CS 590R: Peer-to-Peer (P2P) Network

by Chun-Kong Cheng (ckcheng@cs.purdue.edu)

1 What is a Peer-to-Peer (P2P) Network?

A P2P network is characterized by the following:

1. A P2P network is a distributed network of computers; in contrast to the server-client model, a node in the P2P network has no distinction between a server and a client. In fact, each P2P node acts as a server as well as a client. It is equipped with a software called *servent* (*ser*ver-cli*ent*) to participate in the P2P protocol.

2. It is a dynamic network: nodes (peers) and edges (currently established connections) appear and disappear over time. In contrast to a static network, a P2P network has no fixed infrastructure with respect to the participating peers, although the underlying network can be static (e.g., Internet). A P2P network can be considered as an *overlay network* over an underlying network, where the communication between adjacent peers in the P2P network may in fact go through one or more intermediate peers in the underlying network.

3. Peers communicate using only local information, in order to minimize the traffic injected to the network.

4. An incoming node does not have global knowledge of the current topology, or even the identities (e.g., IP addresses) of other peers in the current network.

One obvious advantages of a P2P network is that decentralized computing can be achieved – not depending on one particular machine as in the client-server model. This can be important to applications that share data and resources. An example is searching, where we can store the indexes in peers

instead of in one single computer; the peers can share the search query loads, and the indexes can be kept fresh compared with keeping one single index in one computer. The trade-off to this approach is the dramatical increase in network traffic; so, the issue is how a P2P network can be constructed to maintain minimal traffic.

Real-life examples of P2P network include Gnutella, Freenet and `www.kazaa.com`. We examine Gnutella in more detail in the next section.

1.1 Case Study: Gnutella

In Gnutella, a new node can **join** the P2P network by contacting a (central) host server to get entry-points to the network. To perform a **search query**, a node takes the following steps:

- It sends query to its neighbors.
- They in turn forward it to their neighbors. To prevent too much flooding, the neighbors decrement Time to Live (TTL) for the query. The query dies when TTL = 0.
- Search results are sent back along requested path to the requesting node.

Note that in Gnutella, it does not mention which neighbors a servent joining the network should connect to. Also, there is no standard on what a node should do when any of its neighbor drops out of the network.

1.2 Building P2P Networks

The two critical properties for a good P2P network are:

- Connectivity: maintaining (even) connectivity under a dynamic setting is a non-trivial issue.
- Low-diameter: small diameter to reduce the packet transmission time.

In current real-life systems (e.g., Gnutella), the network is generated in an ad-hoc approach. This can result in partitioning of the network into disconnected pieces. It may also possess large diameter. Thus, the challenge is to design distributed protocols which operate with only local knowledge.

In the next section, we discuss a distributed protocol to build P2P networks with the following properties:

1. reasonable connectivity;
2. logarithmic diameter;
3. constant degree;
4. low overhead;
5. operates with no global knowledge; and
6. can be easily implemented with local message passing.

2 A P2P Protocol

Let us now study the protocol introduced by Pandurangan et. al [1].

2.1 Protocol Basics

The protocol is composed of a set of rules that are applicable to the following situations a node may find itself in:

1. How to join the network ?
2. What happens if a neighbor drops out ?
3. How to maintain a bounded number of connections ?

Similar to Gnutella, the protocol assigns a central host server which has the following properties:

1. It acts as a gateway mechanism to enter the network.
2. It maintains a **cache**, which is essentially a list of K nodes (i.e., their IP addresses) at all times. Since K is a constant, implies the cache *does not* keep the information of all nodes.
3. It is reachable by all nodes at all times.

4. It does not need to know the network topology or the identities of all the nodes in the network.

Some other definitions for the protocol are:

1. When a node is in the cache, we refer to it as a *cache node*. It accepts connections from all other nodes.

2. A node is *new* when it joins the network, otherwise it is *old*.

3. The protocol ensures that the degree (number of neighbors) of all nodes will be in the interval $[D, C+1]$, for two constants D and C.

4. A **c-node** is a node that was a cache node at some time.

5. A **d-node** is a node that is not a c-node.

2.2 Protocol for Node v

Here we present the protocol for joining and reconnecting to the network, for a node v:

1. **On joining the network:** Connect to D cache nodes, which are chosen uniformly at random from the current cache. Note that $D < K$.

2. **Reconnect rule:** If a neighbor of v leaves the network, and that connection was not a preferred connection, connect to a random node in the cache with probability $D/d(v)$, where $d(v)$ is the degree of v before losing the neighbor.

3. **Cache Replacement rule:** When a cache node v reaches degree C while in the cache (or if v drops out of the network), it is replaced in the cache by a d-node from the network. Let $r_0(v) = v$, and let $r_k(v)$ be the node replaced by $r_{k-1}(v)$ in the cache. The replacement d-node is found by the following rule:

   $k = 0$;
   while (a d-node is not found) **do**
   search neighbors of $r_k(v)$ for a d-node;
   $k = k + 1$;
   endwhile

4. **Preferred Node rule:** When v leaves the cache as a c-node it maintains a *preferred connection* to the d-node that replaced it in the cache. If v is not already connected to that node this adds another connection to v. Thus the maximum degree of a node is $C+1$. Also, a c-node can follow a chain of preferred connection to reach a cache node.

5. **Preferred Reconnect rule:** If v is a c-node and its preferred connection is lost, then v reconnects to a random node in the cache and this becomes its new preferred connection.

Finally, note that the degree of a d-node is always D. Moreover, every d-node connects to a c-node. A c-node may lose connections after it leaves the cache, but its degree is always at least D. A c-node has always one *preferred* connection to another c-node.

3 Protocol Analysis

The protocol assumes the following model:

1. The arrival of new nodes is Poisson distributed with rate λ.

2. The duration of time a node stays connected to the network is independently and exponentially distributed with parameter μ.

3. Let G_t be the network at time t. We are interested in the evolution in time of the stochastic process $\mathcal{G} = (G_t)_{t \geq 0}$.

4. The evolution of the size of $\mathcal{G}$ depends only on the ratio $\lambda/\mu = N$, the steady state size of the network.

Using the above assumption, the dynamic nature of a P2P network can be captured by a M/M/∞ model. Also, it is assumed that λ is much larger than μ. The objective is to prove that by using the proposed protocol, the network is connected at most of the time i.e., good connectivity. But we first need to derive some facts for the network size.

3.1 Network Size

W.l.o.g let $\lambda = 1$; thus $\mu = 1/N$. Let $G_t = (V_t, E_t)$ be the network at time t.

Theorem 3.1 *1. For any $t = \Omega(N)$, w.h.p. $|V_t| = \Theta(N)$.*

2. If $\frac{t}{N} \to \infty$ then w.h.p. $|V_t| = N \pm o(N)$.

Proof:

Consider a node that arrived at time $\tau \leq t$. Since its arrival is Poisson, its probability of still being in the network at time t is $e^{-(t-\tau)/N}$. Let $p(t)$ be the probability that a random node that arrives any time during the time interval $[0, t]$ is still in the network at time t. Since in a Poisson process the arrival time of a random element is uniformly distributed in $[0, t]$, we have

$$p(t) = \frac{1}{t}\int_0^t e^{-(t-\tau)/N} d\tau = \frac{1}{t} N(1 - e^{-t/N}).$$

For $t = \Omega(N)$, according to the M/M/∞ model, the number of nodes in the graph at time t has a Poisson distribution with expectation $tp(t)$. Thus, $E[|V_t|] = \Theta(N)$. Also, when $t/N \to \infty$, $E[|V_t|] = N - o(N)$.

Using the Chernoff bound, for $t = \Omega(N)$,

$$Pr\left(||V_t| - E[|V_t|]| \leq \sqrt{bN \log N}\right) \geq 1 - 1/N^c$$

for some constants b and $c > 1$. Thus w.h.p. $E[|V_t|] = \Theta(N)$ for $t = \Omega(N)$. □

References

[1] G. Pandurangan, P.Raghavan, and E. Upfal. Building low-diameter P2P networks. In *IEEE FOCS*, 2001.