the dispatching and scheduling entities.

ACKNOWLEDGEMENTS

This research was part of the ITEA project Space4U, whose aim is to define a component based software framework for the middleware layer of high volume embedded appliances (http://www.win.tue.nl/space4u).

REFERENCES

[1] K. Matkovic, H. Hauser, R. Sainitzer, M. E. Gröller, *Process Visualization with Levels of Detail*, Proc. Info Vis '02, pp. 67-70

[2] R.A. Becker, S.G. Eick, A.R. Wilks, *Visualizing Network Data*, IEEE TVCG, vol. 1, no. 1 , March 1995, pp.16-28

[3] S.T. Eick, *Aspects of Network Visualization*, IEEE Comp. Graph. & Appl., vol. 16 , no. 2 , March 1996, pp. 69-72

[4] C. Stolte, R. Bosch, P. Hanrahan, M. Rosenblum, *Visualizing Application Behavior on Superscalar Processors*, Proc. Info Vis '99, pp. 10-17,141

[5] E. H. Chi, S. K. Card, *Sensemaking of Evolving Web Sites Using Visualization Spreadsheets*, Proc. Info Vis '99, pp. 18-25,142

[6] J. J. van Wijk, H. van de Wetering, *Cushion Treemaps: Visualization of Hierarchical Information*, Proc. Info Vis, 1999, pp. 73-78

[7] R. Spence, *Information Visualization*, ACM Press, 2001.

[8] R. M. Wilson, R. D. Bergeron, *Dynamic Hierarchy Specification and Visualization*, Proc. Info Vis '99, pp. 65-72

[9] K.-P. Yee, D. Fisher, R. Dhamija, M. Hearst, *Animated Exploration of Dynamic Graphs with Radial Layout*, Proc. Info Vis '01, pp. 43 – 50

[10] eMule : http://www.emule-project.net/

[11] ED2K : http://www.edonkey2000.com/