Why Gnutella Can’t Scale. No, Really.

by Jordan Ritter <jpr5@darkridge.com>

Please note that this paper was first released in February of 2001.

Forward

In the spring of 2000, when Gnutella was a hot topic on everyone’s mind, a concerned few of us in the open-source community just sat back and shook our heads. Something just wasn’t right. Any competent network engineer that observed a running gnutella application would tell you, through simple empirical observation alone, that the application was an incredible burden on modern networks and would probably never scale. I myself was just stupefied at the gross abuse of my limited bandwidth, and that was just DSL -- god help the dialup folks! We wondered to ourselves, Is no one paying attention, was no one bothered?

That summer we all saw a rush of press on Gnutella, and the rumour mill started churning. Most stories covering Gnutella were grossly and inappropriately evangelistic, praising the not-yet-analyzed Gnutella as a technology capable of delivering on wildly fantastic promises of fully distributed, undeterrable, unstoppable, larger-than-life file sharing on the grandest scale. Many folks were convinced that Gnutella was the *next generation Napster*. Gene Kan, the first to spearhead the Gnutella evangelistic movement, claimed in one early interview: "Gnutella is going to kick Napster in the pants." Later Kan admitted "Gnutella isn’t perfect", but still went on to say that "there’s no huge glaring thing missing". Well, something just wasn’t right, and though we couldn’t see it, it did seem pretty glaring.

We all understood the excitement. Herein was a technology that could potentially prove the true magnitude of Metcalfe’s Law. That realization evoked nothing short of the phrase *"holy shit!"*. But what I couldn’t understand was why no one was questioning the legitimacy of these claims. For several months the only analyses anyone heard of practical implementations were generalizations and speculative comments, without much scientific or mathematical basis.

So I quickly got fed up, and resolved to write a research paper. Sometime in late March, I had begun analyzing the network structure of the Gnutella system, trying to find a way to gauge the capacity of a GnutellaNet in generalized terms, and to predict its realistic limits. What later resulted was a set of mathematical equations that could describe reachability, capacity, and bandwidth throughput. I then fed those equations into Mathematica to produce 3-D plots depicting, much to my own satisfaction, visual realizations of *exactly what didn’t make sense*.

At about the same time, a fellow colleague in the security industry wrote a short paper detailing the various and flagrant insecurities inherent in this particular implementation of a distributed system. Seth McGann’s security advisory titled **Self-Replication Using Gnutella** centered on the characteristics an Internet Worm inside a GnutellaNet could thrive from, and also touched on a few other flaws that would be useful to an attacker. His advisory posted in May of 2000, and unfortunately went mostly unnoticed (or misunderstood, because of its technical nature).

Later in August, Xerox PARC published a **research paper** on the *social* characteristics of a GnutellaNet, proving through empirical observation that transience hurts this type of fully distributed network considerably, and that Gnutella was not such an invincible proposition after all.

These days the Internet doesn't lack for useful papers on Gnutella. **Research papers** by the folks at **Distributed Search Solutions** are fairly high in quality and remain objective, if not optimistic about the future of Gnutella. Other informative articles persist on **O'Reilly's P2P Website**, and elsewhere.

So where's my paper, and why haven't you seen it? Well, in case you didn't know, I'm one of the founding developers of **Napster**, and for several good reasons, including the sobering fact that I was one of the leaders of the main competitor, I did not release my material to the public. Several times I resigned myself to re-writing my paper to accommodate the release of new information and analyses, but I never finished. Now I regret having sat on this for so long, for every paper on Gnutella that has come out in the last year has served as nothing but vindication of my conclusion from so early on: *Gnutella will never scale*.

Following is what remains of my paper, hacked up, sliced, diced and re-written. The information and analyses are still useful, but as I just said, the conclusions are the same. This paper simply proves those conclusions through mathematics.

Onward, Through the Fog

This paper assumes a working knowledge of Gnutella networks and internals, and therefore uses terminology and phraseage specific to Gnutella. If the wording seems somewhat strange or foreign to you, please stop reading this paper and seek other documentation before proceeding. Furthermore, explanation of the accompanying math is intentionally terse. Every effort has been made to verify the accuracy of the equations herein, but this discussion is intentionally limited to that which is solely relevant to Gnutella in order to keep at a minimum any distraction from an already complex topic.

To Scale, or Not to Scale

Scaling Gnutella will require more than just better resource management tools -- in its current incarnation Gnutella is mathematically and technologically unable to scale to a network of any reasonably large size. Following herein is a discussion focused on mathematically describing the metrics of a GnutellaNet topology, and using derived equations to interpret and visualize realistic limits of the technology. In order to keep the math as simple as possible, let's assume we're examining a relatively quiet GnutellaNet network, and dissect the flow of information one step at a time.

Variables and Equations

P	The number of users connected to the GnutellaNet.
N	The number of connections held open to other servents in the network. In the default configuration of the original Gnutella client, this is 4.

T	Our TTL, or Time To Live, on packets. TTL's are used to age a packet and ensure that it is relayed a finite number of times before being discarded.
B	The amount of available bandwidth, or alternatively, the maximum capacity of the network transport.
f(n, x, y)	A function describing the maximum number of reachable users that are at least ***x*** hops away, but no more than ***y*** hops away. ***f(n, x, y) = Sum[((n-1)^(t-1))*n, t = x->y]***
g(n, t)	A function describing the maximum number of reachable users for any given ***n*** and ***t***. ***g(n, t) = f(n, 1, t)***
h(n, t, s)	A function describing the maximum amount of bandwidth *generated* by relaying a transmission of ***s*** bytes given any ***n*** and ***t***. *Generation* is defined as the formulation and outbound delivery of data. ***h(n, t, s) = n*s + f(n, 1, t-1)*(n-1)*s***
i(n, t, s)	A function describing the maximum amount of bandwidth *incurred* by relaying transmission of ***s*** bytes given any ***n*** and ***t***. *Incurrence* is defined as the reception or transmission of data across a unique connection to a network. ***i(n, t, s) = (1 + f(n, 1, t-1))*n*s + f(n, t, t)*s***

It benefits the casual reader to first explain in terms of a balanced, equally distributed GnutellaNet, so for this exercise assume that everyone has the same ***N*** and ***T***. In the initial release of Gnutella, ***N = 4*** and ***T = 5***. Further, let ***P = 2000***, arbitrarily. Finally, let us assume no other interfering factors exist (for now).

Early reports of Gnutella's usage claimed upwards of 2000 to 4000 users on the GnutellaNet. This is significant because these reports inaccurately implied that all 4,000 users on the GnutellaNet were reachable and searchable. The reality is that even in an ideally balanced GnutellaNet, ***P*** is never relevant to your *potential* reach; ***N*** and ***T*** are the only limiting factors.

Reachable Users

	T=1	***T=2***	***T=3***	***T=4***	***T=5***	***T=6***	***T=7***	***T=8***
N=2	2	4	6	8	10	12	14	16
N=3	3	9	21	45	93	189	381	765
N=4	4	16	52	160	484	1,456	4,372	**13,120**
N=5	5	25	105	425	1,705	6,825	**27,305**	**109,225**
N=6	6	36	186	936	4,686	**23,436**	**117,186**	**585,936**
N=7	7	49	301	1,813	**10,885**	**65,317**	**391,909**	**2,351,461**
N=8	8	64	456	3,200	**22,408**	**156,864**	**1,098,056**	**7,686,400**

Raising ***N*** (number of connections open) and ***T*** (number of hops) extend the number of reachable users geometrically.

Keep in mind, the above illustrates *potential* reach given two assumptions: the network is *fully balanced*, and everyone shares the *same **N** and **T***.

So, the next obvious step for an intrepid and now better-informed Gnutella user is to increase N and T, so as to extend their potential reach into the GnutellaNet web. Not so fast! As your reach increases geometrically, so does the amount of bandwidth generated and incurred. Let's now move the discussion towards ***B***.

Delving Deeper into B

Before proceeding, it is *very important* to understand that many assumptions must be made in order to carry out these computations. Observed characteristics of GnutellaNet topologies are simply too varying to accurately generalize. That said, I still believe that there exists a statistical mean of each characteristic in a GnutellaNet, which is to say that if I were to take a snapshot of the current topology of a public GnutellaNet I could derive an average ***N***, ***T***, and so forth. While potentially inaccurate as a realistic representation, these means can still produce a useful generalization.

In our discussion of ***B***, there are really two different perspectives on how to measure the amount of bandwidth: the amount *generated*, and the amount *incurred*. This is a very important distinction to make, because knowing the amount of raw data *generated* is statistically useful, but understanding the bandwidth cost *incurred* by individual events in the network is much more important since it more realistically signifies the impact on an Internet connection. As previously stated, ***h(n, t, s)*** represents the amount of bandwidth generated by relaying a packet through the network, counting only data that is outbound to another destination. ***i(n ,t, s)***, on the other hand, counts all outbound *and inbound* transmissions, yielding a more accurate perspective on bandwidth usage. Let's introduce an example.

Joe Smith likes classic rock, and is desperately searching for any live recordings of The Grateful Dead. Joe loads up his Gnutella client, connects to the GnutellaNet, and executes his search, "grateful dead live". What actually happens?

Search Query Packet Makeup

IP header	**20 bytes**
TCP header	**20 bytes**
Gnutella header	**23 bytes**
Minimum Speed	**1 byte**
Search string	**18 bytes + 1 byte** *(trailing null)*
Total:	**83 bytes**

It isn't useful to account for Data Link Layer transmissions since they vary widely and don't significantly impact these calculations, so they have been intentionally ommitted.
IP and TCP header calculations assume simplest case scenario.

Joe's search request results in an 83 byte data packet. Initially, everyone would agree that it looks like a tiny, unnoticeable amount of data. Let's take a look at the bandwidth cost of simply relaying the search

request. ***h(n, t, s)*** is comprised of the data Joe transmits across his connections to other Gnutella users (***n*s***), plus transmissions of all tiers between Joe and the last tier, which is only receiving.

Bandwidth Generated in Bytes (*S=83*)

	T=1	***T=2***	***T=3***	***T=4***	***T=5***	***T=6***	***T=7***	***T=8***
N=2	166	332	498	664	830	996	1,162	1,328
N=3	249	747	1,743	3,735	7,719	15,687	31,623	63,495
N=4	332	1,328	4,316	13,280	40,172	120,848	362,876	1,088,960
N=5	415	2,075	8,715	35,275	141,515	566,475	2,266,315	**9,065,675**
N=6	498	2,988	15,438	77,688	388,938	1,945,188	**9,726,438**	**48,632,688**
N=7	581	4,067	24,983	150,479	903,455	**5,421,311**	**32,528,447**	**195,171,263**
N=8	664	5,312	37,848	265,600	1,859,864	**13,019,712**	**91,138,648**	**637,971,200**

From above, given a concurrent demographic comparable to Napster (assuming equally balanced), searching for a simple 18 byte string "grateful dead live" unleashes 90 megabytes worth of data to be transmitted.

Even so, I don't consider ***h(n, t, s)*** to be the best measure. Let's now look at ***i(n, t, s)***, which is comprised of the originating transmission, 1 reception and N-1 transmission for tiers 1 through T-1, and 1 reception for the last tier.

Bandwidth Incurred in Bytes (*S=83*)

	T=1	***T=2***	***T=3***	***T=4***	***T=5***	***T=6***	***T=7***	***T=8***
N=2	332	664	996	1,328	1,660	1,992	2,324	2,656
N=3	498	1,494	3,486	7,470	15,438	31,374	63,246	126,990
N=4	664	2,656	8,632	26,560	80,344	241,696	725,752	2,177,920
N=5	830	4,150	17,430	70,550	283,030	1,132,950	4,532,630	**18,131,350**
N=6	996	5,976	30,876	155,376	777,876	3,890,376	**19,452,876**	**97,265,376**
N=7	1,162	8,134	49,966	300,958	1,806,910	**10,842,622**	**65,056,894**	**390,342,526**
N=8	1,328	10,624	75,696	531,200	3,719,728	**26,039,424**	**182,277,296**	**1,275,942,400**

i(n, t, s) has the unique property of representing double ***h(n, t, s)***.

From above, a whopping *1.2 gigabytes* of aggregate data could potentially cross everyone's networks, just to relay an 18 byte search query. This is of course where Gnutella suffers greatly from being fully distributed.

Also, let's not forget that there is no consideration of time in this set of calculations. In the average case, 1.2 gigabytes worth of data takes a *very long time* to generate and propagate through the Internet. However, even in more realistic cases, propagating a few megabytes worth of data through several hundred thousand nodes across the Internet still takes a considerable amount of time.

At this point, though, our exercise is still incomplete. What percentage of Gnutella clients share content? Of them, what percentage are likely to respond to Joe's query? And of those, what would be the mean number of responses, and their mean length?

The Anatomy of a Firestorm

This is where we'll begin to see generalizations diverging from reality. Still though, let's take a quick gander at what evangelists thought Gnutella would be capable of. For this, we'll need to introduce a few more variables and equations.

More Variables and Equations

a	Mean percentage of users who typically share content.
b	Mean percentage of users who typically have responses to search queries.
r	Mean number of search responses the typical respondent offers.
l	Mean length of search responses the typical respondent offers.
R	A function representing the *Response Factor*, a constant value that describes the product of the percentage of users responding and the amount of data generated by each user. ***R = (a*b) * (88 + r*(10 + l))***
j(n, T, R)	A function describing the amount of data generated in response to a search query by tier ***T***, given any ***n*** and *Response Factor* ***R***. ***j(n, T, R) = f(n, T, T) * R***
k(n, t, R)	A function decsribing the maximum amount of bandwidth *generated* in response to a search query, including relayed data, given any ***n*** and ***t*** and Response Factor ***R***. ***k(n, t, R) = Sum[j(n, T, R) * T, T = 1->t]***

Assuming that a mean exists for the characteristics of our measurement makes these calculations much simpler. That said, recall that I don't believe this assumption to be false; that at any given moment there does exist some measurable ***a***, ***b***, ***r*** and ***l***. Let's assume conservative estimates for now, and apply observed behaviour from other reports later.

The difficulty in gauging the sheer amount of data coming back to us stems from our inability to realistically discern where in the partial mesh of connections the data is coming from. By design, the only thing we will know about about the packets received is the (hopefully) unique message ID. If the message ID correlates to the message ID of one of our pending queries, the response is ours. Otherwise, the response is someone else's traffic, and if it correlates to an known ID in our routing table, it is simply passed along.

Search Response Packet Makeup

IP header	**20 bytes**
TCP header	**20 bytes**
Gnutella header	**23 bytes**
Number of hits	**1 byte**
Port	**1 byte**
IP Address	**4 bytes**
Speed	**3 bytes**
Result Set	***r* * (8 + *l* + *2*) bytes**
Servent Identifier	**16 bytes**
Total:	**88 + *r**(10 + *l*) bytes**

Let's take a look now at what the variation of ***N*** and ***T*** yields in terms of bandwidth costs. For our first case, let's choose some reasonable values: ***a = 30%***, ***b = 50%***, ***r = 5*** and ***l = 40***, or ***R = 50.7***.

Bandwidth Generated in Bytes (*R=50.7*)

	T=1	*T=2*	*T=3*	*T=4*	*T=5*	*T=6*	*T=7*	*T=8*
N=2								
N=3	152.1	760.5	2,585.7	7,452.9	19,620.9	48,824.1	116,965	272,715
N=4	202.8	1,419.6	6,895.2	28,797.6	110,932	496,614	1,441,500	4,989,690
N=5	253.5	2,281.5	14,449.5	79,345.5	403,826	1,961,330	**9,229,680**	**42,456,400**
N=6	304.2	3,346.2	26,161.2	178,261	1,128,890	**6,832,640**	**40,104,500**	**230,230,000**
N=7	354.9	4,613.7	42,942.9	349,577	2,649,330	**19,207,500**	**135,115,000**	**929,909,000**
N=8	405.6	6,084	65,707.2	622,190	**5,491,420**	**46,392,900**	**380,422,000**	**3,052,650,000**

Precision is limited to 6 or less digits; sorry, I don't know how to make mathematica behave differently in this case.

With 30% of Gnutella users sharing, and only half of them responding, the standard client settings yield over 14MB of return responses. I believe this particular ***R*** value to be near reality as far as percentages are concerned, but ***r*** and ***l*** are probably conservative, given recent reports by Clip2 DSS and others. Let's raise ***R*** a bit, here's ***R = 72***.

Bandwidth Generated in Bytes (*R=72*)

	T=1	*T=2*	*T=3*	*T=4*	*T=5*	*T=6*	*T=7*	*T=8*
N=2								
N=3	216	1,080	3,672	10,584	27,864	69,336	166,104	387,288
N=4	288	2,016	9,792	40,896	157,536	577,440	2,047,104	**7,085,952**
N=5	360	3,240	20,520	112,680	573,480	2,785,320	**13,107,240**	**60,293,160**

N=6	432	4,752	37,152	253,152	1,603,152	**9,703,152**	**56,953,152**	**326,953,152**
N=7	504	6,552	60,984	496,440	3,762,360	**27,276,984**	**191,879,352**	**1,320,581,304**
N=8	576	8,640	93,312	883,584	**7,798,464**	**65,883,456**	**540,244,224**	**4,335,130,368**

These different values don't appear to have much of an impact on the overall bottom line; just over 13MB of traffic generated in response with standard client settings. Let's take one more look and adjust some of the values: ***a = 30%***, ***b = 40%***, ***r = 10*** and ***l = 60***, or ***R = 94.56***. I believe this ***R*** to be the most realistic.

Bandwidth Generated in Bytes (*R=94.56*)

	T=1	***T=2***	***T=3***	***T=4***	***T=5***	***T=6***	***T=7***	***T=8***
N=2								
N=3	283.68	1,418.4	4,822.56	13,900.3	36,594.7	91,061.3	218,150	508.638
N=4	378.24	2,647.68	12,860.2	53,710.1	206,897	758,371	2,688,530	**9,306,220**
N=5	472.8	4,255.2	26,949.6	147,986	753,170	3,658,050	**17,214,200**	**79,185,000**
N=6	567.36	6,240.96	48,793	332,473	2,105,470	**12,743,500**	**74,798,500**	**429,398,000**
N=7	661.92	8,604.96	80,092.3	651,991	4,941,123	**35,823,800**	**252,002,000**	**1,734,360,000**
N=8	756.48	11,347.2	122,550	1,160,440	**10,242,000**	**86,526,900**	**709,521,000**	**5,693,470,000**

Standard client settings yield a whopping 17MB generated in response to Joe's search query.

In order to better understand the results above, one must understand the *Response Factor*, ***R***, and the reasoning behind it. Recent analyses of Gnutella networks show a small percentage of participants actually sharing content, and a disproportionately small percentage of those sharing actually having most of the content. It is highly improbable that a means to statistically describe the widely varying response characteristics of participants in a GnutellaNet exists. ***R*** is a compromise for this difficult task, representing a gross mean across an ideal GnutellaNet of responses we can expect the average query to generate. The key word here is *ideal*; we know these gross means to exist, but they are as yet unmeasurable, or at least at this point unverifiable, given the quickly changing network topology.

Bringing it all together

So, now that we have all the pieces to the puzzle, let's fit them together. How much aggregate data, including request and response, is generated by Joe's search for "grateful dead live"? Let's intersect ***h(n, t, s)*** with ***k(n, t, R)*** to get The Big Picture.

Bandwidth Generated in Bytes (*S=83, R=94.56*)

	T=1	***T=2***	***T=3***	***T=4***	***T=5***	***T=6***	***T=7***	***T=8***
N=2								
N=3	532.68	2,165.4	6,565.56	17,635.3	44,313.7	106,748	249,773	572,133
N=4	710.24	3,975.68	17,176.2	66,990.1	247,069	879,219	3,051,410	**10,395,200**
N=5	887.8	6330.2	35,664.6	183,261	894,685	4,224,530	**19,480,500**	**88,250,700**

N=6	1,065.36	9,228.96	64,231	410,161	2,494,410	**14,688,700**	**84,524,900**	**478,031,000**
N=7	1,242.92	12,672	105,075	802,470	**5,844,690**	**31,245,100**	**284,530,000**	**1,929,530,000**
N=8	1,420.48	16,659.2	160,398	1,426,040	**12,101,800**	**99,546,700**	**800,659,000**	**6,331,440,000**

The Big Picture, ***h(n, t, s)*** and ***k(n, t, R)*** combined.

What's really stunning about the above table is the stark realization that in supporting numbers of users comparable to Napster, Gnutella would generate more than an unbelievably significant *800MB* worth of data for *just one* of those users to search the entire network for "grateful dead live" and receive responses.

Our job is still not finished yet, though. What remains is to apply these statistics to observed query rates to gain an understanding of the real-time impact of a GnutellaNet on a network.

Behold, The Firestorm

When Napster, Inc. was served with an injunction designed to halt all file-sharing service through the Napster network, Gnutella and similar services experienced what is now commonly referred to as the "Napster Flood". While an inordinate number of users perceived the injunction as their personal charge to download from Napster as much as possible before the service was brought down, still a great many flocked to other file-sharing services such as Gnutella.

During this period of time, Clip2 DSS observed query rates peaking at 10 queries per second, double the normal 3-5 per second. The possibility of exceeding 10 qps during periods of heavy usage these days is not unlikely.

The final item of interest in this paper is the extrapolation of bandwidth rates (per second) from the bandwidth costs calculated above and observed rates. For thoroughness, query rates for a quiet (3qps), normal (5 qps), and burdened (10 qps) GnutellaNet are examined. For each test case, the main assumption is that Joe Smith's behaviour satisfies the typical user demographic.

Bandwidth rates for 3 qps (*S=83, R=94.56*)

	T=1	***T=2***	***T=3***	***T=4***	***T=5***	***T=6***	***T=7***	***T=8***
N=2								
N=3	1.6KBps	6.5KBps	19.7KBps	52.9KBps	132.9KBps	320.2KBps	749.3KBps	**1.7MBps**
N=4	2.1KBps	11.9KBps	51.5KBps	201KBps	741KBps	**2.6MBps**	**9.1MBps**	**31.2MBps**
N=5	2.7KBps	19KBps	107KBps	548.8KBps	**2.7MBps**	**12.7MBps**	**58.4MBps**	**264MBps**
N=6	3.2KBps	27.7KBps	192.7KBps	**1.2MBps**	**7.5MBps**	**44.1MBps**	**253.6MBps**	**1.4GBps**
N=7	3.7KBps	38.1KBps	315.2KBps	**2.4MBps**	**17.5MBps**	**123.7MBps**	**853.6MBps**	**5.8GBps**
N=8	4.2KBps	50KBps	481.2KBps	**4.3MBps**	**36.3MBps**	**298.6MBps**	**2.4GBps**	**19GBps**

Bandwidth rates for 5 qps (*S=83, R=94.56*)

T=1	***T=2***	***T=3***	***T=4***	***T=5***	***T=6***	***T=7***	***T=8***

N=2								
N=3	2.7KBps	10.8KBps	32.8KBps	88.1KBps	221.6KBps	533.7KBps	**1.2MBps**	**2.9MBps**
N=4	3.6KBps	19.9KBps	85.9KBps	335KBps	**1.2MBps**	**4.4MBps**	**15.3MBps**	**52MBps**
N=5	4.4KBps	31.7KBps	178.3KBps	916.3KBps	**4.5MBps**	**21.1MBps**	**97.4MBps**	**441.3MBps**
N=6	5.3KBps	46.1KBps	321.2KBps	**2.1MBps**	**12.5MBps**	**73.4MBps**	**422.6MBps**	**2.4GBps**
N=7	6.2KBps	63.4KBps	525.4KBps	**4MBps**	**29.2MBps**	**206.2MBps**	**1.4GBps**	**9.6GBps**
N=8	7.1KBps	83.3KBps	802KBps	**7.1MBps**	**60.5MBps**	**497.7MBps**	**4GBps**	**31.7GBps**

Bandwidth rates for 10 qps (*S=83, R=94.56*)

	T=1	***T=2***	***T=3***	***T=4***	***T=5***	***T=6***	***T=7***	***T=8***
N=2								
N=3	5.4KBps	21.6KBps	65.6KBps	176.2KBps	443.2KBps	**1.1MBps**	**2.4MBps**	**5.8MBps**
N=4	7.2KBps	39.8KBps	171.8KBps	670KBps	**2.4MBps**	**8.8MBps**	**30.6MBps**	**104MBps**
N=5	8.8KBps	63.4KBps	356.6KBps	**1.8MBps**	**9MBps**	**42.2MBps**	**194.8MBps**	**882.6MBps**
N=6	10.6KBps	92.2KBps	642.4KBps	**4.2MBps**	**25MBps**	**146.8MBps**	**845.2MBps**	**4.8GBps**
N=7	12.4KBps	126.8KBps	**1.1MBps**	**8MBps**	**58.4MBps**	**412.4MBps**	**2.8GBps**	**19.2GBps**
N=8	14.2KBps	166.6KBps	**1.6MBps**	**14.2MBps**	**121MBps**	**995.4MBps**	**8GBps**	**63.4GBps**

Keeping things in Perspective

From the charts above, it becomes mind-numbingly clear that the Gnutella distributed architecture is fundamentally flawed and can have a horrific impact on any network. On a slow day, a GnutellaNet would have to move *2.4 gigabytes per second* in order to support numbers of users comparable to Napster. On a heavy day, *8 gigabytes per second*.

A lot of potentially obscure assumptions are made here, though, and they should be carefully examined and understood before making conclusions:

- the test GnutellaNet is ideal, which is to say that all participants form a topology which conforms to ***g(n, t)***;
- being *ideal*, its topology is static -- meaning all responses to a search query are received by the requestor, without being cut off by transient nodes;
- query rates are constant,
- query demographics correlate to the average case presented above,
- all GnutellaNet participants are capable of supporting the bandwidth rates incurred,
- search queries and responses represent the only relevant and bandwidth-significant activity on the GnutellaNet.

So why should the above charts be taken with a grain of salt? Well, the real GnutellaNet that exists today is certainly not *ideal*, and has been occasionally observed persisting as several smaller, fractured GnutellaNets. Also, there's a great deal of transience in the GnutellaNet; observations yield only roughly 30-40% of participants remain for 24 hours or more. And it should be obvious to even the most casual

observer that query rates are not constant, and are more likely to burst and lull as the topology shifts and usage varies.

One important factor in evaluating the usefulness of the above is to consider the usage demographic. Current usage may show 3-5 queries per second with anywhere between 4,000 and 8,000 users, but if Gnutella were to ever grow in size, both by users and consequentially by files, search rates would likely increase dramatically. This would be for at least two reasons: more users equates to more people interested in locating content equates to more aggregate queries per second, and more content equates to wider variance in type of material equates to, quite simply, more to search for. So, applying query rates involving only thousands of users to GnutellaNet populations orders of magnitude greater in size is probably inaccurate; instead, at greater sizes, the above computed bandwidth rates are probably *much too small*. Indeed, one can extrapolate from the above, using the test case of 1,000,000 users:

- 8,000 users generate 5 queries per second, which simplified means
- 1,600 users generate 1 query per second, which then leads to
- 1,000,000 users / 1,600 users per query per second == 625 queries per second

Therefore it is more likely that, given an ideal GnutellaNet and a capable Internet, Gnutella would generate 625 queries per second with one million users instead of our test case of 5, which generates 4GBps worth of traffic just by itself. So how much data does a query rate of 625 qps generate? The calculation is left as a thoughtful exercise to the reader.

Most important of all, though, the above numbers assume a capable network connection exists for all participants. If networks weren't capable of relaying the amounts of traffic discussed above, traffic jams would occur and query rates would drop, query response rates would drop, and overall traffic rates, as a result, would drop. And we know they aren't capable; we know that a significant percentage of participants are dialup users, and their low bandwidth capabilities cause significant traffic congestion and topology fragmentation when improperly configured.

Conclusions

Even though many assumptions were made throughout the course of these calculations, some of which are provably unrealistic, these exercises still yield a useful perspective. In an ideal world, Gnutella is truly a "broadband killer app" in the most literal of senses -- it can easily bring the Internet infrastructure to its knees. And it should also be noted that only search query and response traffic was accounted for, omitting various other types of Gnutella traffic such as PING, PONG, and most importantly, the bandwidth costs incurred by actual file transfers. 2.4GBps is just search and response traffic, but what about the obnoxiously large amount of bandwidth necessary to transfer files between clients?

Those reading this paper should be careful to note that non-intended uses of the GnutellaNet also incur noticeable bandwidth hits: using search queries to chat with other participants, SPAM placed inside search queries and results to advertise various things, and gibberish, typically resulting from misbehaving users or clients. Futhermore, with individuals writing their own clients and protocol extensions, we may begin to see loop detection being rendered useless. Depending on how individual

clients implement loop detection (comparing message ID's versus comparing message ID's + a checksum of the packet's payload), protocol extensions may interfere with legacy clients and result in more traffic than necessary being generated and relayed.

The main argument against this paper is that GnutellaNets are never ideal, and as adoption and usage grows, are statistically less likely to be ideal, given the increase in complexity of the topology as the number of participants increase. I would agree with this principle, but I believe it only serves as better proof of the premise: if an ideally distributed and fully capable network generates 2.4GBps to accomodate 1M users (and we already know this figure to be unrealistic in terms of what the modern Internet is capable of), then a poorly distributed network with insufficient bandwidth will certainly not be able to support the same number of participants or the traffic they generate. In other words, again, *Gnutella can't scale*.

Another key argument against these computations is that they are all focused on the *center* of an ideal GnutellaNet, and applying this generalization to all configurations of nodes is misleading and inaccurate. Traffic is measured and generalized from a maximizable point; this is to say that the "center" node will always generate the most amount of traffic given the same configuration throughout, whereas a leaf node in an *ideal* GnutellaNet generates only a fraction of that bandwidth. However, empirical analysis yields the observation that, in practice, leaf nodes don't generally have only one connection into the GnutellaNet. As a matter of fact, leaf nodes don't tend to occur naturally at all, since it is rarely in a participant's best interest to limit themselves to one connection, in maximizing bandwidth capacity versus search depth. To date I've only observed this happening on a large scale with Reflectors, or strategically placed Gnutella "proxies" at high bandwidth locations on the Internet aimed at serving dialup and other small capacity clients. So, the inaccuracy of these numbers likely lies in their being, again, *much too small*. Also, regardless of how intertwined and convoluted the connection paths are, the data path is effectively rendered semi-ideal through loop detection, so the methodology turns out to be more realistic than first thought.

Yet another valid question to raise against the premise is, What is a reasonable size? Is it 100 users? Is it 1,000? Or 100,000? Or 1,000,000? Nothing short of global domination? Discerning what's reasonable is assuredly a subjective comparison, however, I use the phrasage interchangably with original statements like "Gnutella will kick Napster in the pants." Common sense dictates that in order to accomplish that, Gnutella would have to perform more efficiently, scale higher, and be more capable. These exercises prove that, on a perfect level, Gnutella just can't rise to meet the challenge. Consequentially, they prove that on an imperfect level Gnutella has no *hope* of performing on the same level.

In the final assessment, it's painfully obvious that Gnutella needs a complete overhaul. Major architectural flaws are fundamental in nature and cannot be mitigated effectively without redesign at the most basic level. Some intelligent caching could likely benefit the Gnutella architecture, since observations yield that many searches and responses result in repetitive, duplicate transmissions. However, given the transience of GnutellaNet participants, and the wide variety of participating clients, it would be difficult to predict with any amount of accuracy how effective technology like this would be.

Various efforts claim to be underway to redesign the protocol; among them, gPulp stands out as the farthest along, with message boards and mailing lists set up for those wanting to get involved. But, with its mission of consentual changes implemented through a working group, I harbor significant doubt as to whether they will ever be timely and effective at producing an alternative. GnutellaWorld, another revamp effort recently publicized by CNet's news.com, takes the lead on the initiative for developing

Gnutella2. J.C. Nicholas, apparently representing GnutellaWorld, claimed in an interview with CNet that Gnutella2 technology would be out "soon". Characterized as an "Internet Earthquake" and promised to be "the greatest revolution since Linux", Gnutella2 sounds more like the same old hype than anything else. And with only 8-9 months under their collective belt as an organization, I personally wonder how far along efforts could be. If the fact that this open-source project's CVS repository remains quite empty, or that its mailing lists appear dormant presents any indication of progress, the Internet probably has some time to go before experiencing the next internet cataclysm. Considering GnutellaWorld's intentions of supporting 20 million people or more, I can only hope that it's nothing like the original Gnutella.