Errors in the Levine 2017 paper on attacks against Freenet

This is a short two-hour braindump from checking the weaknesses of the paper Levine et al. (2017) "Statistical Detection of Downloaders in Freenet".

It is neither diplomatic nor aggressive, just unfiltered analysis of the publication with the background of knowing Freenet and being interested in its communication for more than 13 years.

Update 2023: at the end of this article there's now a more detailed and math-heavy breakdown of the errors Levine et al. committed: 7

Update 2020-12: The Levine group released yet another report, ~~not peer reviewed,~~ (**correction: it was reviewed, see the CCS abstract**) in which they claim to correct their errors. I did not read the details yet, because they note a previous lawsuit as success which was built on the results of a previous publication which was of even worse quality than the one debunked here.
Due to that their statement is so seriously misleading that I would characterize it as **academic misconduct**; I don't want to waste any more time on it.
This now got them in the unfavorable situation that integrity would require them to tell at least the first court that their new research proves what Freenet developers warned them against: that the evidence about which they claimed in court that it were robust had so little substance that it essentially was a bunch of lies.
They also claim that they can track uploaders, but they only show that for known keys. Recommendations Freenet has been giving for over a decade: Don't insert known files under CHK (which leads to a known key). Use the friend-to-friend mode.
Also they claim that their method works against Freenets friend-to-friend mode while admitting that this requires getting a direct connection to a target node — which is just what the friend-to-friend mode protects against, so their statement is false.
Based on misleading and false statements like these they argument in their conclusions that Freenet offers no meaningful security.
Again they did not contact Freenet developers prior to publication. If they had done that, they would have received constructive criticism as answer, since they did not, all they get is a damning review.

Contents

1 Wrong false positives rate

The core pillar of the detection they name is their claim of a 2.3% **false positives rate**. But this claim is **wrong**, because they only reach it through many **false assumptions**:

- They **ignore** that friend-of-a-friend routing breaks their metric when
  - an intermediary node, or
  - the observing node **has many connections**.

  which is not the rare case but **the normal case**.
- They assume that they only get a false positive, if a request for a given file reached them with both HTL 18/17 and HTL 16. But the routing algorithm within Freenet causes them to **almost always** receive requests from a given node **over the same route**. So they get the same HTL, regardless of the actual number of hops from the source. Therefore Their 2.3% false positives rate contains a mixture of
  - the probability of **two people** requesting the file in the same interval and
  - the rate of **routing-changes** within Freenet (for example because a node on the path went offline). If a request from a given peer is received both from HTL 17 and from HTL 16 then routing changed, otherwise this should not happen.

  Their **false positives rate** when measuring with only one node is therefore **meaningless**. They would need multiple nodes that all see the request.

2 Wrong math

In addition **their math is wrong**:

> We construct a model by assuming that each request the downloader makes is sent to exactly one of its peers, and that the selection of that peer is made **uniformly at random**.

This does not take friend of a friend routing into account. Therefore their math is wrong: It does not match the actual selection of peers, so the results are meaningless for the actual Freenet.

3 Wrong model

And their **model** of how measurement works **is wrong**:

> a simple expected fraction of 1/degree for the adjacent and $(1/\text{degree})^2$ for the two-hop case.

This does not take the degree of the measuring node into account, therefore it is not a model of routing in Freenet.

4 Summary (TLDR)

Their false positives rate is wrong, their math is wrong, and their model is wrong. Therefore results you get when using their method are false.

5 Remarks

Keep in mind, that this is the result of a quick two hour check of the paper. I might also have gotten things wrong here. Also, to somewhat save the grace for the Levine-team: They at least tried to actually measure the false positives rate. They did it wrong and drew false conclusions, and that they tried doesn't make it right and it doesn't excuse persecuting people based on their flawed reasoning, but at least they tried.

That the Levine-team **did not contact Freenet developers** prior to publication is **inexcusable**, though. It's like publishing a paper based on evaluations of particle beams from CERN without ever talking to someone from CERN. Can it be so hard to write an email to press -at- freenetproject.org saying *"Hi, we found a method to track Freenet downloaders and drafted a paper based on that. Could you have a look to see whether we missed something?"*

6 Final words

If you want to know the actual requirements for calculating a false positives rate in Freenet, head over to https://www.freenetproject.org and read the article Statistical results without false positives check are most likely wrong.

7 Update 2023: breakdown

Since Levine et al. still operate with their flawed math, here's a longer breakdown. This is still a quick explanation, not reviewed, so I may have errors in here.

7.1 False math

This is the equation Levine et al. use in their 2017 text.

$$Pr(H1|r) = \frac{\frac{1}{g+1}B(r;T,\frac{1}{g})}{\frac{1}{g+1}B(r;T,\frac{1}{g}) + \frac{g}{g+1}B(r;T,\frac{1}{gh})} \quad (1)$$

This is a probability calculation that models how likely it is that the originating node is one hop apart (H1) when receiving a number of requests (r).

By using a binomial distribution (B), they model the pobability of receiving a specific number of requests from a node with a given number of peers.

Let's take it apart:

- H1 means "one hop apart": a downloader.
- H2 means "two hops apart": a relayer.
- r is the number of requests received.
- T is the total number of requests for the file sent by a downloader.
- g is the number of peers of the observed node.
- g+1 is the number of possible downloaders: the connected node and each of its peers.
- 1/g is the definition of even share.
- 1/gh is the definition of taking even share twice (h is the guessed number of peers of a node two steps away).

Pr(H1|r) means the probability that the node is a downloader if r requests were observed (H1: one hop apart).

B(r; T, 1/g) means: the probability of receiving r requests from a downloader. B(r; T, 1/gh) means: the probability of receiving r requests from a relayer.

*Using 1/g and 1/gh in there is why they must assume that Freenet uses **even share**. Since Freenet does no use even share, their math is wrong and their whole calculation breaks apart. More on that later.*

They now weight the probability of the downloader by 1/g+1, because it is only one node. Then they weight the probability of the relayer-case by g/g+1, because there are then g possible actual sources.

This all looks good, except for the **definition of g and h** (the **most influencial variables** in the formula).

A correct formula would not use the number of peers for g and h.

For the **downloader** it would replace **(1/g) with the fraction of FOAFs**.

For the **relayer** it would use replace **(1/h: with the fraction of FOAFs** and **(1/g) with a weighted fraction of FOAFs** with **higher weight** to FOAFs close to the location of the request, because here requests are already likely to be specialized by location. If the connected node is close to the locations of the received requests, up to 70% of its FOAFs will have to get higher weight.

This latter part is only required since the fix in response to the 2014 paper. Before 2014 this could be ignored, because the link length distribution was broken.

It would still possible to scale g for the downloader and h for the relayer to account for the likely number of FOAFs assuming a known distribution of peer-counts. This part of the formula they could salvage (it is currently wrong).

But g for the relayer is plain wrong, because it does not take weighting of closeness to the request into account. In other words: for a relayer they can no longer assume a flat distribution of requests in location space.

Since g and h are wrong the whole calculation is wrong. You cannot just fudge binomial coefficients and then assume that the result is still correct, because binomials are extremely sensitive to these coefficients.

$$B(r; T, \frac{1}{g}) = \binom{T}{r} \frac{1}{g}^r \left(1 - \frac{1}{g}\right)^{T-r} \tag{2}$$

The factor (1/g)ˆr is **exponential** in 1/g.

7.2 Sensitivity to claimed even share: Their math in code

To check how sensitive their code really is to the factors I wrote a small tool that does their calculation — you need Guile Scheme to run it.

```
;; base n!/(n! * (n-k)!) implementation
(define (factorial n)
  (let lp ((m n)
           (res 1))
    (if (zero? m) res
        (lp (1- m)
            (* m res)))))


(define (nük n k)
  (if (> k n) 0
      (/ (factorial n)
         (factorial k)
         (factorial (- n k)))))

(define (binom p n k)
    (* (nük n k)
       (expt p k)
       (expt (- 1 p) (- n k))))

(define (B k n p) (binom p n k))

(define (pr_levine r T g h)
  (exact->inexact
    (/ (* (/ 1 (+ g 1)) (B r T (/ 1 g)))
       (+ (* (/ 1 (+ g 1)) (B r T (/ 1 g)))
          (* (/ g (+ g 1)) (B r T (/ 1 (* g h))))))))
```

(h is set by Levine et al. to 8 (peers) for the real downloader, a conservative setting)

But since this math uses even share, it is wrong.

The minimal adjustment to have the formula not fully wrong before 2014[1] would be to approximate FOAF routing by taking into account that their node does not have the average peer count of 30 but 60 (similar to the observed node).

This can be done very roughly[2] by changing `1/g` to `2/g` and changes connected to that, because their probability to receive any given packet is twice as high as for an average node.

[1] Since 2014 this needs more factors to account for the better routing. This will drive the probability even lower.

[2] The standard disclaimer applies: This is not reviewed by a statisticean and could have errors.

```
(define (pr_fast_observer r T g h)
  (exact->inexact
    (/ (* (/ 2 (+ g 2)) (B r T (/ 2 g)))
       (+ (* (/ 2 (+ g 2)) (B r T (/ 2 g)))
          (* (/ g (+ g 2)) (B r T (/ 2 (* g h))))))))
```

I ran this against a scenario where their original formula yielded almost 100% probability that this is the downloader and the result was less than 0.0000001%: less than one in a billion.

Translated: compared to the scenario where the real downloader has 8 peers, by the semi-fixed math there's no chance that an observed node with these results was the downloader.

Now they could argue that 8 peers is too little for the real dowloader. That would then drive the probabilities up again. But then they are in the game of fudging statistics to yield the result they want.

And this fix still misses the adjustments for more targeted routing since the fixes in response to the 2014 paper by Roos et al which would drive the probabilities lower again.

Note also that this is so heavily dependent on the factors of the binomials, that anything it says will fake a much too high precision: it misses the messy real world with changing backoff, different routing, malicious patched nodes that claim more bandwidth than they should get, and so forth.

It hides the probability that any result they get can have been a random fluctuation in routing.

7.3 No even share: explanation

The math is broken without even share, but there is the claim that Freenet uses random peer selection and therefore even share request distribution. They claim that Roos et al. 2014 shows that.

The Roos paper never stated that there is even share routing. It said that there was over 70% random routing. Random routing does not mean even share.

Random routing means that while selection is actually biased towards larger nodes (via FOAF), it does not move towards the general goal.

This is as if you walked around a city and at crossings you selected the direction by following a car chosen at random.

You'd think at every crossing:

- "Where should I go? That car looks nice, I'll follow it!"

If you do that, you will most times choose large roads, not small sideroads, because there are more cars driving along the large roads. By selecting a car to follow at random, you are most likely to walk on large roads, because there are more cars driving on those large roads than on small roads.

You do not move towards your goal, though.

This problem (not moving towards your goal because most time you select a car at random) is what Roos 2014 described: over 70% of our routing decisions were effectively like that and less than 30% were actually directed.[3]

The preferred selection of the larger road is what Levine et al. ignore. It has been true before 2014 and it is still true today.

What changed after 2014 is the layout of the roads: now at least 70% of the roads are connected such that you can actually navigate by following cars with a license plate close to your goal.

Why does it so often come down to cars when searching an explanation that all can understand?

7.4 No even share: the code

That's a nice analogy, but how can one check whether Freenet operates this way? There is the claim that Freenet developers intended to have FOAF routing, but that this is not what is actually happening.

Regardless of who asks, I would ask them to look into the sourcecode: the absolute law of software. That says clearly in PeerManager.java:

```
for(int i = 0; i < peers.length; i++) { // this checks every peer
    PeerNode p = peers[i]; // the peer being checked
    // ...
    double l = p.getClosestPeerLocation(target, excludeLocations);
    // ^ this asks for the FOAF that is closest to the target.
    // ...
    if(!backedOff && (diff < closestNotBackedOffDistance // ...
        closestNotBackedOffDistance = diff;
        closestNotBackedOff = p; // here the peer is remembered if its
        ↪  FOAF is best
```

[3] In Freenet routing the larger roads are the peers with more connections. The long link lengths described by Roos et al. mean that the roads lead to different places, but they don't mean that they are large. If you drive up a highway and the next stop is in another city, that's a long link. Imagine having to navigate in a city where most roads are highways to other cities (though not necessarily fast or much used, just having a target very far away), so to get to your destinations, you'd have to bounce around between cities until one highway drops you off in the right neighborhood. That was the situation before 2014.

```
    // ...
PeerNode best = closestNotBackedOff;
// ^ here the best peer is chosen based on having the best FOAF location
```

and p.getClosestPeerLocation() that is run for every single peer says:

```
locs = currentPeersLocation; // these are the FOAFs: all the peers of
↪  that peer
// ...
closest = findClosestLocation(locs, l); // binary search in the list of
↪  FOAFs
// ...
return locs[closest]; // this is the peer of that peer closest to the
↪  target - the FOAF
```

That is how Freenet determines which node to choose: Check for every single peer every FOAF it reported to see whether it is closer than the best FOAF of any other peer. Send the packet to the peer (that's not backed off) with the closest FOAF.

These are the laws of Freenet routing. The nodes cannot disobey them, except if someone changes these laws for their node (by patching it).

This is not even share.

And there is no "they intended to". What Freenet actually does is right there in the source-code: the absolute law how a program operates. It cannot disobey that, it cannot bend that. Laws for programs are absolute and unbending. They can contain bugs, but that's why they get reviewed.

The core of the routing are just these 10 lines of code (not counting comments, the parts of lines starting at //). There are contests for writing obfuscated code that does something else than what it seems to do, but there's no obfuscation here: the code is plain and simple and the program must obey it.

This is software, and the laws of software do not bend.

7.5 No even share: measured

OK, but did you actually measure that? Maybe there is a bug you did not see!

Freenet is used by people with very different internet connection speeds. It can utilize speeds from 10kiB/s (roughly a phone line) to 1000kiB/s (this can stream HD video).[4]

[4]Freenet is not limited to at most 1MiB/s. It can utilize higher speeds — that's how a node can have 69 peers — but it's not guaranteed to find enough peers for that, and at such high speeds additional rules apply that increase the share of the requests that large nodes receive even further.

But the peer count only increases with the square root of the bandwidth, so a node with 10kiB/s gets 5 peers, a node with 100kiB gets 17 peers and a node with 1000kiB/s gets 55 peers.

If it did even share, nodes with 5 peers would constantly be overloaded, because they would receive on average 25 kiB/s of messages (5kiB/s in data **per connection**), more than 2x as much as their total bandwidth.

But nodes with 5 peers work and are **not** constantly overloaded.

So Freenet uses FOAF routing — and this is measured constantly, because otherwise the Freenet network would break down.

But what exactly would you measure (test)?

You would measure backoff. Since slower nodes would be overloaded by faster nodes, you'd expect a backoff well above 30%. It's convenient that Levine et al. 2020 measured backoff and found only about 11% backoff. They just did not note that by measuring this they proved that Freenet does not use even share routing.

Also (observable locally) since the average node in Freenet has 30 peers, if Freenet used even share you would expect a node with 15 peers to be constantly in backoff / overload. I run such a slower node (because my internet speed is limited) and this is not the case.

7.6 False positives: proven non-downloaders indicted

The Levine group tested their formula in their 2020 text for false positives, and it detected their own nodes twice.

They claim that this is a low false positives rate, because they did not point out their own nodes much more often, but in fact it means that if they had waited a bit longer, they would have reported themselves as downloaders (they wait for three marked runs), even though it is certain that their nodes did not download any of the files.

Their own nodes were pointed out by their method not just once but twice. Since they monitor thousands of other nodes, over time they will find more and more innocents.

And most of their nodes had much fewer peers than the over 60 which some people had whom they targeted (their text implies that most Levine nodes had around 30 peers).

If actually guilty people use darknet-mode or just block the Levine nodes (there are some patches floating around that would have that effect), then the Levine method will only find relayers, because they cannot connect to the actual downloaders directly, so their method cannot ever point to the actual dowloader.

They will only be able to target the clueless and the innocent.

And since they do this knowingly (they know about the debunking here), that's actually despicable.

7.7 Conclusion

Freenet does not use even share routing and the Levine method is wrong.

List of Links