Assignment 5

Georgia Koloniari

2. "Peer-to-Peer Computing"

1. What is the definition of a p2p system given by the authors in sec 1?
Compare it with at least one of the definitions surveyed in the last paragraph of pg 2.

The term "peer-to-peer" refers to a class of systems and applications that employ distributed resources (computational power, data storage and content, network bandwidth and presence) to perform a critical function (distributed computing, data sharing, communication and collaboration, platform services) in a decentralized manner (decentralized algorithms, data, meta-data).
The Intel P2P working group defines it as "the sharing of computer resources and services by direct exchange between systems". Like the first definition this also focuses on the sharing of resources. It emphasizes on the direct exchange between systems, without the interference of a central point of control. However it does not point out the distributed nature of the resources. Furthermore, it fails to recognize that this is done to achieve in most cases a common goal se with distributed computing or collaboration.

2. In Fig 2 (pg 3), the authors compare some aspects of the client-server and the p2p computing models. List and explain these aspects.

The client – server model is managed because the server has a control of the clients it receives and the services it grants them, whereas a peer-to-peer system is self organized since peers are acting autonomously and no one has the role of a manager in their communication. Also the client-server model is configured, the placement of the servers is predetermined and in most cases known and permanent. A server does not change its location and is almost always available apart from failures. In contrast, peer-to-peer systems support ad-hoc networks where peers join and leave the system dynamically as they wish. Furthermore, in a client-server model a client knows the server that offers a specific service or knows who to contact (a central authority) that can inform it a bout its place. In peer-to-peer systems data and services must be discovered since we cannot know the peer that provides them and there is no central authority with global knowledge about the system. Also the client-server model defines a hierarchy where the server is on top of all clients while in p2p there is no superior peer and all peers are equal forming a mesh. As we mentioned before, a server's address is in most cases static so that it can be known by the clients while in p2p we have to deal with mobile entities and users that change locations or even leave and re-enter the system frequently. Since a server in the client-server model is known by its address this model is based on IP addressing in contrast with p2p where other approaches are feasible as well. The same reason applies for the need of a DNS-based naming in the first model while p2p is more flexible and requires only a naming scheme understandable by the participating peers. Communications in a client-server model are based on remote procedure calls since we know the endpoint of the server and the service we wish to invoke. In p2p where reliability poses a strong issue asynchronous communications are a more suitable approach. Finally, a client-server model exists while the server is alive, thus it is server dependent whereas in p2p each peer has its own lifecycle and thus the services and data it provides.

3. What is a hierarchical and what is a flat client-server model?

The client-server model can be flat where all clients only communicate with a single server (possibly replicated for improved reliability), or it can be hierarchical for improved scalability. In a hierarchal model, the servers of one level are acting as clients to higher level servers. Examples of a flat model include traditional middleware solutions, such as object request brokers and distributed objects. Examples of a hierarchical model include DNS server and mounted file systems.

4. What is a super peer?

Between a pure peer –to-peer system, where no centralized server exists and a hybrid where servers are used to store meta-data about other peers there is an intermediate solution based on SuperPeers, such as KaZaa. SuperPeers contain some of the information that others may not have. Other peers typically lookup information at SuperPeers if they cannot find it otherwise. They are actually peers with some additional capabilities.

5. What is the difference between a compute-intensive and a componentized application? How does this relate to vertical and horizontal distribution?

The general idea behind compute-intensive applications is that idle cycles from any computer connected to the Internet can be leverage to solve difficult problems that require extreme amounts of computation. The same task is performed on each peer using different sets of parameters. Componentized applications, although yet not very recognized, can be envisioned as applications that can be built out of finer-grain components that execute over many nodes in parallel. In contrast to compute-intensive applications that run the same task on many peers, componentized applications run different components on each peer.

6. What is according to the authors the main challenge of communication in p2p?

The fundamental challenge of communication in a P2P community is overcoming the problems associated with the dynamic nature of peers. Either intentionally (e.g., because a user turns off her computer) or unintentionally (e.g., due to a, possibly dial-up, network link failing) peer groups frequently change. Maintaining application-level connectivity in such an environment with frequent disconnections is one of the biggest challenges in P2P systems.

7. What is the most common solution to reliability across p2p systems?

Reliability in P2P systems is a hard problem. The inherently distributed nature of peer networks makes it difficult to guarantee reliable behavior. The most common solution to reliability across P2P systems is to take advantage of redundancy. For example, in case of compute-intensive applications upon a detection of a failure the task can be restarted on other available machines. Alternatively, the same task can be initially assigned to multiple peers. In file sharing applications, data can be replicated across many peers. Finally, in messaging applications, lost messages can be resent or can be sent along multiple paths simultaneously.

8. What are the advantages/disadvantages of the centralized directory, the flooded requests, and the document routing models.

The advantages of a centralized directory is that the search is fast, the directory knows all the peers and can also find the “best” peer offering some data based on its load, network latencies and user preferences. However it introduces scalability issues. Since the directory has information

about all the users it need much space when the users increase, and also since it receives all the requests it becomes a bottleneck and its performance deteriorates.

The flooded requests are completely decentralized. No managing or advertising is required in contrast with the centralized approach. However the model requires a lot of network bandwidth, and hence does not prove to be very scalable, but it is efficient in limited communities such as a company network. To solve the problem super-peers have been used, but this lead to an increase in computations by the CPU.

The last model that uses IDs for documents and peers is very efficient for large, global communities, it has the problem that the document IDs must be known before posting a request for a given document. Hence it is more difficult to implement a search than in the flooded requests model. Also, network partitioning can lead to an *islanding* problem, where the community splits into independent sub-communities, that don't have links to each other.

9. In the centralized directory approach, after the best peer is located, the file exchange occurs directly between it and the requesting peer.
What are the advantages/disadvantages of this?

The advantage is that the server is no longer involved in their communication and thus is relieved from the load. The peers can communicate directly without its co-ordination without any problems.

10. What can be considered as a closure mechanism in Gnutella?

In fully decentralized file systems, such as Gnutella, just finding the network becomes difficult. The closure mechanism for a new node that wants to join Gnutella, is either knowing the address of another Gnutella node or use a host list with known IP addresses of other peers. The node joins the network of peers by establishing a connection with at least one peer currently in the network. Then, it can begin discovering other peers and cache their IP addresses locally. When a user wishes to find a file, the user issues a query for the file to the Gnutella users about which it knows. Those users may or may not respond with results, and will forward the query request to any other Gnutella nodes they know about.

11. What are the factors that affect scalability, give one example for each.

Scalability is limited by factors such as the amount of centralized operations (e.g., synchronization and coordination) that needs to be performed, these require many messages to be exchanged between the peers thus creating excessive network traffic, for example, if we have a distributed transaction that need to commit. The amount of state that needs to be maintained, for example, logs for use while network or other failures occur. The inherent parallelism an application exhibits for example an interactive application cannot be paralleled since it requires input maybe from a single user. Finally, the programming model that is used to represent the computation which may for example not use threads or procedures that can run in different machines.

12. Given the ad-hoc nature of connectivity in p2p, comment on what type of (message-oriented) communication (i.e., synchronous/asynchronous, transient/persistent) would be more appropriate.

Since peers in a p2p join and leave dynamically and there are no guarantees for reliability and availability, non-blocking asynchronous communications should be used. Otherwise, we would experience great delays and nodes would be blocked without knowing if their connections is too slow or there is a failure between them or in one of them. Also, because of these reliability and availability issues persistent primitives (with the messages stored at the sender buffer) should be used only if the application requirements are more demanding in terms of reliability. Otherwise, transient communications seems the natural choice for such an environment.

13. pg 17, 1st column, last par "The geographical distribution of the peers help to reduce congestion on both peers and the network". Explain.

Since peers are geographically dispersed, there is no part of the network that becomes overloaded with request and no peer receives many requests at the same time. Since replicas are also distributed there is a big chance that a node will find the file it wants close to it and will not have to travel far and since there is no single owner of a file we will not have bottlenecks.

14. What is the goal of caching in p2p? What are the advantages/disadvantages of caching the reply at all nodes in the return path? Can you think of any alternatives? Is this possible in Gnutella?

The goal of caching is to minimize peer access latencies, to maximize query throughput and to balance the workload in the system. The object replicas can be used for load balancing and latency reduction. The advantage is an increase in performance if another query for the same recourse arrives at any of the peers in the route. The disadvantage is the space overhead that this caching imposes in every node. Maybe some update scheme can be used so that only frequent queries are cached. Since Gnutella uses a flooded request model and the answer backtracks from the route it was originated this caching is possible.

15. What does the "power-law distribution of the p2p network" (pg 17) mean?

A Power-law distribution is a concentration of links within a few hubs, a scale-free network. This is in contrast to a random network where each node has the same scale of links, resulting in a bell-curve distribution. Power-law is driven by "preferential attachment." When a new node enters the network it prefers to link to more connected hubs. Over time, hubs grow faster, with only more random exceptions.

16. Compare/relate the definition of distributed systems in sec 5.2 (pg 21) with sec 1.4 of the textbook.

“A computing grid can be seen and used as a single, transparent computer. A user logs in, starts jobs, moves files, and receives results in a standard way.” This definition is similar that of a distributed operating system that completely hides the underlying machines and the multiple users and provides the user with a view of a single-user single processor

machine. The difference is that DOSs are used either with multi-processors or homogeneous computers while in a distributed system although we have the same goal we have to deal with lots of heterogeneous systems. This definition in sec 5.2 focuses on transparency and this is what relates it to a DOS and makes it different from the corresponding definition of a NOS, which however deals with heterogeneity. The middleware solutions developed only prove how important the role of transparency is in a distributed system.

17. Why is the fault tolerance problem a greater challenge in collaborative p2p systems than in file sharing p2p systems?

Fault tolerance is another challenge for collaborative systems. In shared applications, messages often must be delivered reliably to ensure that all peers have the same view of the information. In some cases, message ordering may be important. For file sharing systems no guarantees for the delivery time or order of the messages are necessary since we are only interested in finally receiving a consist copy of a file. So with looser demands reliability is much easier to be obtained. In most cases, we are mostly concerned with availability, which can be achieved through replication.