# Bridging The Gap: Augmenting Centralized Systems with P2P Technologies*

Jay A. Patel and Indranil Gupta
Department of Computer Science
University of Illinois at Urbana-Champaign
201 N. Goodwin Ave, Urbana, IL 61801
{jaypatel,indy}@cs.uiuc.edu

## ABSTRACT

The last few years have seen a burgeoning of p2p designs, but also a worrisome push back and skeptical resistance against much of this research (and development). We propose a new research direction that seeks to find common ground between the purely centralized architecture and the p2p systems. We propose a new class of systems called "*P2P-izers*", which incentivize existing centralized infrastructures to utilize p2p solutions. A P2P-izer ensures that p2p technologies compliment existing centralized solutions. Finally, we show the feasibility of this direction by experimentally quantifying the advantages of a simple proof of concept P2P-izer solution for website hosting.

## 1. MOTIVATION

In the last few years, the peer-to-peer systems research community has experienced a boom in new ideas, technologies, algorithms, and even potential applications for several distributed computing problems. However, at the same time, the p2p community has also faced considerable push back from vendors, proponents and supporters of existing centralized solutions, i.e., "non-p2p" solutions to the same problems. Today, the field of researchers is highly polarized into "pro-p2p" and "anti-p2p" camps.

For instance, while the pro-p2p community has proposed "serverless" (hence p2p) email applications such as POST [6], a common argument from anti-p2p camp is that "SMTP/POP (and others) works just fine - why mess with it?" In other words, the existing centralized (hence non-p2p) solution works just fine - "if it ain't broke, don't fix it." Another instance is backup and data storage schemes - the first wave of p2p systems [1, 7, 10] were in fact distributed backup systems. Their popularity has waned due to the (correct) argument that today's centralized tape backup systems, typically managed separately at each institution or University department, are more economical, manageable, and come equipped with system administrators experienced in troubleshooting them.

The above anecdotes point to several reasons for the push back against p2p systems. We propose a new approach called the *P2P-izer* that can incentivize both camps to use p2p technologies (not the only approach by any means!). In a nutshell, a P2P-izer allows an existing centralized solution utilize a p2p system to improve (its own) performance. In some instances, the centralized and the p2p solution work *side-by-side* rather than mutually exclusively. P2P-izers address adaptivity of p2p systems, making them deployable alongside centralized systems. P2P-izers aim to reduce the "gap" between centralized systems and p2p systems by designing systems with the intended goal of graceful collaboration. Note that this paper does not seek to solve the political problems of the pro vs anti-p2p camps, but proposes a small first step.

With the emergence of the next generation Internet (cf., potential efforts towards GENI), it becomes very important to think about how centralized solutions and p2p solutions can coexist and collaborate. Only when there is graceful collaboration between these two classes of solutions can new solutions for the Internet operate in conjunction with existing ones, thus offering users the best of all worlds performance.

Finally, we show that P2P-izer design can be very simple, and present experimental results from a proof of concept P2P-izer designed for website hosting. The key advantage of using a web P2P-izer is that it accommodates flexibility in managing server machines, without degrading the quality-of-service to the clients. For instance, in the absence of the P2P-izer, a transient increase in the traffic load (with a restrictive bandwidth availability) may result in several client requests being delayed, or even dropped. In the presence of the P2P-izer, the ISP is given flexibility in maintaining high quality of service at the client-end without the need for overprovisioning at the server-end.

**Other Related Work:** Several researchers have made cases for the extended use of p2p technology. Foster et al [2] and Seltzer et al [5] argued for the convergence of p2p and Grid technologies. Roussopoulos et al [9] classify situations in which p2p architectures provide substantial benefit. These approaches can all be used orthogonally with the P2P-izer approach.

## 2. A SMALL STEP TO HEAL THE GAP

We believe that the gap between anti- and pro- p2p researchers can be bridged by building solutions that allow p2p technologies to be used *along-side* (and not as a replacement for) centralized solutions to the same problem.

### 2.1 An United Front

A unified approach allows deployers of the centralized solution to harness the benefits, robustness and scalability of p2p technologies without letting go of the time-tested centralized solution. At the same time, this approach allows p2p technologies to gracefully supplant the centralized solution(s), if proven successful in real world usage.

The most important design principle here is one of *graceful collaboration* between the p2p solution and the centralized solution. In order to achieve this, there is a need for a system that sits in between the centralized solution and the p2p solution. We call such an intermediary system the *P2P-izer*. The P2P-izer, *at run-time*, decides either whether the centralized solution is to be used or the p2p solution, or decides what proportion (e.g., fraction of time) each is to be used. An overview of the P2P-izer concept is presented in

*This work was supported in part by NSF CAREER grant CNS-0448246 and in part by NSF ITR grant CMS-0427089.

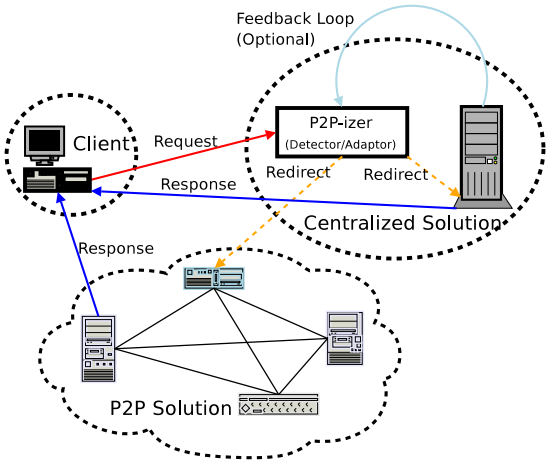


**Figure 1: The design of a P2P-izer application.**

Figure 1.

We have identified two broad classes of P2P-izers : (1) *bimodal detectors*, and (2) *hybrid adaptors*. As shown in Figure 2, a **bimodal detector** specifies conditions under which *either* the centralized *or* the p2p solution will be used exclusively – it does so by detecting transition stimuli at run-time. On the other hand, a **hybrid adaptor** allocates part of the responsibilities to the centralized solution, and part (i.e., the remnant) to the p2p solution.

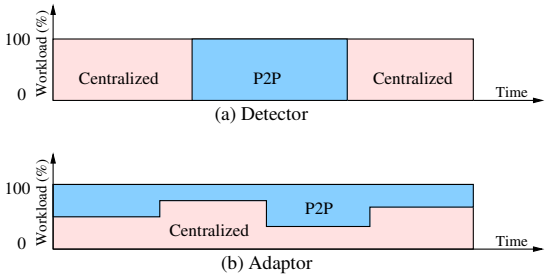


**Figure 2: The two types of P2P-izers: a detector transitions between a centralized solution to a p2p solution, whereas an adaptor seeks to exploit both solutions simultaneously.**

For instance, for website hosting, flash-crowd detectors can be used to transition from server-only to a cooperative caching mode, based on the detection of a stimulus (in this case the arrival of a flash crowd [8]). Alternately, a P2P-izer adaptor can be used to determine what fraction of client requests are serviced by the server and what fraction by the p2p cooperative cache. We describe the latter design (which is new) in the next subsection.

**Trends:** The continued development of p2p technologies will likely lead to a world where deployments combine both elements of centralized solutions and p2p solutions. We believe that future applications will be derived using this hybrid approach. Many of the current p2p design can easily be adapted to be used with centralized solution.

- P2P storage systems can be used to perform redundant backups of the enterprise backup systems. A p2p back end may be used for low-cost long-term archival, whereas a centralized system maybe used to provide daily and weekly backups.
- Another area which may benefit from a P2P-izer approach is workload distribution within Utility Grids. In such systems, a scheduling node (which is kept updated on system status) assigns workloads to other nodes. At times when a scheduling node is overburdened, it may utilize a p2p approach to defer the task of workload assignment to secondary nodes. The secondary nodes can be used temporarily to assign workloads, until the primary node revokes that responsibility.
- A popular domain for the use of P2P-izers can be within E-mail services. Generally, e-mails are spooled to an ISP's server and sent out to another ISP's server. However, if the target e-mail server (and its backup server, if any) does not respond, the sending server periodically checks the status of the target server. The relaying of e-mail is delayed until the target server becomes operational again. A backup p2p e-mail system (such as POST [6]) can be deployed to reduce the "down time".
- Content distribution can also be improved upon by P2P-izers, and we illustrate a proof of concept example in the next section.

In a nutshell, we believe that P2P-izers can bootstrap the growth of p2p services by leveraging today's popular centralized infrastructures.

## 2.2 An Adaptor P2P-izer for Web Servers

In this section, we argue that P2P-izers can be very simple in design, yet can have a vast impact on performance. Consider a ISP sysadmin that wishes to allocate only a portion of its resources to a given website. Further, the sysadmin may wish to dynamically change the allocation of resources to this website, perhaps transiently (for example, in view of satisfying preferred high-paying customer's SLAs, or, to try out different configurations, fit in extra websites, etc.), *without the need for over provisioning*. In effect, the sysadmin of the web server desires more *flexibility in managing the website*, without any *associated loss in performance at clients.*

A p2p solution for cooperative web caching can be used in conjunction with the web server in this scenario. While any webserver and any cooperative caching solution can be used, we use an Apache-compatible web server and the Overhaul cooperative caching solution [8].

We develop a simple P2P-izer that takes as input the *server bandwidth cap*. The response of our P2P-izer is simple: it takes the $k$ most popular documents (calculated at run time) for the website and feeds them to a cooperative cache (instead of serving them directly to the client). The value of $k$ is dynamically adjusted to maintain utilization below the specified bandwidth cap. An unrestricted (or sufficiently high) bandwidth cap thus results in $k = 0$ (i.e., a purely centralized solution). A restrictive cap implies a non-zero value for $k$. Finally, the P2P-izer allows the bandwidth cap to be adjusted at run time.[1]

The key advantage of using a P2P-izer is that it accommodates certain *flexibility* in managing server machines, without degrading the quality-of-service to the clients. For instance, in the absence of the P2P-izer, a transient increase in the traffic load (with a restrictive bandwidth availability) would result in several client requests

[1]We do not present the implementation details of our proof-of-concept web adaptor in this paper, as it does not seek to optimally exploit available bandwidth. Rather, our implementation takes a very conservative approach, and *aggressively* adapts the value of $k$ (i.e., sets $k$ to a high value) due to the unpredictable nature of (future) web requests. A better web adaptor may be designed in the future by using control theory models [4].

being delayed, or even dropped. In the presence of the P2P-izer, the ISP is given flexibility in maintaining high quality of service at the client-end without the need for overprovisioning at the server-end. For example, this allows website hosting ISPs to bound bandwidth utilization, etc. on a per-site basis.

# 3. THE WEB P2P-IZER: EVALUATION

We perform a set of experiments to evaluate the performance of our proposed Web P2P-izer. We emulate a part the Internet topology and replay portions of actual website traces in real-time.

## 3.1 Experimental Setup

We use the ModelNet [11] network emulator for our experiments. ModelNet emulates large-scale networks over a compute cluster by mimicking the properties of a specified virtual topology. All network traffic flows through a designated router core, where each packet is processed to adhere to the specified topology (i.e., by slowing down the packet flow accordingly). Furthermore, each physical host can support a great many virtual nodes. Hence, our 8-machine experimental cluster can support up to 1600 virtual nodes.

In our experiments, we use the Internet AS topology as produced by Zhang et al [12] on October 1, 2005. Our underlying assumption is that the network bottleneck in HTTP transactions is primarily at the final hop. Hence, we assign plentiful network capacity to each AS link, with each transit-transit link having a full-duplex capacity of 20 Mbps and each stub-transit link having 10 Mbps capacity. To model latency, we use a uniform latency of 5 ms per hop.

We map virtual nodes (i.e., end systems) to the Internet AS topology by randomly choosing a stub node as an ISP. We use the following methodology to determine the network capacity for the final hops: (i) One of the virtual nodes serves as the web server, its final hop capacity is 5 Mbps (this is the default, changes are described in section 3.2); (ii) The remaining virtual nodes are used to model website clients, 50% of which are connected by dial-up connections (we base this on a September 2005 Pew Internet Project report [3], which finds that 53% of US residential Internet connections are broadband enabled). The speed of dial-up connections is 33.6/56.6 Kbps (uplink/downlink capacity). The broadband connections are partitioned into two equal segments based on current low-grade (i.e., low cost) residential market offerings. Half the broadband connections have a 128/384 Kbps capacity, while the rest are modeled with 384/1500 kbps capacity. We believe this is a *staunchly* conservative approach to modeling the final hop capacity of Internet users.

We replay the traffic of a real website on our experimental test bed. The trace was collected during an hour of regular operations from a community website for Macintosh enthusiasts. The trace exhibits 718 unique clients making 8129 document requests. We map each unique client (identified by IP) to a virtual node on the ModelNet/Zhang topology. We reconstruct the corpus of server documents by generating a random document (of equivalent file size) for each URL, as present in the trace. However, if a document is listed with multiple sizes (i.e., possibly either because it is a dynamic document, or because the document was updated), we map the URL to multiple files with appropriate sizes.

## 3.2 Results

For our first experiment, we replay the traces with regular HTTP protocol, and with the P2P-izer Web adaptor (using various static values of $k$). Figure 3 summarizes our results. We find that using a low value of $k$ provided negligent benefit (i.e., when $k$ is below 10) for this particular trace. In fact, with a lower value of $k$, the P2P-izer utilizes more bandwidth than regular HTTP, as the overhead for transmitting the extra headers (of the Overhaul protocol extension) exceed the benefits gained by redirecting the top $k$ requests (which for this trace, were primarily tiny image files). However, as the value of $k$ increases, the benefit of the P2P-izer becomes significant.

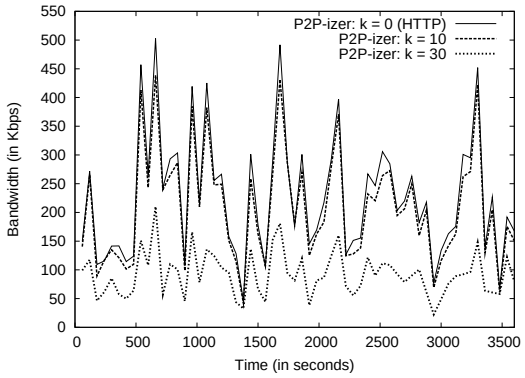


**Figure 3: The server bandwidth utilized (sampled every minute) using a static $k$ value for the P2P-izer adaptor. An increased value of $k$ induces a significant drop in bandwidth utilization.**

In our next experiments, we show the performance of the P2P-izer adaptor under resource constraints. In our first contrived scenario (hence **scenario #1**), there is plentiful bandwidth available for the first twenty minutes, only 250 Kbps for the next twenty, and merely 150 Kbps for the last twenty. This situation may occur in a shared server-farm, when a high-priority SLA (for a high-paying customer) needs to be met during above-normal transient traffic loads. To remedy this situation, a ISP may deploy the P2P-izer web adaptor for budget customers. Figure 4 shows the bandwidth utilized for the base case (HTTP protocol with surplus available bandwidth), and with the P2P-izer (with resource constraints as prescribed by the first scenario). The bottom chart shows the dynamically changing (automatically set by the P2P-izer adaptor) $k$ value.

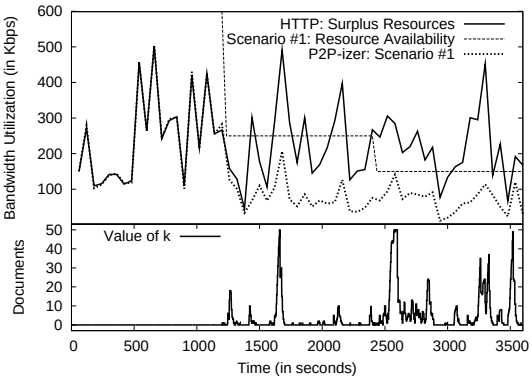


**Figure 4: Scenario #1: The P2P-izer adaptor introduces an entirely new notion of "flexible resource management": the bandwidth utilization remains (conservatively) below the prescribed dynamic cap.**

In our second scenario (hence **scenario #2**), only 300 Kbps (fixed) bandwidth is available. This models the case where a web publisher purchases a strictly fixed supply of bandwidth resources. An impor-

tant distinction between the two scenarios is that data packets may be transmitted 5 Mbps (fully-burstable server-farm model, simply requiring the average utilization – sampled every minute, to remain below the bandwidth cap) in the first scenario, whereas only 300 Kbps (dedicated leased-line model) is available in the second scenario. Figure 5 plots the performance of both HTTP and P2P-izer adaptor under the second scenario.

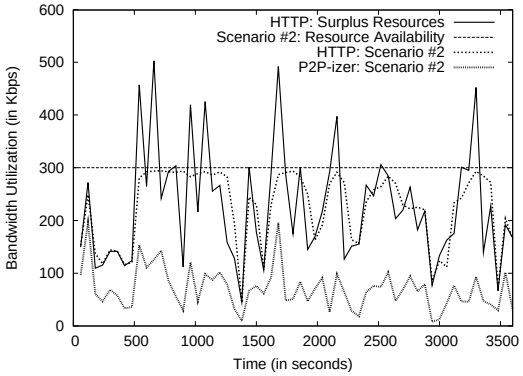


**Figure 5: Scenario #2: The P2P-izer adaptor (in contrast to HTTP) is capable of serving client requests without saturating the available bandwidth resources (i.e., without degrading the quality-of-service to future clients).**

Finally, we show the request turnaround time at the client-end in Figure 6. The best-case HTTP (i.e., with surplus resources) provides the best performance. However, the tail-end of the requests (i.e., approximately 2% of requests with turnaround time greater than 60 seconds) is dominated by large file requests. For the first scenario, the P2P-izer web adaptor is able to perform sufficiently well in comparison to best-case HTTP. The P2P-izer Web adaptor is able to satisfy 97% of requests within approximately 45 seconds, much like best-case HTTP. We believe that this performance can be further improved by choosing peers based on geographic proximity (Overhaul does not do this yet). Another way to improve performance is to improve the uplink capacity of clients (remember, we use a conservative final-hop capacity model), as bandwidth available at the server (5 Mbps) is far greater than what is available at any of the peers (a mere 384 Kbps).

On the other hand, the P2P-izer web adaptor clearly excels in limited-bandwidth conditions (as prescribed in scenario #2). It is able to perform just as well as best-case HTTP by satisfying approximately 97% of requests within 45 seconds. Whereas, HTTP under resource restriction (as prescribed by the second scenario) performs significantly worse with only 90% of requests satisfied within 45 seconds, with a noticeably slower tail.

# 4. CONCLUSION

The boom of p2p technologies has led to a push back from anti-p2p researchers, who contend that today's mostly-centralized solutions "will work just fine". To address these problems, we have proposed a new class of systems called *P2P-izers* that effectively allow p2p solutions to be used to improve the performance of existing of centralized solutions (rather than replacing them immediately with serverless alternatives). This gives flexibility to the solution deployer, while maintaining client satisfaction, and leaves open the door for eventual graceful deployment of p2p technology. We show that P2P-izers are simple to design, but can be very effective.

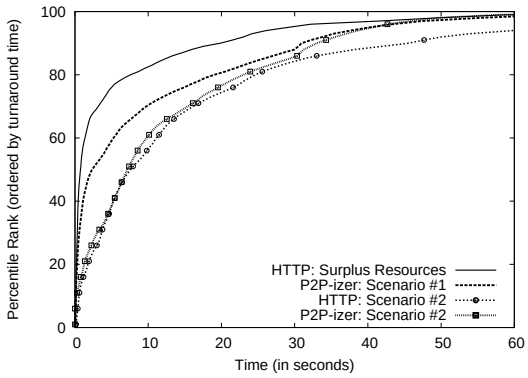


**Figure 6: The quality of service at the client-end remains approximately the same (i.e., converges at approximately 97%) for the P2P-izer in comparison to HTTP (with surplus resources). However, HTTP with restricted resources performs significantly worse.**

# 5. REFERENCES


[1] F. Dabek, M. F. Kaashoek, D.R. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proc. of ACM SOSP*, pages 200–215, 2001.

[2] Ian Foster and Adriana Iamnitchi. On death, taxes, and the convergence of peer-to-peer and grid computing. In *Proc. of IPTPS*, 2003.

[3] John B. Horrigan. Broadband adoption in the United States: Growing but slowing. In *Proc. of Telecommunications Policy Research Conference*, September 2005.

[4] Christos Karamanolis, Magnus Karlsson, and Xiaoyun Zhu. Designing controllable computer systems. In *Proc. of HotOS*, 2005.

[5] J. Ledlie, J. Shneidman, M. Seltzer, and J. Huth. Scooped, again. In *Proc. of IPTPS*, pages 129–138, 2003.

[6] Alan Mislove, Ansley Post, Charles Reis, Paul Willmann, Peter Druschel, Dan S. Wallach, Xavier Bonnaire, Pierre Sens, Jean-Michel Busca, and Luciana Arantes-Bezerra. POST: A secure, resilient, cooperative messaging system. In *Proc. of HotOS*, pages 61–66, 2003.

[7] Athicha Muthitacharoen, Robert Morris, Thomer M. Gil, and Benjie Chen. Ivy: A read/write peer-to-peer file system. In *Proc. of OSDI*, 2002.

[8] Jay A. Patel and Indranil Gupta. Overhaul: Extending HTTP to combat flash crowds. In *Proc. of WCW*, pages 34–43, 2004.

[9] Mema Roussopoulos, Mary Baker, David S. H. Rosenthal, TJ Giuli, Petros Maniatis, and Jeff Mogul. 2 P2P or Not 2 P2P? In *Proc. of IPTPS*, pages 33–43, 2004.

[10] Antony Rowstron and Peter Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Proc. of ACM SOSP*, pages 188–201, 2001.

[11] Amin Vahdat, Ken Yocum, Kevin Walsh, Priya Mahadevan, Dejan Kostic, Jeff Chase, and David Becker. Scalability and accuracy in a large-scale network emulator. In *Proc. of OSDI*, pages 271–284, 2002.

[12] Beichuan Zhang, Raymond Liu, Daniel Massey, and Lixia Zhang. Collecting the Internet AS-level topology. *ACM SIGCOMM CCR*, 35(1):53–61, January 2005.