Implementation of a Portable Peer-to-Peer Client-Server

Mauricio Toro
Universidad EAFIT

Abstract

Several peer-to-peer programs have been developed since the creation of the peer-to-peer protocols and models. However, the most popular ones such as Kazaa and Napster are not free software and are not portable to all operative systems. This paper describes Toro-P2P, an implementation of the P2P technology devolved in Python, a functional object-oriented language, which is very portable because of its interpreter. It includes two main modules which are the server and the client. The first one includes a RPC server and a HTTP server providing functions to add new files to the list of shared files and to query the list. The second one, the client, provides functions to query the server, to download, and upload files with other clients. Finally a GUI was designed using wxPython which is a portable object-oriented library wrapped from C++. One of the most remarkable features included in Toro-P2P is the easy configuration of the program including its ports and sockets.

Keywords: P2P, client, server, Python, wxPython, RPC, HTTP, GNU, peer-to-peer, lableng2.

Resumen

Muchos programas con tecnología P2P han sido desarrollados desde sus orígenes. No obstante, los más populares tales como el Kazaa y el Napster no son de software libre y no son portables a todos los sistemas operativos. Este trabajo describe la el Toro-P2P que es un implementación de la tecnología P2P desarrollado en Python, un lenguaje funcional orientado a objetos, que es muy portable dado que es interpretado. Este incluye dos módulos principales que son el servidor y el cliente. El primero incluye un servidor RPC y un servidor HTTP que proveen funciones para agregar nuevos archivos a una lista de archivos compartidos y para consultar dicha lista. El segundo, el cliente, provee funciones para consultar el servidor, para bajar archivos y para subir archivos de otros clientes. Finalmente se diseñó una interfaz de usuario con wxPython que es una librería portable orientada a objetos portada de C++ hacia Python. Una de las características más notables incluidas en Toro-P2P es la fácil configuración del programa incluyendo sus puertos y sockets.

Palabras claves: P2P, cliente, servidor, Python, wxPython, RPC, HTTP, GNU, peer-to-peer, lableng2.

Introduction

Peer-to-Peer file sharing is not a new phenomenon. The first of the file sharing clients came out in early 1999 and reached mainstream popularity within a few months. Despite media hype to the contrary, Peer-to-Peer usage has grown consistently. The consumption of network resources by Peer-to-Peer has continued to grow since 1999 [50]. The objectives are: First, to put in practice the networking and web application design concepts and to improve our abilities to develop software in an organized, clean and documented way. Another important objective we traced before developing this application is to confirm that the *P2P* technology is not wrong or illegal, we think the real problem with this technology is the uses that user gave to it, not the protocols, not the application, or the design.

Process calculi have been applied to the modeling of interactive music systems [36, 35, 33, 29, 34, 28, 42, 31, 32, 27, 30, 26, 41, 48, 25, 47, 1, 43, 37, 6, 23, 20, 22, 24, 2, 40, 21, 38, 39, 19] and ecological systems [45, 9, 46, 10]. In addition, research on algorithms [7, 5, 15, 11, 13, 16, 14, 44, 3, 8], software engineering [18, 4], precision agriculture [51] and intelligent transport systems [12, 17, 49, 50] also contributes to this field.

The rest of the paper is organized as follows: Section 1 describes the implementation of the Server, explaining each of its modules. Section 2 explains the implementation of the Client, its modules and the files it uses. Section 3 explains the implementation of the GUI and finally we explain some conclusions and future works.

1 Implementation of the Server

Figure 1: Diagram of the Server implementation

The Server implementation includes 4 modules; they are represented clearly in the figure bellow (Figure 1). Default ports are configurable as well as many other attributes.

1.1. The server

The main one which we will call Server (Server.py) includes two threads: The main one is the *HTTP* server, because this one has the possibility to be turned off remotely, allowing to control the execution of the program and freeing OS resources. When one starts the server, it configures itself in case of being necessary, creating a configuration file or reading the configuration file, either way, both threads are started calling the *RPC* module (rpcserver.py) and the *HTTP* module (sobreweb2.py)

1.2. The RCP module

This module (rpcserver.py) was designed using the *SimpleXMLRPCServer* class from Python. It includes functions for searching, adding and deleting files from the server set of shared files, and besides it has functions for get the client IP address and turning on the server.

1.3. The HTTP module

This module (sobreweb2.py) is a configurable *HTTP* server. It accepts request of files. It has some *CGI* parsing used to accept specific *CGI* request in order to search for files in the **set of shared files** of the server. Besides that, a nice appearance was developed for web users.

1.4. The Directory module

This module (directorio.py) is in charge of managing the **set of shared files** (implemented using Python Set class) which is saved into a file due to some concurrency issues. The first one, is that different threads may want to access to the set of shared files, and may modify the information of the current shared files. Because of that, a file is created with the information of the set and it is blocked while is being used by a thread. In **Windows OS** the files are automatically block when they are being read which helps us to preserve the integrity of the information. This module has functions to add, delete and search files.

2 Implementation of the Client

Figure 2: Diagram of the Client implementation

The client implementation is composed by the modules and files, threads and processes are presented with circles and files are presented with cylinders (Figure 2).

2.1 The Client

The **Client** implementation includes 3 modules. The main one called **Client** (cliente.py) includes the **Upload module** (clienteserver.py), this one is an agent it self, processing each request in a different thread. The last module is the **Download module** (clienteciente.py), which is able to download files from another **Upload module**, for convenience, multiple threads containing instances of this module, may be created.

2.2 The Upload module

The **Upload module** contains a class called “clienteserver” which encapsulates the uploading process. The initialization method starts an uploading server with the default port. As soon as it receives a client download request, a new thread is started with the function in charge to send the entire file.

2.3 The Download module

This module allows multi-thread downloading from different clients. Each time we want to download a file, a “clienteciente” class is created in a different thread by using the download function. Besides that, there are two functions to supervise the downloading process, one to check the actual download percentage and the other one to check the received size in bytes. Some functions regarding timeout processing were included in the implementation allowing us to report when a timeout has occurred or when the connection from the **Upload module** of another client has been refused.

3 Implementation of the GUI

The *GUI* was implemented in one module called *interface* (*interfaz.py*). In order to achieve this goal, we used **wxPython** which a portable *GUI* library wrapped from *C++* into *Python*, allowing us to develop a simple *GUI* portable to *Windows*, *Linux* and *Mac*. The application includes one frame, containing a menu bar, where we have *connect*, *disconnect*, *exit* and *configuration* functions among others and we have a “notebook” which is a class that allow us to have three sheets: *Help*, *Search* and *Download*. The **Download Sheet** uses a list control, allowing us to show the tuples that contains information about the files that we are searching or downloading. One difficulty with this interface is that it is not possible to search more files until the files being download are done. This is not possible because the answer of the query and the download progress are shown in the same place, because of that a new query will update the list control, deleting the download progress information.

4 Concluding remarks and future work

The implementation of the P2P client-server technology was important to put in practice the concepts studied in class. Besides that we learned that using threads may be harder than imperative programming but it is closer to reality and it can be very useful in the implementation of servers and web application. In the other hand using Low-level networking interface (sockets) will help us to understand complex web applications in a future, and it was a good practice preparing us to other courses.

Finally in a long term we want to allow the user to search files while he’s downloading, provide a better way to get the Client address, download a file from different sources and resume a download, make a user friendly interface and finally, realizing the inefficiencies of the distributed/decentralized architecture, we will try to combined the benefits of a centralized topology with the stealth characteristics of distributed /decentralized topology.

References

- [1] ALLOMBERT, A., DESAINTE-CATHERINE, M., AND TORO, M. Modeling temporal constraints for a system of interactive score. In *Constraint Programming in Music*, G. Assayag and C. Truchet, Eds. Wiley, 2011, ch. 1, pp. 1–23.
- [2] ARANDA, J., ASSAYAG, G., OLARTE, C., PÉREZ, J. A., RUEDA, C., TORO, M., AND VALENCIA, F. D. An overview of FORCES: an INRIA project on declarative formalisms for emergent systems. In *Logic Programming, 25th International Conference, ICLP 2009, Pasadena, CA, USA, July 14-17, 2009. Proceedings* (2009), P. M. Hill and D. S. Warren, Eds., vol. 5649 of *Lecture Notes in Computer Science*, Springer, pp. 509–513.
- [3] ARCILA, V., PIEDRAHITA, I., AND TORO, M. Robotic bees: Algorithms for collision detection and prevention. *arXiv e-prints* (May 2019), arXiv:1905.11822.
- [4] MAZO, R., TORO, M., AND COBALEDA, L. Definicion de la arquitectura de referencia de un dominio: de la elucidacion al modelado. In *Guia para la adopcion industrial de lineas de productos de software*, R. Mazo, Ed. Editorial Eafit, 2018, pp. 193–210.
- [5] MORENO, J. D. A., PASSOS, S., AND TORO, M. On-line assembling mitochondrial DNA from de novo transcriptome. *CoRR abs/1706.02828* (2017).
- [6] OLARTE, C., RUEDA, C., SARRIA, G., TORO, M., AND VALENCIA, F. Concurrent Constraints Models of Music Interaction. In *Constraint Programming in Music*, G. Assayag and C. Truchet, Eds. Wiley, Hoboken, NJ, USA., 2011, ch. 6, pp. 133–153.
- [7] PATIÑO-FORERO, C., AGUDELO-TORO, M., AND TORO, M. Planning system for deliveries in Medellín. *ArXiv e-prints* (Nov. 2016).
- [8] PEREZ BERNAL, G., TORO VILLEGAS, L., AND TORO, M. Saber pro success prediction model using decision tree based learning. *arXiv e-prints* (2020).
- [9] PHILIPPOU, A., AND TORO, M. Process Ordering in a Process Calculus for Spatially-Explicit Ecological Models. In *Proceedings of MOKMASD’13* (2013), LNCS 8368, Springer, pp. 345–361.
- [10] PHILIPPOU, A., TORO, M., AND ANTONAKI, M. Simulation and Verification for a Process Calculus for Spatially-Explicit Ecological Models. *Scientific Annals of Computer Science* 23, 1 (2013), 119–167.
- [11] QUINTERO, E., SANCHEZ, M., ROLDAN, N., AND TORO, M. Genetic algorithm for optimal distribution in cities. *ArXiv e-prints* (Nov. 2018).

- [12] RAMÍREZ, T., CASTAÑEDA, L., AND TORO, M. Sistema inteligente para transporte público tipo SETP. In *XIII Congreso Colombiano de Transporte y Tránsito, Movilizando el futuro de Colombia* (2019), Universidad Tecnológica de Bolívar, p. 57.
- [13] RENDON, C., HERNÁNDEZ, J. L., RUIZ-SALGUEDO, O., ÁLVAREZ, C. A., AND TORO, M. Wing Profile Evolution driven by Computational Fluid Dynamics. In *Tecnologías Avanzadas de Mecatrónica, Diseño y Manufactura: Memorias Cuarto Congreso AMDM* (2018), Editorial Universidad Autónoma de Manizales, pp. 345–347.
- [14] RENDON, C., HERNÁNDEZ, J. L., RUIZ-SALGUEDO, O., ÁLVAREZ, C. A., AND TORO, M. Wing Profile Evolution driven by Computational Fluid Dynamics. *Revista UIS Ingenierías* 19, 2 (2019), 139–146.
- [15] RESTREPO, J. M. C., PALACIO, A. F. Z., AND TORO, M. Assembling sequences of DNA using an on-line algorithm based on debruijn graphs. *CoRR abs/1705.05105* (2017).
- [16] RODRIGUEZ, J., ESCUDERO, D., AND TORO, M. Data Structure for efficient indexing of files and directories. *Research Gate* (2018).
- [17] SAGBINI, K., RAMÍREZ-GUERRERO, T., CASTAÑEDA, L., AND TORO, M. Aplicaciones móviles para la evaluación de conductores y usuarios en Sistemas Estratégicos de Transporte Público. *Desafíos en Ingeniería: Investigación Aplicada* (2019).
- [18] SUESCÚN, E., TORO, M., MAZO, R., VELASQUEZ, D., VALLEJO, P., CARDONA, J. F., RINCÓN, R., WERNECK, V. M., AND DO PRADO LEITE, J. C. C. SimulES-W: A Collaborative Game to Improve Software Engineering Teaching. *Computación y Sistemas* 22, 3 (2018), 953–983.
- [19] TORO, M. Exploring the possibilities and limitations of concurrent programming for multimedia interaction and graphical representations to solve musical csp’s. Tech. Rep. 2008-3, Ircam, Paris.(FRANCE), 2008.
- [20] TORO, M. Probabilistic Extension to the Factor Oracle Model for Music Improvisation. Master’s thesis, Pontificia Universidad Javeriana Cali, Colombia, 2009.
- [21] TORO, M. Towards a correct and efficient implementation of simulation and verification tools for probabilistic ntcc. Tech. rep., Pontificia Universidad Javeriana, May 2009.
- [22] TORO, M. Structured interactive musical scores. In *Technical Communications of the 26th International Conference on Logic Programming, ICLP 2010, July 16-19, 2010, Edinburgh, Scotland, UK* (2010), M. V. Hermenegildo and T. Schaub, Eds., vol. 7 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 300–302.
- [23] TORO, M. *Structured Interactive Scores: From a simple structural description of a multimedia scenario to a real-time capable implementation with formal semantics*. PhD thesis, Univeristé de Bordeaux 1, France, 2012.
- [24] TORO, M. Structured interactive music scores. *CoRR abs/1508.05559* (2015).
TORO, M. Probabilistic Extension to the Concurrent Constraint Factor Oracle Model for Music Improvisation. *ArXiv e-prints* (Feb. 2016).
- [25] TORO, M. Probabilistic Extension to the Concurrent Constraint Factor Oracle Model for Music Improvisation. *Inteligencia Artificial* 57, 19 (2016), 37–73.
- [26] TORO, M. Backtracking-free interactive music scores with temporal relations over rythms. *OSF Preprints 10.31219/osf.io/bwzsk* (Oct 2018).
- [27] TORO, M. Combining temporal relations and signal processing into interactive scores. *OSF Preprints osf.io/sqmc* (Oct 2018).
- [28] TORO, M. Current Trends and Future Research Directions for Interactive Music. *ArXiv e-prints* (Oct. 2018).
- [29] TORO, M. CURRENT TRENDS AND FUTURE RESEARCH DIRECTIONS FOR INTERACTIVE MUSIC. *Journal of Theoretical and Applied Information Technology* 69, 16 (2018), 5569–5606.
- [30] TORO, M. Modeling michael jarrell’s csp in gelisp. *OSF Preprints osf.io/2ebzj* (Oct 2018).
TORO, M. ntccmc: A bounded-time model checker for ntcc. *OSF Preprints osf.io/c8kah* (Oct 2018).
- [31] TORO, M. Towards a correct and efficient implementation of simulation and verification tools for probabilistic ntcc. *ArXiv e-prints* (Oct. 2018).

- [32] TORO, M. Towards a correct and efficient implementation of simulation and verification tools for probabilistic ntc. *OSF Preprints* 10.31219/osf.io/3wp5z (Oct 2018).
- [33] TORO, M. Gelisp 1.0: A graphical interface for openmusic. *OSF Preprints* osf.io/w4yfe (Jan 2019).
- [34] TORO, M. How to implement lightweight threads. *OSF Preprints* osf.io/84tvj (Jan 2019).
- [35] TORO, M., AGÓN, C., ASSAYAG, G., AND RUEDA, C. Ntcrt: A concurrent constraint framework for real-time interaction. In *Proc. of ICMC '09* (Montreal, Canada, 2009).
- [36] TORO, M., AND DESAINTE-CATHERINE, M. Concurrent constraint conditional branching interactive scores. In *Proc. of SMC '10* (Barcelona, Spain, 2010).
- [37] TORO, M., DESAINTE-CATHERINE, M., AND BALTAZAR, P. A model for interactive scores with temporal constraints and conditional branching. In *Proc. of Journées d'Informatique Musicale (JIM) '10* (May 2010).
- [38] TORO, M., DESAINTE-CATHERINE, M., AND CASTET, J. An extension of interactive scores for multimedia scenarios with temporal relations for micro and macro controls. In *Proc. of Sound and Music Computing (SMC) '12* (Copenhagen, Denmark, July 2012).
- [39] TORO, M., DESAINTE-CATHERINE, M., AND CASTET, J. An extension of interactive scores for multimedia scenarios with temporal relations for micro and macro controls. *European Journal of Scientific Research* 137, 4 (2016), 396–409.
- [40] TORO, M., DESAINTE-CATHERINE, M., JANIN, D., AND ORLAREY, Y. Real-time interactive streams and temporal objects language. *OSF Preprints* 10.31219/osf.io/a6stk (Oct 2018).
- [41] TORO, M., DESAINTE-CATHERINE, M., AND RUEDA, C. Formal semantics for interactive music scores: a framework to design, specify properties and execute interactive scenarios. *Journal of Mathematics and Music* 8, 1 (2014), 93–112.
- [42] TORO, M., AND LANIADO, H. Big data: historia, definición, herramientas y aplicaciones en la industria. *Revista Virtual Pro* 204 (2019), 1–5.
- [43] TORO, M., PHILIPPOU, A., ARBOLEDA, S., PUERTA, M., AND VÉLEZ S., C. M. Mean-field semantics for a process calculus for spatially-explicit ecological models. In *Proceedings of the Eleventh International Workshop on Developments in Computational Models*, Cali, Colombia, October 28, 2015 (2016), C. A. Muñoz and J. A. Pérez, Eds., vol. 204 of *Electronic Proceedings in Theoretical Computer Science*, Open Publishing Association, pp. 79–94.
- [44] TORO, M., PHILIPPOU, A., KASSARA, C., AND SFENTHOURAKIS, S. Synchronous parallel composition in a process calculus for ecological models. In *Proceedings of the 11th International Colloquium on Theoretical Aspects of Computing - ICTAC 2014, Bucharest, Romania, September 17-19 (2014)*, G. Ciobanu and D. Méry, Eds., vol. 8687 of *Lecture Notes in Computer Science*, Springer, pp. 424–441.
- [45] TORO, M., RUEDA, C., AGÓN, C., AND ASSAYAG, G. Ntcrt: A concurrent constraint framework for soft real-time music interaction. *Journal of Theoretical & Applied Information Technology* 82, 1 (2015).
- [46] TORO, M., RUEDA, C., AGÓN, C., AND ASSAYAG, G. Gelisp: A framework to represent musical constraint satisfaction problems and search strategies. *Journal of Theoretical & Applied Information Technology* 86, 2 (2016).
- [47] VALENCIA, J. F., RAMÍREZ-GUERRERO, T., CASTAÑEDA, L. F., AND TORO, M. Automatic detection of number-plate and traffic infractions of motorcyclists by intelligent transportation systems. *IOP Conference Series: Materials Science and Engineering* 844 (jul 2020), 012035.
- [48] VALENCIA, J. F., RAMÍREZ-GUERRERO, T., CASTAÑEDA, L. F., AND TORO, M. Detección de infracciones y matrículas en motocicletas, mediante visión artificial, aplicado a sistemas inteligentes de transporte. *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao* 37 (2020).
- [49] VELÁSQUEZ, D., SÁNCHEZ, A., SARMIENTO, S., TORO, M., MAIZA, M., AND SIERRA, B. A method for detecting coffee leaf rust through wireless sensor networks, remote sensing, and deep learning: Case study of the caturra variety in colombia. *Applied Sciences* 10, 2 (2020).
- [50] Cache Logic Website. <http://www.cachelogic.com/p2p/>