DEVS Simulation of Peer-To-Peer File-Sharing

Alan Davoust, Gabriel Wainer, Babak Esfandiari
Dept. of Systems and Computer Engineering
Carleton University Ottawa, Canada
Email: {adavoust,gwainer,babak}@sce.carleton.ca

***Abstract*—We present a framework to simulate a peer-to-peer (P2P) file-sharing network, based on the Discrete Event Systems Specification (DEVS) formalism. Our framework models a file-sharing network as a coupled model, comprising a network model and a large number of peer models. While most available network simulation tools focus on transport-level dynamics, we provide extensible and reusable models for the file-sharing protocol and for the behavior of peers. These models, implemented using the CD++ toolkit, can readily be used on existing simulators, including parallel and real-time simulators.**

As a case study, we apply our framework to simulate a P2P web, and show the emergence of an interesting page distribution.

***Keywords*-Peer-To-Peer; Simulation; DEVS**

I. Introduction

In a peer-to-peer (P2P) network, the system of interest is an *overlay* network, i.e. a network of peers connected in an arbitrary topology, different from the physical network topology. Simulating a P2P network therefore requires a (real or simulated) transport layer infrastructure, and, at the application level, users (or simulated users) to provide input. As larger-scale networks are needed to evaluate such aspects as the efficiency of a search protocol or the impact of an incentive scheme, the required network and manpower become unavailable, and must be simulated.

The simulation of transport networks is a well researched topic, with many tools available, such as the popular Omnet++ [21] or NS2 [17]. On the other hand, to simulate the peers (users), reusable models are scarce.

Many P2P-related studies include validation by simulation, but the peer models used in those simulations are rarely reported in detail, if at all. This, along with the variety of existing simulators, makes it very difficult to compare or reproduce any reported results.

In our research, we develop new applications on top of a P2P file-sharing platform. We then use simulation to study the evolution of the network content, as an emergent side-effect of the peers' behavior. For this purpose, we need elaborate peer models, possibly connected with models of other domains, which would be difficult to adapt to P2P-specific simulators.

In this paper, we present a framework for simulating a P2P file-sharing network, based on the Discrete Event Systems Specification (DEVS) formalism [26]. The use of DEVS theory provides with a method with sound semantics to represent both discrete systems such as the network infrastructure, and continuous systems such as the peers themselves.

Our framework comprises a network model, and a generic peer model with configurable and customizable stochastic behavior. It is implemented using the CD++ toolkit [22], a general purpose DEVS simulation toolkit. As such, our models are highly reusable and can be integrated with models of other domains, for example to model external events which may trigger increased peer activity, or affect network connectivity.

The rest of this paper is organized as follows. After a brief survey of related work in Section II, we present in section III a formal characterization of a P2P network, which we use as a basis for our DEVS models. Then in Section IV we describe our DEVS framework model, and in Section V we detail the principles of our peer model. In Section VI we illustrate the application of our framework to an interesting scenario: we consider a file-sharing network containing hyperlinked files[1], and simulate the behavior of users "surfing" that P2P web. Finally, in section VII we briefly present a separate visualization tool that can show a "behind the scenes" view of the network evolution, and provide visual clues of phenomena that may not appear directly in statistical studies. In Section VIII we draw a few conclusions and present some directions for future work.

II. Related Work

P2P research (and simulation) covers a wide spectrum of applications, such as file-sharing, grid computing, distributed storage, collaborative work, etc. However, we focus here on file-sharing, the main application of our research (and by far the most popular application in the literature).

The main publicly available, discrete-event P2P simulation tools are PeerSim [16] and OverSim [2]. Other available tools include P2PSim [11] and GPS [24].

PeerSim [16] is an open-source java framework for simulating P2P networks, widely used for evaluating P2P

[1]we note that this scenario is motivated by a real application described in [9]

978-1-4673-2362-8/12/$31.00 ©2012 IEEE

protocols. It includes a cycle-based simulation engine that abstracts away the transport layer in return for high scalability, and a discrete-event simulation engine that includes a model for the underlying network. It uses the King dataset to estimate network latency, a dataset of known latency measurement between a number of real hosts on the internet. Users may implement their own overlay network protocol (or reuse existing ones), and they must also define and implement peer behavior models. An extension for distributed simulation (for scalability) was proposed (dPeersim [10]).

OverSim [2] is also an overlay simulation tool, based on the discrete-event simulation tool Omnet++, written in C++. OverSim includes several models for the transport layer, and tools to analyse simulation results. As with PeerSim, no peer behavior models are provided.

Such P2P simulation systems offer sizeable libraries of P2P protocols, and in some cases provide quality tools for packet-level simulation. But the design of peer models is left entirely to the users. As a consequence, many simulation studies use basic – or unspecified – assumptions about the peer behavior, and focus the bulk of their efforts on simulating the protocol. The main problem is that simulation results from different studies cannot be easily compared.

Compared to existing tools, a DEVS-based approach allows us to easily integrate models of arbitrary domains. This includes advanced network-level models: work has been done to integrate a DEVS simulation engine with the popular network simulator NS-2 [13], and a recent project has also appeared to build a pure DEVS network model library [27].

Finally, we note that by using the generic toolkit CD++, our models can be simulated on several readily available simulators, including real-time and distributed simulators.

III. Formal Characterization of a Peer-to-peer network

In previous work [8] we have proposed a formal characterization of a Peer-to-peer network. In this section, we summarize this formal model and show how it can be used to define a framework DEVS model.

A. Scope of the model

For the purpose of our research, a file-sharing network is an infrastructure, at the same level of abstraction as the client-server web. We are primarily interested in analyzing the effect of various *search mechanisms* on the distribution of files in the network.

According to Daswani et al. [7], a P2P search mechanism defines the behavior of peers in three areas:

- Topology: how peers connect to one another;
- Data placement: what data is stored by which peers;
- Query routing: how the peers' queries are propagated through the network.

Our model aims to accomodate arbitrary strategies in terms of network topology and query routing, but restricts the data placement possibilities: we assume *autonomous* peers, in the sense that they have full and exclusive control over their local repositories.

Alternative data placement strategies generally assume that the location and replication of files in the network (in other words, their distribution) is controlled by the protocol. These approaches mostly fall under the category of Distributed Hash Tables. In such networks, the peers' searching and downloading behavior generates traffic but will leave the file distribution in the DHT itself unchanged, as it is dictated by the DHT protocol.

As we are precisely interested in the evolution of the file distribution, our model is designed to accomodate fully autonomous peers, as defined above. Finally, we note that this model does not model time (it considers a succession of states), and abstracts away the transport layer entirely, i.e. the details of the physical interconnection of peers. Aspects relevant to the physical network must be modeled separately, and integrated with the overlay network model, as described in section IV-A.

B. Peer-to-peer Network

An overlay network formed by a set of peers $\{P_i\}$ can be modeled by a Labelled Transistion System (LTS)[2].

Each state of this LTS is a graph, where the nodes are[3] a subset of $\{P_i\}$, and the edges represent the connections between the peers. The initial state of the LTS is an empty graph.

The possible transitions from a given state are the addition or removal of a node or edge in the graph, i.e. the following events:

- a (previously offline) peer goes online;
- a (previously online) peer goes offline;
- an online peer connects to another online peer;
- an online peer drops its connection to another online peer.

The conditions indicated for a given event (e.g. a peer can only go offline if it was previously online) indicate which state this event may occur in. The resulting state after each event is straightforward: if a peer goes online, the next state is the same graph with a new vertex associated with the peer, and so on.

C. File-Sharing Community

We now define the concept of a file-sharing *community*. The file-sharing community represents the functionality of a

[2]a labelled transition system is a special case of a deterministic automaton, where there is no notion of "accepting state" (i.e. all states are accepting)

[3]more formally, we should distinguish the nodes in each graph from the peers themselves. A node only exists within the context of a graph, which is itself a state of an LTS. The peers in a network do not cease to exist when they go offline. In the following discussion we will ignore this distinction.

peer-to-peer client, as offered to the human user, and reliant on the network interconnection.

A file-sharing community is defined by:

- a set of peers
- a network formed by this set of peers (i.e. in any state, a subset of these peers forms the graph)
- a query protocol
- a (possibly infinite) set of data items D
- a (possibly infinite) query language Q

In this community, each peer stores some data and is capable of outputting queries.

In terms of data and queries, we can consider the data items $\{d \in D\}$ and the queries $\{q \in Q\}$ to be two disjoint sets of abstract entities, and we have a binary relation *match(.,.)* on $Q \times D$.

The community protocol is a function that takes a graph and a particular node of this graph as input, and returns a subset of nodes of this graph. This abstract function can be applied to a particular state of a network: if a peer (connected in the network) outputs a query, the protocol specifies which peers receive the query.

A peer has access to the following data operations:

- publish (d: data item): add d to the peer's local repository;
- remove (d: data item): if d is in the peer's local repository, then remove d;
- query (q: query): output the query q to the network. The peer receives as a response the list of data items matching the query, and stored by the peers that were reached by the query (this being determined by the protocol);
- download (d: data item): this operation assumes as a precondition that the peer does not store d locally, and that d was a response to a previous query by the peer. The post-condition of the operation is that the peer publishes a copy of d to its local repository.

The community concept allows for a user to take part in several different communities, with a different behavior, and different types of data. In our research, we have created for example a community to share science papers and citations, and a P2P wiki system [5].

IV. DEVS Model Framework

We now present the realization of this general model using the DEVS formalism. At this level of abstraction, our formal model can be represented as a framework for a coupled DEVS model.

Our framework model has the structure illustrated in Figure 1. In the following sections we detail the components of the model, and for each component, we specify the behavior of our default implementation.

A. Network Model

1) Network Graph: The NetworkGraph model keeps track of the overlay network topology. It is a state machine (an LTS, as discussed in section III-B), and its inputs are messages from the peers that go online, offline, or attempt to connect to other peers. The component only changes the graph state following legal transitions, and outputs messages to notify peers of successfully established connections, and of dropped connections.

Our default implementation manages this functionality, and introduces only minimal latency: successful connections are almost instantaneous.

2) Physical Network: This model transfers messages from one peer to another. It represents the network connectivity: the component accepts input messages from any peer, and outputs the same message to a specified destination peer.

Our implementation of the model delivers the messages reliably, with a random latency, following a configurable statistical distribution (independent of network congestion). For more fine-grained performance studies, this model would need to be replaced by a more complex model of the network, e.g. a packet-level simulator such as NS-2.

B. Peer Model

1) Connection Manager: This component represents the behavior of a peer regarding connections, i.e. the "topology" aspect of a search mechanism, as discussed in section III-A. The peer communicates with the NetworkGraph component to establish connections to other peers, and maintains a list of its neighbours.

In our implementation of this component, the component is initially configured with a list of acquaintances. Its behavior is then to periodically attempt to connect to each of its acquaintances. When the connections are established, the component remains idle. This behavior creates an unstructured network based on a social network.

2) Router: The router component represents the query routing behavior of a peer. The component maintains the current list of neighbours of the peer (based on inputs from the NetworkGraph component), and receives input queries, both from the local peer (from the SessionManager component) and from other peers, via the network. The peer then routes the queries to its neighbours, implementing a specific search protocol.

Our implementation runs the Gnutella protocol v.0.4 [1].

3) Repository: The repository component maintains the list of documents stored locally by the peer. Its inputs are the *publish* and *remove* operations discussed in section III-C. This model is also used to answer queries; it therefore implements the *match* relation described in section III-C. For this purpose, it takes queries as input, and has an output for query answers.

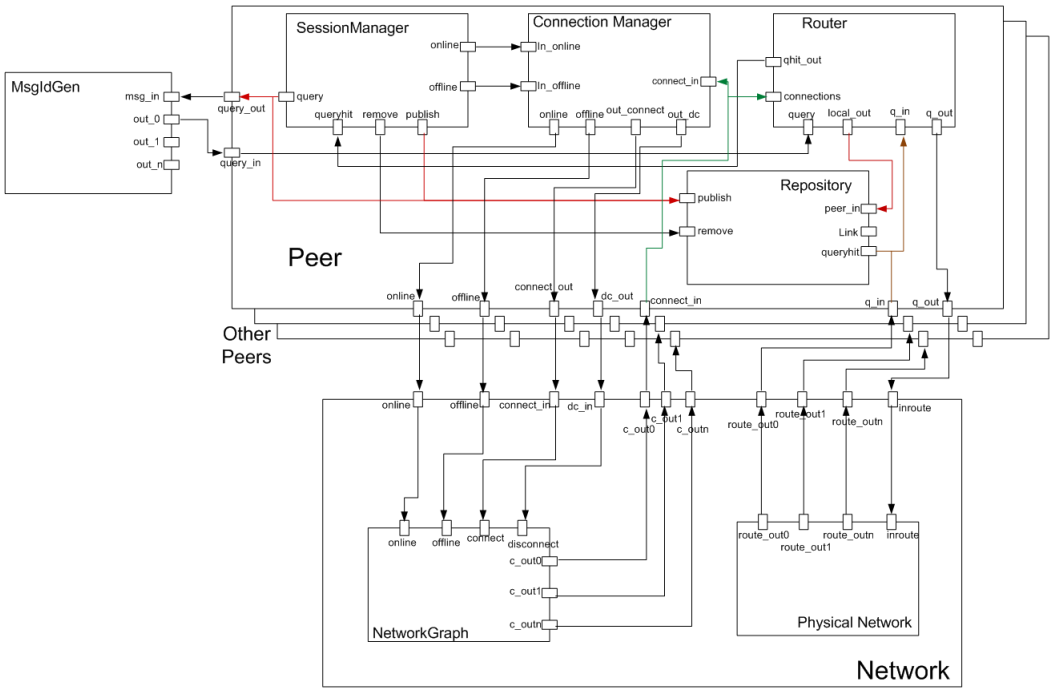

Figure 1. DEVS Model Framework, showing the structure and interconnections of the network and peer models.

Our default implementation of this model is based on a graph loaded on initialization of the model, which describes the *match* relation. It is a bipartite graph where query nodes are related to document nodes by "match" edges. It then separately maintains the list of documents stored by the peers, based on the "publish" and "remove" inputs. The model responds to queries with a small, fixed latency.

4) Session Manager: The Session Manager represents the behavior of the human user of the P2P application.

The outputs of this model are the events initiated by a human user: the peer begins a session and goes online, creates queries, or manages its local repository, by publishing or removing documents.

We discuss our implementation of this model in more detail in section V.

5) MsgIdGen: This model is a simple convenience model that is used by all the peers to generate globally unique identifiers. Many P2P protocols, including Gnutella, make use of unique identifiers, which are typically large integers generated by random functions or hash functions. Using an external model to allocate sequential but unique IDs avoids the need for large values and the risk of collisions.

V. A Behavioral Model for Peers

The behavior of the human users is an important aspect of simulating P2P networks, and for our specific research it is fundamental. Although a number of measurement studies exist, to the best of our knowledge, no model has emerged as a reference benchmark for simulating P2P networks. In this section we briefly review the measurement studies of P2P user behavior, then describe our generic model, which is the basis of the Session Manager DEVS model described above.

A. Measurement studies

The main aspects that have been measured in P2P file-sharing networks are on one hand *churn*, which is the phenomenon of peers joining and leaving the network over time, and on the other hand the querying and downloading behavior of peers during their active sessions.

Churn, characterized by session duration, peer inter-arrival time, and downtime, has been analyzed in several networks, including the Gnutella, Overnet, Kad and BitTorrent networks [3], [19], [14].

Stutzbach et al. [19] show that the overall distribution of sessions lengths can be fit to Weibull distributions, and Bhagwan et al. [3] analyze the combined influences of daily activity patterns, with overall long-term peer turnover.

Klemm et al. [14] provide models for the number of queries and their inter-arrival time within peer sessions, and show significant differences between geographical regions (North America, Europe, East Asia), both in query frequencies and session durations.

The popularity of queries and content was analyzed in different networks by Gummadi et al. [12] in 2002, then by Klemm et al. [14], [15] in 2004, then more recently by Dan and Carlsson [6] (2010).

There are several major challenges in adapting these statistical models for the purpose of simulation. For one thing, the available studies do not all agree on their conclusions.

Secondly, for applications that are not about sharing music and video, there is little or no data available. We can assume certain similarities, but the exact distribution parameters are unlikely to generalize across domains. This problem was faced by Siberski et al. for the simulation of their system Edutella [18], and we are exploring entirely new applications, e.g. a P2P wiki [5].

For these reasons, we conclude that it is more important to offer a model that is generic and easy to configure, rather than one that closely matches all of the statistical distributions reported in the literature. Therefore, we have selected the behavior that appears most general and unrelated to the application context, and made the statistical distributions configurable.

B. A Stateful User Model

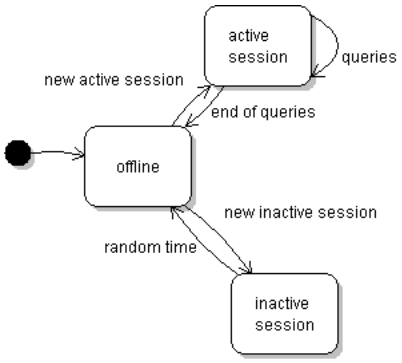

Figure 2. A peer's stateful behavior

From the studies referenced above, we found that peers had a general stateful behavior, which follows a somewhat intuitive pattern. This pattern, illustrated in Figure 2, can be described as follows:

1) the peer starts offline, and after some random time, goes online, and chooses between an active and a passive session;
2)
   - in an active session, the peer outputs a number of queries, and upon receiving responses to the queries, may choose some files to download. The downloaded files are published.
   - in an inactive session, the peers outputs no queries, but is online, and routes and responds to queries from others.
3) after a random "sleeping time", the peer restarts the cycle.

The randomized aspects of this behavior can be configured for each peer:

- offline time;
- probability of selecting an active session;
- in an active session: number of queries, inter-query time, after-querying time, probability of downloading files from queryhits;
- duration of an inactive session.

C. Reusing the Model

The modular structure of the DEVS models makes it easy to adapt and reuse this peer model for a particular problem.

The CD++ toolkit includes most of the classic probability distributions, including all of those reported in the analytical studies referenced above. Modifying a probability distribution does not require modifying the model source code, but only a configuration file, where the name of the distribution and its parameter values must be set.

In addition, each peer in the network can be configured separately: this allows the designer of the network to take *classes* of peers into consideration. For example, one could create peers from different geographical regions, and assign them the different distributions reported in [14]. For large-scale networks, this configuration can be done easily with scripts.

Finally, we note that an appropriate strategy to use this model (as we did for our case study in section VI) would be to configure the time-related session parameters, then to manually code more advanced logic for choosing queries and downloading files. In our case study, this allowed us to model a user "surfing" a hyperlinked collection of files.

VI. Case study: the Random Surfer on a P2P web

A. A P2P web

In this case study, we consider the scenario of a P2P Web, i.e. a collection of hyperlinked documents shared in a Gnutella-like file-sharing network. Documents are not always downloaded from their original publisher, they may be available from other peers. Links refer to specific documents, which could be anywhere in the network. When a peer wants to follow a link, it must send out a search message in order to locate a copy of the linked document, which it can then download.

In this context, our goal is to study the propagation and distribution of files, based on the peers' behavior.

B. The Random Surfer

For this we must simulate the behavior of web browsing. We consider a very simple model, the "random surfer model" used as an intuitive justification of the PageRank algorithm, by Brin and Page [4]. The model is described as follows [4]:

> We assume there is a "random surfer" who is given a web page at random and keeps clicking on links, never hitting "back" but eventually gets bored and starts on another random page.

The idea of the PageRank algorithm is to quantify the probability of that a random surfer will land on a particular page, after an infinite time of browsing. Assuming that the

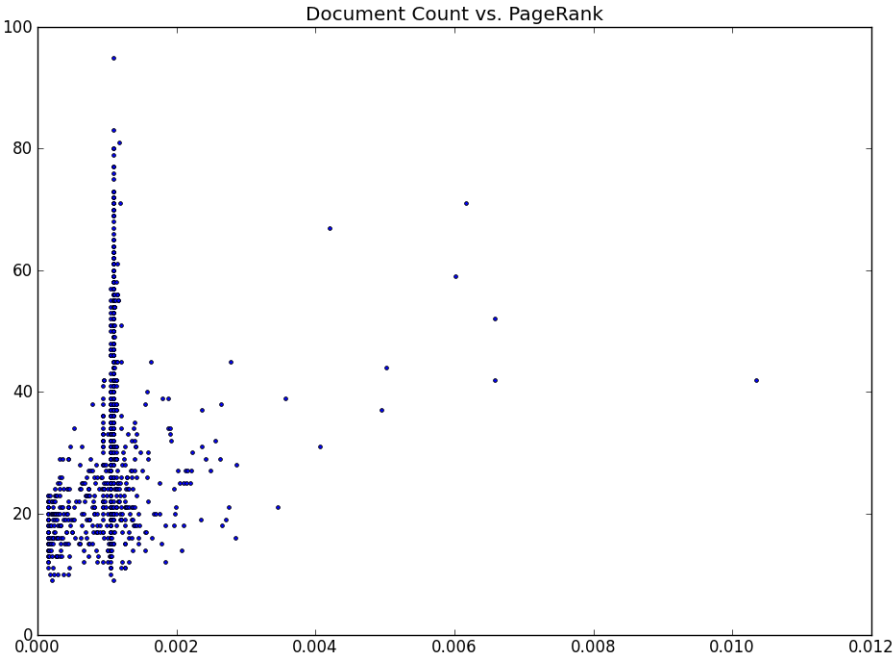

Figure 3. Case study: Document count vs. PageRank values.

links pointing to a page convey an *endorsement* of this page, then the pages with the highest PageRank, those that random surfers are most most likely to land on, are those with the highest reputation (in the sense of this link interpretation).

C. The P2P Random Surfer

In both the traditional and the P2P Web contexts, the users download each page they visit; but in the P2P context, this also means making an additional copy of the page available for download by others. Therefore, in a P2P Web, where the peers randomly surf the collection of pages, the number of copies of each page should be proportional to the page's PageRank value.

However, documents in a Gnutella-like P2P network are not permanently accessible to all the peers. Rather, the documents stored by a peer are accessible only when the peer is online, and only from other peers reachable by the search protocol. The random surfer model must therefore be adapted. Our P2P random surfer will keep clicking on links, and from time to time will get bored, but will also only follow links for pages that are available in the network. If no pages are available, then the surfer searches for another random page.

Research Question: Assuming that the peers implement the P2P random surfer model, is the number of copies of the pages still proportional, or at least positively correlated with the PageRank values?

D. Simulation

We ran simulations with around 50 peers, surfing a collection of 1000 hyperlinked pages. The hyperlink graph was based on a real set of Wikipedia pages. At the beginning of each simulation, 1 to 3 copies of each page were randomly distributed among the peers.

The peers were configured with a number of acquaintances, forming a social graph with a "small world topology"[4]. While they were online, the peers attempted to connect to their acquaintances.

The peer's behavior follows the stateful model described in section V; but during the active sessions, the peers randomly surfed the network, as described above.

In order to mitigate the effects of the initial random document distribution, we ran 6 separate simulations, with different social networks and initial document distributions.

E. Results

The document counts, plotted against their PageRank values, are shown in Figure 3. The correlation between the document counts and the PageRank is quite high: we found a Spearman rank correlation value of 0.43, significant with very high confidence[5].

[4]This graph was generated with an algorithm from [20]

[5]Using randomization tests we obtain a confidence value above 99.99%

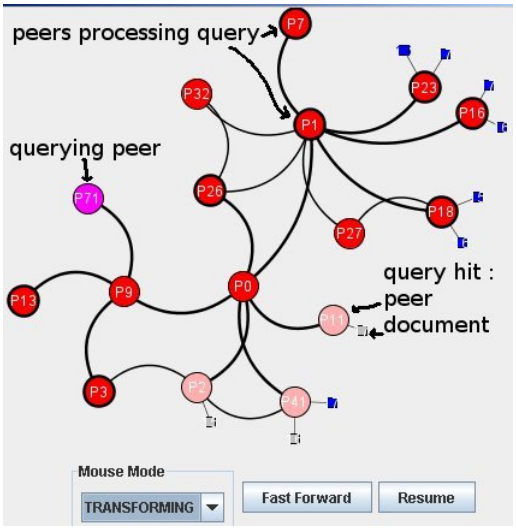

Figure 4. View of a query

The plot itself shows a general "cloud" of data points shaped diagonally in a way that suggests the correlation, and the vertical line at a PageRank value around 0.001 is due to a very large number of pages (several hundred) with identical or close values. Due to the random deviation from a real linear relationship, these documents have different counts, and their sheer number in that narrow region creates a vertical line.

This correlation has interesting consequences. It implies that in a P2P web, the number of copies of a page is likely to reflect its PageRank, which itself has proved to be a valuable quality indicator. The number of copies can thus be used as a quality metric, as we do in our P2P wiki [5].

VII. Visualization Tool

In order to conduct more "exploratory" analysis, we have also developed a visualization tool, for the purpose of observing peer activity in the network. Simulation logs can be fed to this application, which can then re-play the simulation as an animated graph, with the possibility of pausing, reversing, fast-forwarding the animation, and recording videos with third party-tools. The graph of active peers is displayed, with the documents stored by each peer. Nodes appear as the corresponding peers go online, and change colors and shapes as queries are sent and answered.

The propagation of a query is shown in Figure 4. As a query is output by a peer, the corresponding node changes color briefly. Then, as the query is propagated through the network, its path appears as wider edges between the nodes. As peers process queries, their outline becomes wider, and if they respond with a queryhit, then the peer node changes color, as well as the smaller node representing the matching document.

We have found this tool to be of great didactic value when presenting our research to non-specialists.

VIII. Conclusion

We have presented a discrete-event P2P simulation framework that includes configurable and customizable peer models. Implemented using CD++, a general purpose DEVS simulation toolkit, our framework can be easily extended with models of arbitrary domains.

One could for example model the periodic release of new films and music, as a factor influencing queries and downloads, or even consider integrating the network model with a model for a large-scale environmental disaster, which would cause a black-out in parts of the network.

In addition, CD++ can be readily simulated on existing distributed and real-time simulators. Distributed simulators dramatically improve the scalability of simulations [23], while real-time simulators can be used to integrate simulated systems with real deployments [25]. In our case, connecting a real deployment to a simulated network would allow a lab deployment to appear considerably larger than it actually is, and allow for more valuable user studies. This is a possibility we intend to explore in future work.

As an application of our framework, we have simulated peers navigating a hyperlinked collection of documents, and shown that their activity results in an interesting page distribution.

Our models, as well as the visualization tool, are publicly available on our lab web site[6].

Acknowledgement

Our summer interns Denis Dionne and Matthew Smith contributed valuable development work on this project.

References

[1] Gnutella: The gnutella protocol specification 0.4. rfc-gnutella.sourceforge.net/developer/stable/index.html, 2002.

[2] I. Baumgart, B. Heep, and S. Krause. OverSim: A flexible overlay network simulation framework. In *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA*, pages 79–84, May 2007.

[3] R. Bhagwan, S. Savage, and G. M. Voelker. Understanding availability. In M. F. Kaashoek and I. Stoica, editors, *IPTPS*, volume 2735 of *Lecture Notes in Computer Science*, pages 256–267. Springer, 2003.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Seventh International World-Wide Web Conference (WWW 1998)*, 1998.

[6] http://www.nmai.ca

[5] A. Craig, A. Davoust, and B. Esfandiari. A distributed wiki system based on peer-to-peer file sharing principles. In *Proceedings of the 2011 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2011, Lyon, France, August 23 - 25, 2011*, 2011.

[6] G. Dán and N. Carlsson. Power-law revisited: large scale measurement study of p2p content popularity. In *Proceedings of the 9th international conference on Peer-to-peer systems*, IPTPS'10, pages 12–12, Berkeley, CA, USA, 2010. USENIX Association.

[7] N. Daswani, H. Garcia-Molina, and B. Yang. Open problems in data-sharing peer-to-peer systems. In *Proceedings of the 9th International Conference on Database Theory*, ICDT '03, pages 1–15, London, UK, 2002. Springer-Verlag.

[8] A. Davoust. Collaborative knowledge construction in a peer-to-peer file sharing network. Master's thesis, Carleton University, 2009.

[9] A. Davoust and B. Esfandiari. Towards semantically enhanced peer-to-peer file-sharing. *Journal of Software*, 4, 2009.

[10] T. T. A. Dinh, G. Theodoropoulos, and R. Minson. Evaluating large scale distributed simulation of p2p networks. In *Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, DS-RT '08, pages 51–58, Washington, DC, USA, 2008. IEEE Computer Society.

[11] T. M. Gil, F. Kaashoek, J. Li, R. Morris, and J. Stribling. P2Psim: a simulator for peer-to-peer (P2P) protocols, 2006.

[12] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 314–329, New York, NY, USA, 2003. ACM.

[13] T. Kim, M. H. Hwang, D. Kim, and B. P. Zeigler. Devs/ns-2 environment: integrated tool for efficient networks modeling and simulation. In M. J. Ades, editor, *SpringSim (2)*, pages 219–226. SCS/ACM, 2007.

[14] A. Klemm, C. Lindemann, M. K. Vernon, and O. P. Waldhorst. Characterizing the query behavior in peer-to-peer file sharing systems. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, IMC '04, pages 55–67, New York, NY, USA, 2004. ACM.

[15] A. Klemm, C. Lindemann, and O. P. Waldhorst. Relating query popularity and file replication in the gnutella peer-to-peer network. In P. Buchholz, R. Lehnert, and M. Pióro, editors, *MMB*, pages 305–314. VDE Verlag, 2004.

[16] A. Montresor and M. Jelasity. Peersim: A scalable p2p simulator. In H. Schulzrinne, K. Aberer, and A. Datta, editors, *Peer-to-Peer Computing*, pages 99–100. IEEE, 2009.

[17] The Network Simulator NS-2. http://www.isi.edu/nsnam/ns/, 2001.

[18] W. Siberski and U. Thaden. A simulation framework for schema-based query routing in p2p-networks. In *Current Trends in Database Technology - EDBT 2004 Workshops*, volume 3268 of *Lecture Notes in Computer Science*, pages 510–510. Springer Berlin / Heidelberg, 2005.

[19] D. Stutzbach and R. Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 189–202, New York, NY, USA, 2006. ACM.

[20] R. Toivonen, J.-P. Onnela, J. Saramaki, J. Hyvonen, and K. Kaski. A model for social networks. *Physica A: Statistical and Theoretical Physics*, 371(2):851–860, 2006.

[21] A. Varga. The OMNET++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference*, pages 319–324, Prague, Czech Republic, June 2001. SCS – European Publishing House.

[22] G. Wainer. Cd++: a toolkit to develop devs models. *Software: Practice and Experience*, 32(13):1261–1306, 2002.

[23] G. A. Wainer, R. Madhoun, and K. Al-Zoubi. Distributed simulation of devs and cell-devs models in cd++ using web-services. *Simulation Modelling Practice and Theory*, 16(9):1266 – 1292, 2008.

[24] W. Yang and N. Abu-Ghazaleh. GPS: A general peer-to-peer simulator and its use for modeling bittorrent. In *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 425–434, Washington, DC, USA, 2005. IEEE Computer Society.

[25] Y. H. Yu and G. Wainer. ecd++: an engine for executing devs models in embedded platforms. In *Proceedings of the 2007 summer computer simulation conference*, SCSC, pages 323–330, San Diego, CA, USA, 2007. Society for Computer Simulation International.

[26] B. P. Zeigler, H. Praehofer, and T. G. Kim. *Theory of Modeling and Simulation, Second Edition*. Academic Press, 2 edition, January 2000.

[27] A. Zengin. Large-scale integrated network system simulation with devs-suite. *TIIS*, 4(4):452–474, 2010.