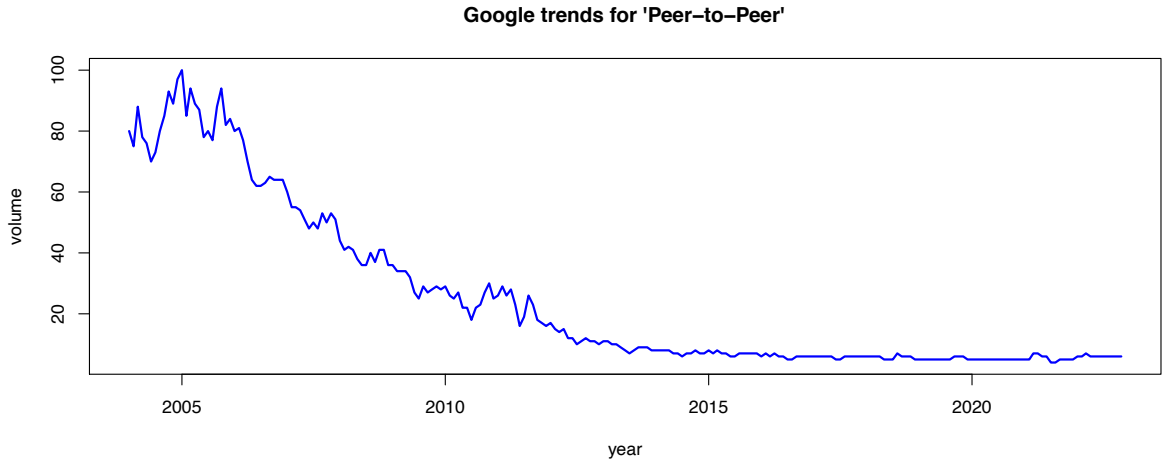
The Legacy of Peer-to-Peer Systems

Peer-to-Peer (P2P) systems became famous at the turn of the millennium, mostly due to their support for direct file sharing among users. In the 80s, the music industry had already evolved from selling analogue vinyl records to digital compact disks, but only with the introduction of lossy data-compression techniques, such as the MP3 coding format, it became feasible to upload/download music files among users' personal computers. Still, content had to be catalogued and found, and P2P systems such as Napster emerged to provide that functionality.

Some of the early systems, such as Napster and SETI@Home, exhibited a mix of P2P and classic client-server architecture. Gnutella and Freenet, the second generation of systems, provided a much larger degree of decentralization. The emergence of P2P greatly impacted the business models of the music, and later film, industries. With time, these industries evolved to offer flat rates and subscription services decreasing the incentives for, music or video, file copying and sharing.

It is not absurd to say that Netflix and other subscription services are a consequence of P2P. But what happened to **peer-to-peer** as a technological concept?



Looking at Google trends, we see that the concept almost faded from our lexicon. Nevertheless, the technology is still used; it evolved and became more specialized. A good portion of the fabric beneath modern data centers (web 2.0) and blockchain technology (web 3.0) evolved from early P2P research. Let's consider some examples.

Search and Lookup

Search engines predated the rise of P2P and had significant development in the 90s. Initially, they were simple inverted indexes that mapped relevant words into document locations on the web. The first inverted indexes did not try to classify the importance of the documents that contained a given word, making difficult the selection of the most relevant choices among the results. In the late-90s this limitation was removed with the introduction of PageRank, which used the linking

structure to rank the importance of pages; later Google was created by leveraging that functionality. However, search is easier in a centralized setting.

Gnutella, born in 2000, was one of the first fully decentralized P2P systems. But the search was very primitive, words of interest were propagated to all nodes, by flooding, and nodes replied to the source if they had documents matching the search terms. There was no ranking. Some users even modified their local node code to pretend they always matched the search terms, to spam other nodes with fake documents. It was becoming clear that P2P systems, due to their open and decentralized nature, were particularly subject to attacks. More on this later.

Around the same time, the Freenet system was introducing additional advances. To avoid document manipulation, it provided *Content Hash Keys*, which are tied to the document content and allow verifying if the returned content matches the key. Another type of key, *Keyword Signed Keys*, allows for storing content under a readable string, e.g. “The Hobbit”, that is transformed by a hash function into a binary key. If different content ended up stored under the same key, the most popular versions would prevail. Both types of keys allowed the lookup of arbitrary data in the network, but efficient lookup was not solved at the time.

Freenet created a topology among the nodes that exhibited small-world properties. This guaranteed that short paths (with a small number of hops) existed among nodes. However, the routing algorithm was yet unable to consistently detect those paths with local information, which impacted the lookup times. An additional ingredient was missing, and that was **locality**.

The next generation of systems, in the early 2000s, solved this problem by introducing topologies that exhibited locality. The closer one came to the target, the more paths one had to the target and the routing algorithms could pinpoint the next hop with local information and a distance metric. Functionally, these systems provided the users with a Distributed Hash Table (DHT). These efficient content addressable networks (Chord, CAN, Pastry, Tapestry) allowed structuring N nodes in a topology that supported log(N) routing steps while only storing log(N) network contacts on each node.

Not all these algorithms were easily translated into usable systems. Keeping a structure with nice properties is hard under churn when nodes are constantly failing and being replaced. A subsequent algorithm, Kademlia, added simplicity and robustness to content addressable networks. It supported symmetric topologies, used a simple XOR distance metric, and allowed concurrent exploration of routes. Nowadays, variants of Kademlia are used to support BitTorrent, Ethereum and IPFS.

High-Availability

The 2000s decade brought the promise of large-scale P2P systems that aggregated end-user machines and servers. Sadly, these systems tended to have a poor quality of service. Portable machines disconnected for the daily commutes and users terminated their P2P processes once their downloads were completed. At the same time, centralized server solutions, typically built on top of SQL databases, were also

finding problems keeping up with the scaling demand from increasing numbers of internet users. Both paradigms were lacking in availability. Interestingly, they occupied two extremes in the design space: either full decentralization or full centralization.

The paradigm shifted again in the late 2000s. In 2007, Amazon's Dynamo presented a pragmatic system design that built on prior research in DHTs and Eventual Consistency (leveraging filesystems research from the early 90s, Coda and Ficus, that exhibited P2P characteristics before the term was fully established). In Dynamo the focus was high availability, and, unlike prior P2P systems, the nodes were placed under the same administrative control and inside the data centers. The number of nodes scaled down from millions to hundreds, albeit more powerful ones, allowing some simplifications on the DHTs. Availability and low response time was now the key concern, it was good for business.

Quoting from the Dynamo paper: *"To meet these stringent latency requirements, it was imperative for us to avoid routing requests through multiple nodes ... Dynamo can be characterized as a zero-hop DHT, where each node maintains enough routing information locally to route a request to the appropriate node directly"*.

The Dynamo design turned out to be very influential to the NoSQL movement of the 2010s. Several databases were designed following the Dynamo footprints (Project Voldemort, Basho Riak, Apache Cassandra) and featured in common the zero-hop DHT approach, while differing in the handling of concurrent updates to replicas. Nowadays, Apache Cassandra is an essential component in modern cloud infrastructure and is used by Apple, eBay, Instagram, and many others.

Free Riders

The phrase "all successful systems attract parasites" is often cited in Biology, and the same can be applied to P2P systems. Filesharing users terminated nodes once their ongoing downloads finished and did not further contribute to the system. Some nodes refused to forward queries from other nodes or lied about their uptimes to improve their position in the network.

Different strategies were tested to control free riders: enforcing download and upload quotas to avoid unbalanced downloads; partitioning files in blocks and sharing them in a random order, to prevent nodes from quitting when they were close to having the full file. These strategies tried to coerce the users into contributing to the collective, but an important tool was still missing: a clear **incentive** system.

Chains and Hashes

Hash functions were already a key technology in early P2P systems. They provided evidence that some content was most likely not manipulated. Hashes can be combined in (chain) sequences and directed acyclic graphs to provide more complex certification. In 2008 Bitcoin was presented as "a peer-to-peer distributed timestamp server to generate computational proof of the chronological order of transactions.". In these blockchain systems, each peer keeps a local copy of all transactions and can try to augment the chain of transactions by competing/collaborating with other nodes

to append a new entry. To prevent Sybil attacks, with one node simulating many other nodes to accrue voting power, these systems can make use of proof of work, spending computation time, or proof of stake, by showing that tokens are held.

Blockchains can both support the creation of new coins and handle the transfer of coins among users (identified by cryptographic key pairs). Coin generation and distribution, particularly when coins can be traded for fiat currency or goods, creates an incentive mechanism to keep the P2P system running. This incentive mechanism was an important ingredient that was missing in the early P2P systems.

Blockchain technology is still evolving fast and looking for winning applications. It might be seen as a continuation of the P2P disruptive revolution 20 years earlier, and its effects on the financial system are already noticeable. We probably need to wait for another 20 years to study **the legacy of blockchain systems** and see which technological concepts turned out to have a lasting impact.

Acknowledgements

I would like to thank Rodrigo Rodrigues, Pedro Henrique Fernandes and Pedro Souto for their comments on improving this text.

Bibliography

Peer-to-Peer : Harnessing the Power of Disruptive Technologies. Edited by Andy Oram. O'Reilly and Associates. March 2001.

Peer-to-peer systems. 2010. Rodrigo Rodrigues and Peter Druschel. Communications of the ACM. Volume 53, Number 1, January 2010. https://cacm.acm.org/magazines/2010/10/99498-peer-to-peer-systems

Carlos Baquero is a professor in the Department of Informatics Engineering within the Faculty of Engineering at Portugal's Porto University, and also is affiliated with INESC TEC. His research is focused on distributed systems and algorithms.