CptS 464/564
Lecture 29

30th April 2003

Housekeeping

- There are no lecture 28 notes because that was the video lecture by Dr. Butler Lampson on the Global Name Service
- The final exam is next Monday, May 5, 3:10-5:10 PM in the regular classrooms. Part of the class period on Thursday will be devoted to review. The final exam will be comprehensive but emphasize material covered since the mid-term exam.

Continuing with overlay networks

Last Tuesday I talked about overlay networks: how they worked and what kind of performance could be expected.

Today I want to present some recent results suggesting the practical issues of scale in practical large overlay networks and begin to look at security considerations.

Gnutella

Gnutella is a peer-to-peer file sharing service based on an overlay network. Gnutella doesn't have the distributed hash table capability that we talked about last time – instead files are found by "flooding" a request through the overlay network: a less efficient approach but one that gets the job done.

Recently, researchers at the Univ. of Washington conducted a study to determine some of the practical characteristics of gnutella's overlay network. They looked at connectivity characteristics of the nodes, uptime of the nodes, and whether nodes were primarily suppliers or consumers of files. (They also looked at similar characteristics for Napster – see the paper if interested.) The paper is

- S. Saroiu, P.K. Gummadi, S. Gribble. *A Measurement Study of Peer-to-Peer File Sharing Systems*, Technical Report #UW-CSE-01-06-02, Dept. of Computer Science & Engineering, University of Washington, 2001.

I think the results are quite interesting, both for what they say about the current state of people's internet connectivity and for what they suggest about the future of "voluntary" overlay networks .

Gnutella's overlay network is constructed by sending (gnutella) ping messages to an exanding ring of hosts. The response messages (pong) let a node know of other nodes in its neighborhood, and it will add direct connections to some of them.

"Crawling" a gnutella network: send ping messages with large TTL through well-known, popular peers. Collect the hosts identified in the resulting pong messages. (The gnutell pong message contains the IP address of each peer, as well as the number and total size of files being shared.)

A two-minute run of the crawler would typically gather between 8000 and 10000 peers (in early May 2001). Over 8 days in excess of 1.2 million unique peers were identified.

Measured characteristics

- Latency: due to the way the TCP algorithm works, large latency can have a significant effect on achievable bandwidth.
- Bottleneck bandwidth, upstream and downstream: for sharing large files the bottleneck bandwidth is important in the observed performance. Clearly, both the bandwidth and latency depend on where and when they are measured. But distributions observed in a study like this one are likely to be typical, even if the measurements for particular hosts might be quite different if measured elsewhere.
- Lifetime: what fraction of the time are hosts up and available?
- Number of shared files

Results are presented as graphs of cumulative distribution functions. Recall that a CDF is a function representing the fraction of N measurements that are no more than a certain amount:

$$f(x) = \frac{|\{y|y \leq x\}|}{N}$$

- Bottleneck bandwidths: 35% have upstream bandwidth at least 1Mbps, 8% have upstream BW at least 10Mbps, but 22% have upstream bandwidth of 100Kbps or less. Implication: a large fraction of the peers are easily swamped by a small number of connections and are unsuitable as servers. The graph also shows that many hosts have less upstream than downstream bandwidth (typical of ADSL connections, which are marketed as they are in part to deter hanging content servers off of them). The flooding queries used in gnutella are a significant deterrent to use of modem-based connections.
- Latencies are dispersed over 4-5 decimal orders of magnitude. 20% of hosts have latencies at most 70ms, 20% have latency at least 280 ms. Recall how a gnutella

network forms: hosts choose to peer randomly with existing hosts without regard for "closeness". This paper doesn't suggest what the effect of this choice is on query response times for gnutella. But the wide range of latency suggests it is a factor that should be considered in distributed system design at the scale of the Internet.

- The correlation between (downstream) bandwidth and latency is also worth noting. Observe the clusters for modem connections and broadband connections. Within the broadband connections observe the horizontal bands conjectured to correspond to increasing geographic distance.
- Availability: the best 20% of gnutella peers are available 45% of the time or more. The best 10% (interpolating the graph) are available 75% of the time or more. What do we conclude about who carries the burden in a voluntary peer-to-peer network?
- Number of files shared: 25% of peers share no files at all and only 7% share more than 1000 files. These 7% share more than half the files available in the system: a large percentage of "clients" have to rely on a small percentage of "servers".

Robustness of the overlay network: theoretical studies of so-called "scale-free" networks along with confirmation that the existing gnutella overlay structure approximates a scale-free network with particular parameters suggest that the gnutella overlay is robust in the face of loss of a random 30% of the nodes. However, loss of the best-connected 4% of the nodes shatters the connectivity. (Similar results have recently been achieved for the Internet itself).

Conclusion: large peer-to-peer networks may, in fact, be quite heterogeneous with important characteristics varying over 5 orders of magnitude. In designing a peer-to-peer system, the designer would do well to keep this in mind: if high-reliability, high-performance nodes can be identified advantage can possibly be taken to improve the overall system performance. Conversely, failure to acknowledge these differences may be the downfall of a system.

A few remarks on the paper: this paper is a product of a strong *measurement culture* in systems research at UW. Note the considerable care in describing how the measurements were made, completeness of presentation of the data, and its interpretation.

Security in overlay networks

The following discussion is based on

- M. Castro, P. Druschel, A. Ganesh, A. Rowstron and D. Wallach, *Secure routing for structured peer-to-peer overlay networks*, Fifth ACM Symposium on Operating System Design and Implementation (OSDI '02), December 2002.

Let us review: peer-to-peer overlays appear to offer attractive properties of efficiency, robustness and ease of management for networks containing thousands to millions of nodes. We would like to assess the risks associated with use of these networks, which requires understanding *vulnerabilities* and *threats.* This discussion may also be of interest when considering *ad-hoc* mobile networks where end-user-controlled network nodes have to take on networking roles such as routing usually handled by infrastructure in wired networks.

Previously, we've noted that overlay networks such as Pastry and gnutella are resilient in the face of random failure but are susceptible to coordinated attacks that take out key nodes. What other kinds of coordinated attacks are such networks susceptible to by nature of their architecture? (inherent vulnerabilities) How can these vulnerabilities be addressed?

This paper considers vulnerabilities in the routing component of Pastry-based overlay networks.

In my experience any system involving the participation of even hundreds of individuals is susceptible to the nature of large human populations to contain at least some bad apples: thieves, vandals, and worse. These large peer-to-peer systems are no different, so when we put one together we should expect that to a greater or lesser degree there will be some intentionally faulty behavior by some of the nodes.

Thus we have a vulnerability to allowing bad-guys into the system in the first place. We sense that having fewer bad guys is better, so having recognized the inevitability of bad guys, we should try to limit their number and the damage that they can do.

System Model

Our model system consists of N nodes running an overlay peer-to-peer network such as Pastry. We that the fraction of faulty nodes is bounded by f ($0 \leq f \leq 1$). Faulty nodes may behave arbitrarily (Byzantine failures), though we model *constrained collusion:* no more than cN faulty nodes may cooperate ($1/N \leq c \leq f$).

Messages sent over from a correct node to a correct node over an overlay path containing only correct nodes are delivered within time D with probability P_D. Note how already we're only dealing with a probabilistic notion of correctness, even in the correct parts of the system. How can we get similar guarantees when there incorporate faulty nodes?

NodeID assignment vulnerability

Pastry's delivery properties rely on random assignment of node IDs (not correlated with anything). What can happen if an attacker obtains control of the node ID assignments so that he can choose his own node IDs?

Suppose an attacker could choose a nodeID closest to a desirable data item: then it would essentially *own* control of that item. A similar argument holds even if the item is replicated – an attacker could own all the replicas.

Attackers could also choose nodeIDs that would place them on all paths from a particular node and be able to control communication with that node, or choose nodeIDs that would give them preferential placement in the overlay routing tables, giving them access to a large fraction of the traffic.

To overcome this vulnerability, the paper suggests issuing certified nodeIDs from a central certificate authority. The CA makes sure that IDs are indeed random and are not forged (by signing the certificates). A certificate essentially gives a node with a particular IP address the right to join the overlay. (Note if nodes change IP address frequently, and IP addresses are re-used frequently, the vulnerability is not entirely eliminated – by swapping certificates and IP addresses an attacker may be able to optimize his ability to take over routing table entries).

The other problem with node IDs is that an attacker may acquire a lot of them and then choose to use only the ones that further his purpose. Suggested solutions including requiring payment for certificates, thus making a coordinated attack expensive) and limiting the number of certificates granted to any individual (a method which assumes that the CA can identify the requester. That is, of course, hard when dealing with overlay networks of thousands or millions of nodes).

Another approach might be to require nodes to demonstrate that they have computed something that takes a long time to compute in order to get a certificate. Thus although an attacker doesn't have to pay anyone for certificates, he does have to consume something valuable. [This reminds me cultures where a person's wealth is demonstrated by destroying objects of great value.]

It's also worth noting that large overlay networks require attackers to control of more IDs than small ones, so they may be significantly more expensive to attack.

Routing table maintenance vulnerabilities

Illustrate with one example: nodes periodically update their routing tables with information from their neighbors. Thus faulty nodes can feed bogus routing table data to correct nodes. Assuming that correct nodes start with RTEs that are correct with probability $(1-f)$ and they then receive a batch of updates, $(1-f)$ of them from correct nodes and f of them from faulty nodes. After the update the expectation of correctness for an RTE is $(1-f)\times f + f \times 1$. ($f$ of the $1-f$ updates from correct nodes are faulty, and all of the f updates from faulty nodes are faulty. Thus, after 1 round the routing tables are more corrupt than they were before.

The technique for overcoming this vulnerability involves constraining the routing table entries so that the receiving node is less susceptible to faulty entries. The details are beyond this lecture, but the idea is that we can keep a second routing table which will be more correct than the Pastry's optimized routing table because the values in it are

constrained in a particular way. The constraint makes it harder for an attacker to fool the node into accepting an incorrect entry. If, when sending a message, we determine that the message did not reach the correct destination(s) with high probability, we will resend the message using the slower but more-likely-to-be-correct second routing table.

Message forwarding vulnerability

Message delivery requires on average $log_{2^b}N$ hops in Pastry. If the number of hops required is h, and correct delivery requires a correct routing table entry at each hop then the probability of correct delivery is $(1-f)^h$. Unfortunately this probability decreases rapidly with both f and h. 10% faulty nodes in a Pastry network of 100,000 nodes reduces the probability of successful delivery to 65%.

The proposed way to address this vulnerability is to send messages over multiple, diverse routes. A test is proposed that with high probability will indicate that a message was not routed correctly. Nodes first attempt to send messages over the efficient overlay (as described last time for Pastry). If failure is detected, messages are resent in a way that causes them to traverse multiple, diverse paths. Again, the details are beyond the scope of this lecture.

Under conditions similar to those cited before, messages are delivered successfully 99.9% of the time for $f < 0.3$.