P2P Research and Reality: Some Preliminary Thoughts

Zheng Zhang
Microsoft Research Asia
zzhang@microsoft.com

Abstract

In this paper, we start by recounting some of the myth and truth of P2P systems. Getting a good understanding of those issues are good starting points. Even more important will be performing researches that understand the fundamental properties, and that actually build the environment and forester the creativition of new applications.

1. Introduction

Relatively speaking, P2P research has a short history. However, the amount of world-wide efforts that has been devoted to this topic is impressive. As a result, the underpinning theories are getting mature. What is also encouraging is that researchers from China have also made considerable strikes.

Nevertheless, running systems and applications that utilize the latest theoretical results are relatively lacking. Without building real systems and conduct solid experiments to obtain further insights, it will be hard to push to the next stage.

In this paper, we give a very brief overview of P2P systems (Section 2), and then suggest some research topics and approaches (Section 3). The paper is hastily put together, the references are not complete, and many points might be wrong. Nevertheless, we hope this will be enough to get the discussion going.

2. A simplified view of P2P systems

One definition of a P2P system is that it is decentralized and symmetric, from a functional perspective of each participating entity. An ideal P2P system, however, should be decentralized and *asymmetric* – one in which more powerful and able peers are providing more, but the system as a whole does not necessarily rely on these nodes to function correctly.

There are many ways to slice and dice different P2P overlays. Let us try a thought-experiment to show that, after all, things are not that different.

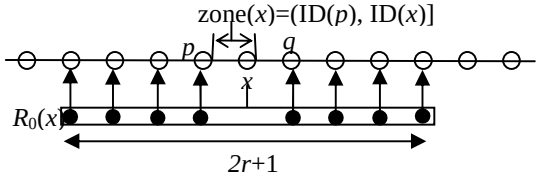

Figure 1: The simplest P2P DHT – a ring

Figure 1 illustrates the simplest DHT (distributed hash table, or structured P2P) where nodes line up in a 1-dimension logical space, with every node remembering a constant set of neighbors to each side in what's known as the *leafset*. Generalizing it to *d*-dimension torus, one gets CAN[13] with O($N^{1/d}$) routing performance. Adding O(log*N*) directional fingers whose targeting nodes are spaced with exponentially increasing logical gaps in between yields proposals such as Chord[18], Pastry[16] and Tapestry[24], with O(log*N*) routing performance. Adding denser fingers gains higher performance, all the way to O(1) steps with O(*N*) fingers, but typically with additional maintenance overhead.

There are several important things to pay attention to. First, the most rudimentary data structure in the so-called structured P2P is the leafset, *not* the fingers. Fingers are there purely as routing optimizations, with the sweet-spot being O(log*N*) fingers. While leafset maintenance has to employ failure detection mechanisms (i.e. periodical heartbeats), there exists different approaches to update fingers. For instance, the 1-hop DHT[1], XRing[21] and SmarBoa[8] all use multicast to update fingers. Second, the quality – instead of quantity, of the state (e.g. leafset and finger table) matters more. Stale entries will result in routing timeout which may cost 10x more than a routing hit. Third, depending on the application scenarios and contexts, we should pragmatically choose different DHTs. For an environment where churn is low or when scale is limited, 1-hop DHT makes perfect sense. There has not been a one-size-fit-all proposal, and doubtful if there is one with low complexity. Finally, a *perfect* 1-hop presents an interesting design point, in that it is equivalent to an eventually reliable global membership service, and can be a building block for large and scalable distributed system.

If we now remove the leafset and let the fingers be bi-directional, we arrive at what is known as unstructured P2P. The distribution of fingers in unstructured P2P can be different, and so are the construction, tuning and maintenance of the fingers. Arguably, any Gnutella-like protocol may be significantly simpler, compared with any of the well-known structured P2P proposals. The absence of leafset dictates that the only semantic that can be reasonably supported is query flooding. For the most up-to-date research results, please refer to [7].

From the above discussion, it is clear that we always deal with a graph. Whether there is a leafset to guarantee the integrity of the space is the key differentiation point. Hence, one can build a structured P2P but use it as an unstructured P2P [3], or augment an unstructured

P2P with a leafset to provide DHT functionality. The latter is something we have been working on.

3. Potential research topics

3.1 Understanding the fundamentals

Before we build applications, it is fruitful to get a sense of what guarantees that the underlying infrastructure provides and whether they meshes well with the requirements of the applications, if at all. The "pick two" paper [2] is compelling precisely because it pointed out quantitatively (even with a back-of-envelop approach) what many have suspected, that it is impossible to build a large-scale and highly-available P2P storage system when peers constantly come and go (e.g. Oceanstore[9], Pond[15], CFS[4], PAST[17] and Ivy[11]), which is a hopeful candidate that will amount to nothing short of a breakthrough if successfully built.

To give some other examples, considering tuning proximity of overlay neighbors. The goal is to "align overlay topology with that of underlay (IP)" so as to obtain better performance with lower tax on network resource. It is also well-known, however, that this comes with a hefty price when resilience of the network is considered, especially for unstructured P2P overlays. Aggressive tuning will tend to make the system more fragile, in that the connectivity can be broken if nodes (effectively) connecting islands of nodes leave or are under attack. If the system relies on some nodes more than others, this indeed violates the P2P spirit to start with. The amount of work proposing various tuning techniques far outweighs those that do take that into account; and we still lack the knowledge of just when pushing for performance will become dangerous.

Problems in structured P2P can be even more. Given a key *k* in the logical space, two queries that lookup *k* will return different nodes that owns *k* and this can happen because: 1) node dynamism – the space that *k* resides has undergone ownership change and 2) there are network jitters such that different nodes can simultaneously claim ownership of *k*. From a system research perspective, it is an interesting question as what properties that even the simplest get/put APIs hold: for instance what is the bound and guarantee of liveness and safety?

It is possible to adopt some practical measures to build applications without digging deep into the system properties. For instance, it is a popular proposal to keep the invariant such that a number of consecutive copies are kept following the root of *k*, as exemplified by DHash[18]. Yet, as [2] points out, enforcing this invariant for the ambitious design point (dynamic, large-scale, highly-available and wide-area) can be prohibitively expensive. Therefore, a more practical approach may be the one advocated by Tapestry[24], in which the upper-layer application manages the availability of the objects all by itself, and uses the P2P as a repository of *soft-state* data storing pointers. The pointers need to be continuously refreshed.

From the above discussion, it then becomes clear that it is not interesting – and indeed can be quite misleading, to talk about overlay maintenance overhead in an isolated fashion: if one adopts the leafset replication approach, the traffic dedicated to maintaining the invariance must be taken into account; if, instead, we adopt the approach of soft-state pointers, then the republishing traffics are to be included. These traffics are indeed not part of the overhead maintaining the overlay topology, but exist if the infrastructure is to serve *any* state for the application. It is not very meaningful to talk about the overlay maintenance overhead alone[10]. If it turns out that traffic to maintain application state greatly dominates that of maintaining the topology, then the practicality of the structured P2P is questionable.

Optimizing the routing performance (by picking up neighbors carefully) and reducing the base of O(log*N*) routing are well-understood. It will make minor contribution to propose yet another (or several more) O(log*N*) DHT. The larger challenge is to figure out the best performance given that heterogeneity is well explored. Yet this pales in comparison to the need of understanding what "overhead" truly is, its magnitude and impacts.

3.2 Pulling through new applications

3.2.1 Switching the context

Today's P2P researches have mostly rooted in wide-area context, and this is due to several factors: the spectacular rise and fall of Napster (which gave birth to several more content sharing systems), and the distributed system research community's quest to find a much tougher environment and hence more challenging research topics.

This is not the only context that P2P technologies should be applied to. Switching the context means that we can and should let go some of the more popular scenarios and drive towards the core of the technology offerings. For instance, the fundamental attributes that P2P have brought to the table – self-organizing, self-managing and self-healing etc., are extremely useful to reducing the management overhead of large IT infrastructures' TCO (total cost of ownership). As the ROC[12] (Recovery Oriented Computing) initiative points out, it is time to focus away from performance and onto manageability and reliability. In a world where commodity components will prevail to be the building blocks of large systems, failure will be norm rather than

exception [6]. Thus, challenges and opportunities are abundant even in the machine-room scenario: how to easily plug-in a new box and let it be integrated as part of the system, and with what speed? How to detect a failed component and make sure that no data are lost? As data will surely outlive the hardware, how to gradually drop in new generations of hardware and let the older ones phase out, all without a glitch to the users? These are hard and interesting problems which have immediate and practical implications. We have designed and implemented several self-organizing distributed storage that can scale from one box to 100K [22][23], and our experience has taught us at least one thing: that it is not at all trivial to build such a system.

We can also switch the context to the wireless and sensor network setting, where the challenges are quite different: range and reach matters, proximity matters, and probably most important to all – power consumption matters. Again, self-organizing is key. While it is not immediately clear how current P2P research results can bear fruits, we are aware of several works that apply the O(log*N*) DHT and routing to these settings. These approaches are somewhat misled because that if one is willing to let go the sexy O(log*N*) label, it becomes obvious that mesh and hence a 2-d CAN would have worked out the best.

3.2.2 Sharing content responsibly

The number one P2P application is content-sharing. While many works have devoted to improving the performance and search, the larger question is whether the sharing is amenable to the call of being responsible in a human community. If we are to question what are being shared, a large fraction of content will be pornographic materials, pirate copies of entertainment clips and software. This is problematic.

It will be easy to duck one's head under the sand and pretend that these are not the problems that the research community ought to solve; it would be even somewhat "heroic" to come up with novel techniques that would get around the counter-measures: the record industry has mounted some trivial but effective attacks by simply polluting the system with corrupted copies. Granted, the P2P content sharing has its place as a high-tech mirror of old-times underground circulation of anti-censorship materials[1]. Yet, sharing content in a responsible way has far-reaching implications: if software and other intellectual properties are not protected, so won't be the future of the budding Chinese software industry.

[1] In the 70's of last century in China, this is the primary form of passing poetry and other literature bits – by hand and underground.

It can be very difficult technically, but doing content-sharing and yet allowing some degree of DRM (digital right management) seems ripe to be tackled. Probably the first that is required is to define the appropriate scenario. The P2P content sharing can continue to serve as the distribution venue, should we then add enforcement in the content itself, or in the P2P system to give the possibility of tracking the flow? The ideal case should be a win-win situation: those who contributed to the content (artists and the record industry, software vendors) get their due, and the users find it easy to obtain the trial copies and, if they indeed enjoy them, pay as they go. Notice that this does not necessarily break the functionality of anti-censorship support: authors can anonymize their contents, but it ought to be possible to track those who injected pirated copies.

3.2.3 Harness the computing power

seti@home is a telling story of how many idle resources there are and how much power lies therein: it is the biggest computer on the planet.

Thus, P2P computing continues to be an interesting research area. It will be great to find the next seti@home application. However, for system researchers, the more interesting problem is to find applications that are not embarrassingly parallel to start with. For example, there are many large scale simulations that require a process to communicate to a set of other processes. It is then a mundane requirement that these neighbors are connected among them with low-latency and high-throughput links. This requires techniques of positioning and finding peers, and which has so far not been used much (DHT of course uses this heavily, but again DHT itself lacks applications). There are several other non-technical factors: the source of the computation should have the incentive to publish the initial set of data free to every participants, and the results of computation is of some interest and value to the general public – think of real-time weather prediction in a tight schedule or, in the other extreme, into a very long future (e.g. 5 years ahead); think of rendering of the evolution history, or simulating a human brain.

3.2.4 Finding other applications

In his keynote speech at SOSP'99, Butler Lampson stated that "the biggest mistake of the system research in the past 10 years is not having invented the Web." So, what can we do right this time?

We should enable a testbed which is open for *everyone* to try. If we are not that creative to foresee the new breed of applications, let us at least contribute by setting up the stage. In summer of this year, MSR-Asia and NSFC will co-found a wide-area, windows-based test-

bed called ImagineONE.net. With Butler's reminder in mind, the codename can not be more appropriate. This testbed is not to be built in one day, and there are many hard problems to solve. They range all the way from resource-isolation issue on a single machine among simultaneous experiments, to appropriating distributed resources per single experiment, and to building other necessary tools. Researchers in China have already accumulated many great experiences doing Grid computing, it is hopeful that some of the technologies can be leveraged.

There are many interesting P2P applications in the web scenario, such as searching[14], spam fighting [25], even troubleshooting [19]. It will also be useful to see if P2P can be used as sensors in the network to detect worms and viruses. Collaborative and interactive learning, P2P gaming etc. are also interesting scenarios.

3.3 Learning through building

It is encouraging to see that researchers in China have quickly caught up. However, most results are mathematic deductions or simulations. Also, topics such as O(log*N*) DHT that are getting increasingly lukewarm acceptance in international research community are still pursued with great rigor. What is perhaps more troubling of all is that there is a lack of concerted effort to actually build and deploy P2P applications. With or without an open platform such as ImagineONE.net, it is important to realize that we only learn through building, especially at this stage when theories are getting mature, and that we don't get to the next stage of insights unless we engage more hands-on practice.

When we do build the system, it is critical to perform concrete experiments to mine the lessons and insights, and make traces and logs available for the community at large. Platforms such as MAZE from Beijing University are ripe for this kind of activities.

4. Conclusion

For system research, theory and practice go hand-in-hand. Furthermore, practice to build application depends on the practice to build infrastructure, and there is a reverse dependencies between the two. P2P systems are fun to study and build, and it is time to examine and take actual steps to cover these grounds.

References

[1] Gupta, A., Liskov, B., and Rodrigues, R. "One Hop Lookups for Peer-to-Peer Overlays", HotOS IX, 2003, Hawaii, USA.

[2] C. Black, R. Rodrigues, "High Availability, Scalable Storage, Dynamic Peer Networks: Pick Two", HOTOS'03.

[3] M. Castro, M. Costa and A. Rowstron**,** "Should we build Gnutella on a structured overlay?" HotNets-II, Cambridge, MA, USA, November 2003.

[4] F. Dabek, M.F. Kaashoek, D. Karger, et al, "Wide-area cooperative storage with CFS", SOSP'01.

[5] S. Frolund, A. Merchant, Y. Saito, et al, "FAB: enterprise storage systems on a shoestring", HOTOS'03.

[6] S. Ghemawat, H. Gobioff, S.T. Leung, "The Google File System", SOSP'03.

[7] Krishna P. Gummadi et al. "The Impact of DHT Routing Geometry on Resilience and Proximity," Sigcomm'03.

[8] J. Hu, M. Li et al. "Smartboa: Constructing p2p Overlay Network in the Heterogeneous Internet using Irregular Routing Tables," IPTPS'04

[9] J. Kubiatowicz, D. Bindel, Y. Chen, et al, "OceanStore: An Architecture for Global-Scale Persistent Storage", ASPLOS'00.

[10] R. Mahajan, M. Castro and A. Rowstron, "Controlling the Cost of Reliability in Peer-to-peer Overlays", IPTPS'03

[11] A. Muthitacharoen, R. Morris, T. M. Gil, et al, "Ivy: A Read/Write Peer-to-peer File System", OSDI'02.

[12] D. Patterson, A. Brown, P. Broadwell, et al, "Recovery Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies", UCB Technical Report No. UCB/CSD-02-1125.

[13] S. Ratnasamy, P. Francis, M. Handley, et al, "A Scalable Content-Addressable Network", SIGCOMM'01.

[14] Reynolds, P. and Vahdat, A.. "Efficient Peer-to-Peer Keyword Searching". Middleware, 2003.

[15] S. Rhea, P. Eaton, D. Geels, et al, "Pond: the OceanStore Prototype". FAST '03

[16] A. Rowstron, P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems", IFIP/ACM Middleware'01.

[17] A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility", SOSP'01.

[18] I. Stoica, R. Morris, D. Karger, et al, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", SIGCOMM'01.

[19] Wang J. H. et al. "Friends Troubleshooting Network: Towards Privacy-Preserving, Automatic Troubleshooting". In IPTPS'04

[20] Q. Xin, E. L. Miller, T. Schwarz, et al, "Reliability Mechanisms for Very Large Storage Systems", Mass Storage System'03.

[21] Z. Zhang, Q. Lian, Y. Chen, "XRing a Robust and High-Performance P2P DHT", Technical Report.

[22] Z. Zhang, S.D. Lin, Q. Lian, et al, "RepStore: A Self-Managing and Self-Tuning Storage Backend with Smart-Bricks", ICAC'04.

[23] Z. Zhang et al. "BitVault: a Highly Reliable Distributed Data Retention Platform", paper under submission.

[24] B.Y. Zhao, J. Kubiatowicz, A.D. Josep, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing", UCB Technical Report No. UCB/CSD-01-1141.

[25] Zhou, Feng et al. "Approximate Object Location and Spam Filtering on Peer-to-Peer Systems". Middleware 2003.