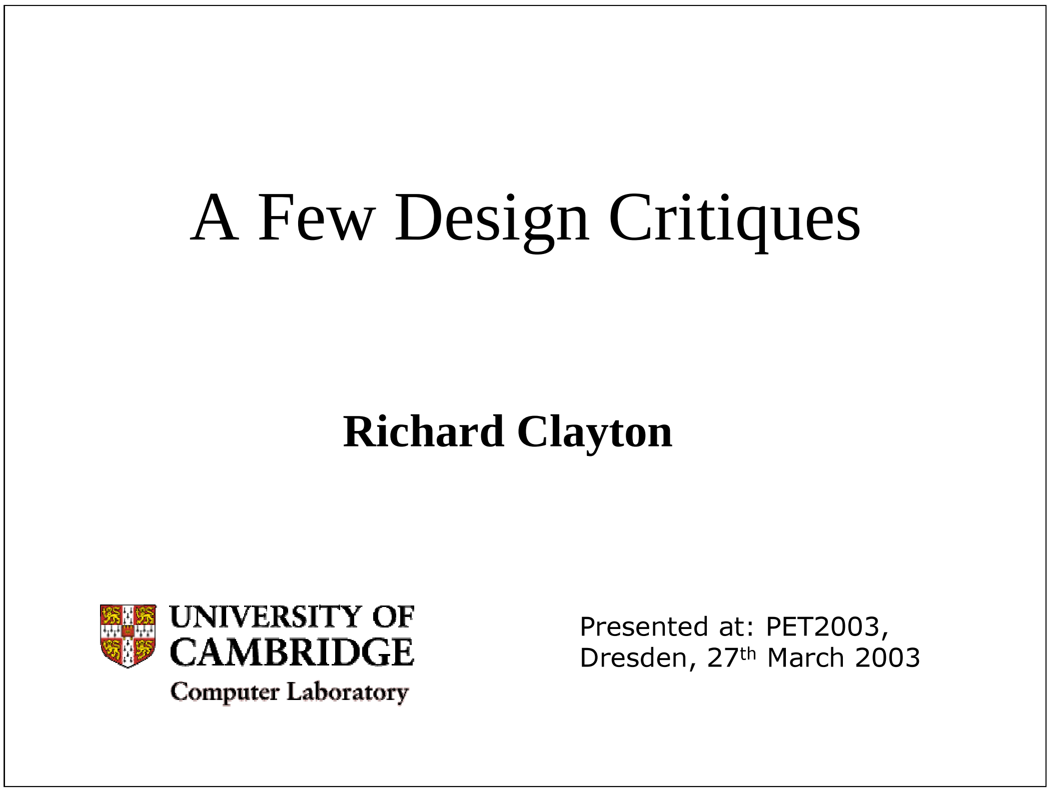


# A Few Design Critiques

**Richard Clayton**

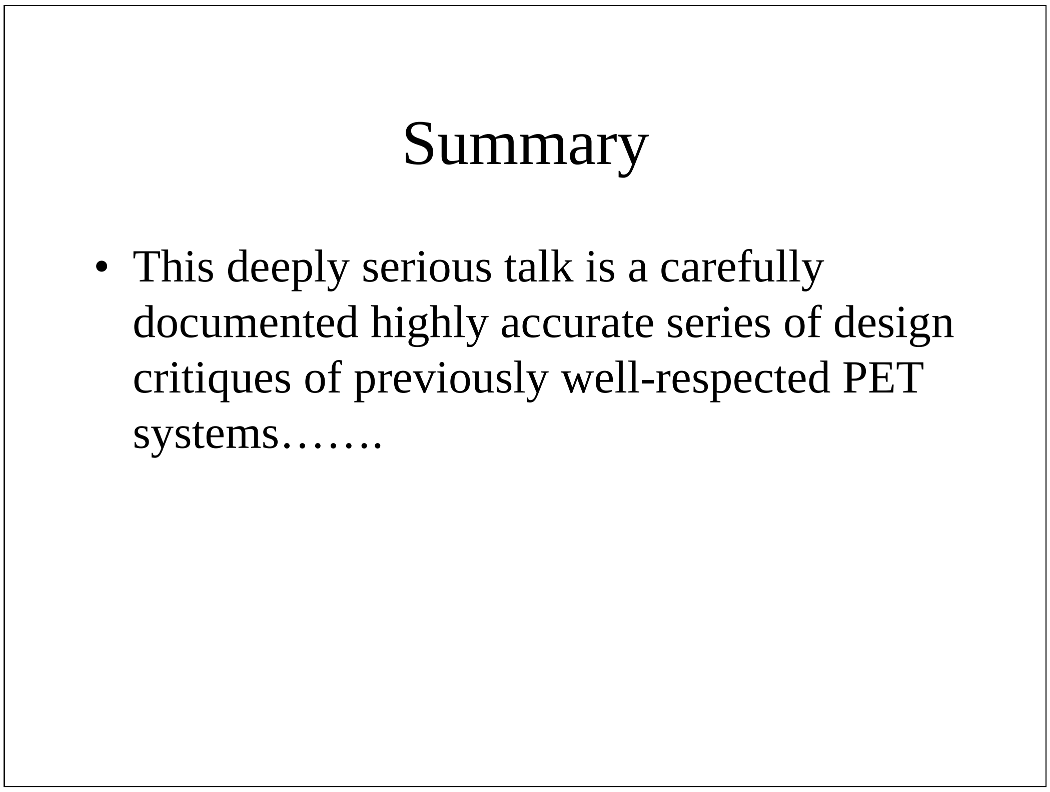
UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Presented at: PET2003,
Dresden, 27th March 2003

These slides accompanied a short talk I gave at the start of a panel discussion "Peer-to-peer, anonymity, and plausible deniability designs" at the Workshop on Privacy Enhancing Technologies (PET2003). Also on the panel were Roger Dingledine, Christian Grothoff, Dennis Kügler and Paul Syverson. The panel was moderated by Andrei Serjantov.

The notes below each slide were added afterwards so as to flesh out some of the remarks I made -- and to substantiate some of the allegations!

*richard.clayton@cl.cam.ac.uk*

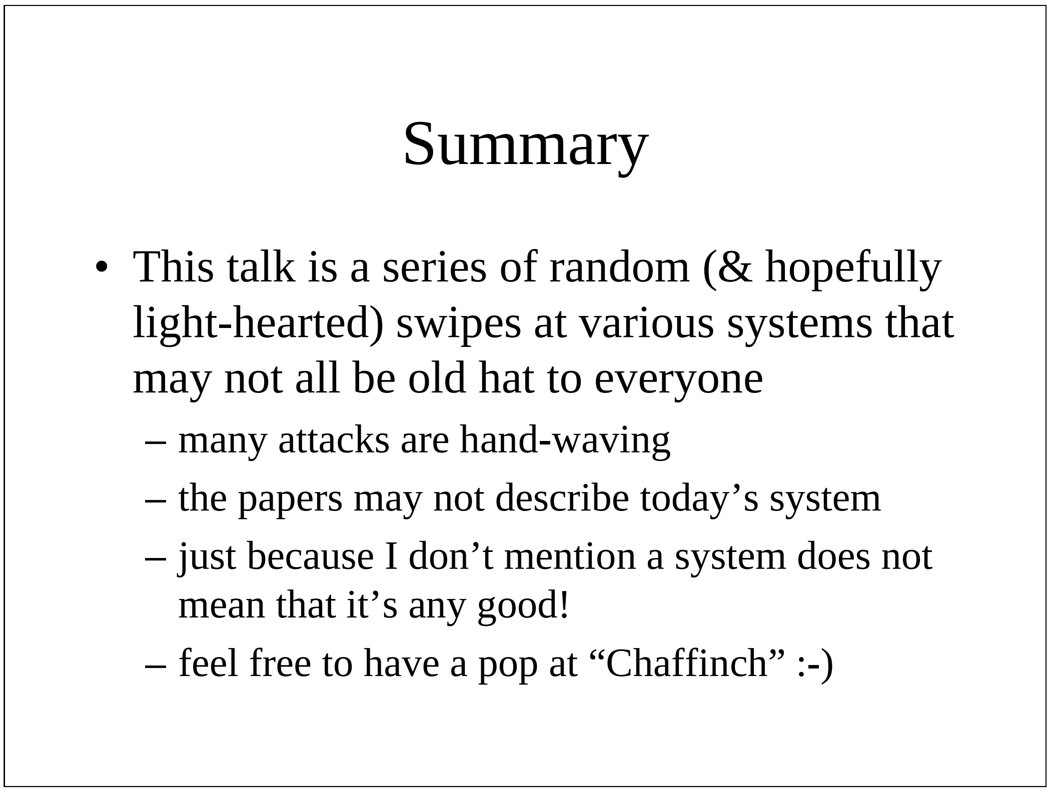


## Summary

- This deeply serious talk is a carefully documented highly accurate series of design critiques of previously well-respected PET systems…….

The talk was given in the session after lunch when it is a prerequisite of speakers to wake up their audience.

So I started by indicating that I had several decades of design experience creating mass-market software, having been involved in several projects that had sold millions (in particular, a home computer and a low-cost word-processor)….



## Summary

- This talk is a series of random (& hopefully light-hearted) swipes at various systems that may not all be old hat to everyone
  - many attacks are hand-waving
  - the papers may not describe today’s system
  - just because I don’t mention a system does not mean that it’s any good!
  - feel free to have a pop at “Chaffinch” :-)

… and then I did the waking up bit, by indicating the tenor of the talk!

The talk discussed fairly generic attacks and failures. I also remarked that it was often hard to determine which version of a system was described in academic papers and which version was currently fielded. It does not seem to be the habit of P2P and PET system authors to give version numbers to their designs so that one knows, for example, that a paper describes v1, a documented attack is on v2 and v3.5 is currently available for download!

Chaffinch is the system for “plausibly deniable” communication that George Danezis and I designed and built. So far no-one has broken it!

“Chaffinch: Confidentiality in the Face of Legal Threats”, R. Clayton and G. Danezis, in “Information Hiding, 5th International Workshop, IH2002, Noordwijkerhout, The Netherlands, October 7-9, 2002”, ed: F. Petitcolas, LNCS 2578, Springer Verlag, 2002.

<rant>

## Predictability matters

- Freenet places content in a predefined place (which is where it is looked for first)
- Content migrates towards the users (design aim for efficiency & to prevent censoring)
- Unused content will be removed and eventually will also be forgotten about
- Combine these and content can be there on the network for some & invisible to others!

Freenet has a homepage at http://www.freenetproject.org/

The predefined place for content storage is derived from the content-hash key. Searches of Freenet space for material proceed by moving towards that canonical location, hoping to find the material on the route or near to the original placing. By caching material, nodes will efficiently satisfy a repeated request and will not bother accessing the original storage location.

Material that is not being accessed will be discarded and (on another timescale) forgotten about. There are no guarantees of permanence in Freenet.

However, this can lead to strange effects where one group of users will be able to access material (because the servers they access have local copies which have, effectively, migrated towards the place the material is most used) and they will perceive the material as being fully available. But another user’s queries will look for the material at the original storage location and, unless they are lucky and a search “nearby” locates the new storage location(s) of the material, they will perceive the material to be missing.

I would argue that having different people acquiring different perceptions of what is stored in a system is an undesirable property.

## Bootstrapping is hard

- Distributions can be ancient! Server lists on web servers is OK if small and static and the web page is updated! (cf Mixmaster) and not slashdotted (cf Mojo Nation)
- Fetching a list of servers at random sounds like a good idea (early Tarzan design) but if observed can remove all anonymity! [see obscure footnote in Mixminion paper!]

Bootstrapping PET systems is a complex practical and theoretical problem.

For instance: many distributions of software for using Mixmaster remailers contain lists of servers (and keys) that are years out of date. When they helpfully provide URLs for accessing up-to-date information, these turn out not to exist either.

Mojo Nation found that their bootstrapping scheme collapsed when they were slashdotted (new joiners went from 300 a day to 10,000). There’s a great deal more practical experience in Zooko’s IPTPS 2002 paper (LNCS 2429, pp 104-110) [and see also p5 for the Workshop Report account].

Tarzan’s original node discovery algorithm (LNCS 2429, pp 121-129) was to make random accesses into a Chord ring and use a subset of the nodes learnt about in that manner. Unfortunately (as George Danezis spotted) if your node discovery process is observed then the randomness works against you. An observer who inspects a particular node in the network (or better yet, corruptly operates it) can determine from the triple (source, here, destination) whether it is your packet that is passing by. This is because triples are effectively unique unless all users go to the effort of learning about large numbers of nodes (a lot more than 10% of the total network).

## Build it properly

- MIXs have been very unreliable
  - one 9 reliability is an issue for many servers
  - clients crash or fail to drive SMTP correctly
- Reputation schemes may just drive traffic onto the NSA's six-9's kit
  - only hope for anonymity then will be to route via MI6 or the DGSE server!
- Don't ship the proof-of-concept code!

People have been measuring the reliability of MIX systems for some time. Published statistics indicate many "available" servers failing to operate correctly 10% of the time though this is better than than in the past because the server code base has been markedly improved. Historical statistics can be viewed at: http://www.google.com/groups?as_usubject=mixmaster%20re mailer%20reliability%20statistics&as_scoring=d and current stats at, for example, http://outel.org/mix/mlist2.html. As I write (mid April 2003) about half the available servers are managing 100% availability for the past week).

The underlying lack of robustness makes the 1999 claim by RProcess [who had spent some time improving reliability in the code (s)he wrote] that a "third influence" was running a selective DoS attack against the remailer network rather harder to credit than it might be otherwise

http://www.eff.org/Privacy/Anonymity/1999_09_DoS_remail_vuln.html

viz: if friends lose lots of mail, how can you detect enemy action?

Remailer reputation schemes suffer from the problem that although you can detect the hop over which traffic is being lost, it is not always possible to allocate the blame to one end or the other. Dingledine et al proposed a witness scheme to tackle this (R.Dingledine, M.J.Freedman, D.Hopgood & D.Molnar, "A reputation system to increase MIX-net reliability", IHW2001, LNCS 2137). However even if such a scheme was available to us, would it not just drive traffic to the ultra-reliable systems that only major governments could afford to fund ? Then one would only be saved from exposure by the CIA and FBI failing to share their traffic information about you!

## Provide instant gratification

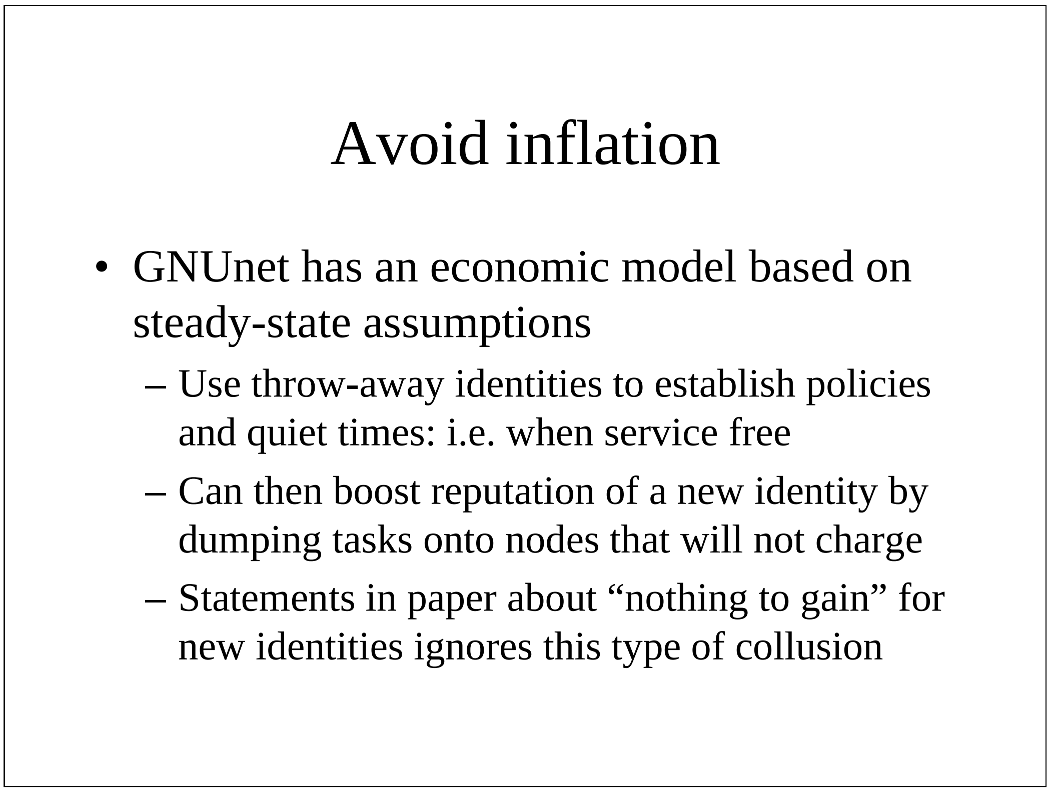
- Program needs to work “out of the box”
- It also needs to be useful “out of the box”
- Users are not going to be very interested in “serving an apprenticeship” before they can do all the cool things
- However, if your system is money based then this will lead to inflation!

The Mojo Nation experience (in the IPTPS 2002 paper already cited) was that 80-85% of users were part of the system one time only, and for less than an hour. A significant fraction of the remainder used the system for less than 24 hours. Putting it another way, the network “half-life” [viz: until 1/2 the network was new] was less than an hour.

Many people are currently proposing economic models (not always using money, but currencies based on trust, reputation etc) as being the way to create solutions to various problems in Peer-to-Peer systems and in anonymity systems generally.

But the lesson from Mojo Nation (and indeed other systems) is that people have a short attention span, and so your system needs to work immediately.

Hence, if your economic system doesn’t allow new users to participate then they’re going to have to be mighty impressed by the future rewards to stick around. If your system does allow new users to consume significant resources then you either have a wasteland (because they don’t pay for anything and so don’t ameliorate their usage), or an economy with rampant inflation (because you print money for them to spend).



## Avoid inflation

- GNUnet has an economic model based on steady-state assumptions
  - Use throw-away identities to establish policies and quiet times: i.e. when service free
  - Can then boost reputation of a new identity by dumping tasks onto nodes that will not charge
  - Statements in paper about “nothing to gain” for new identities ignores this type of collusion

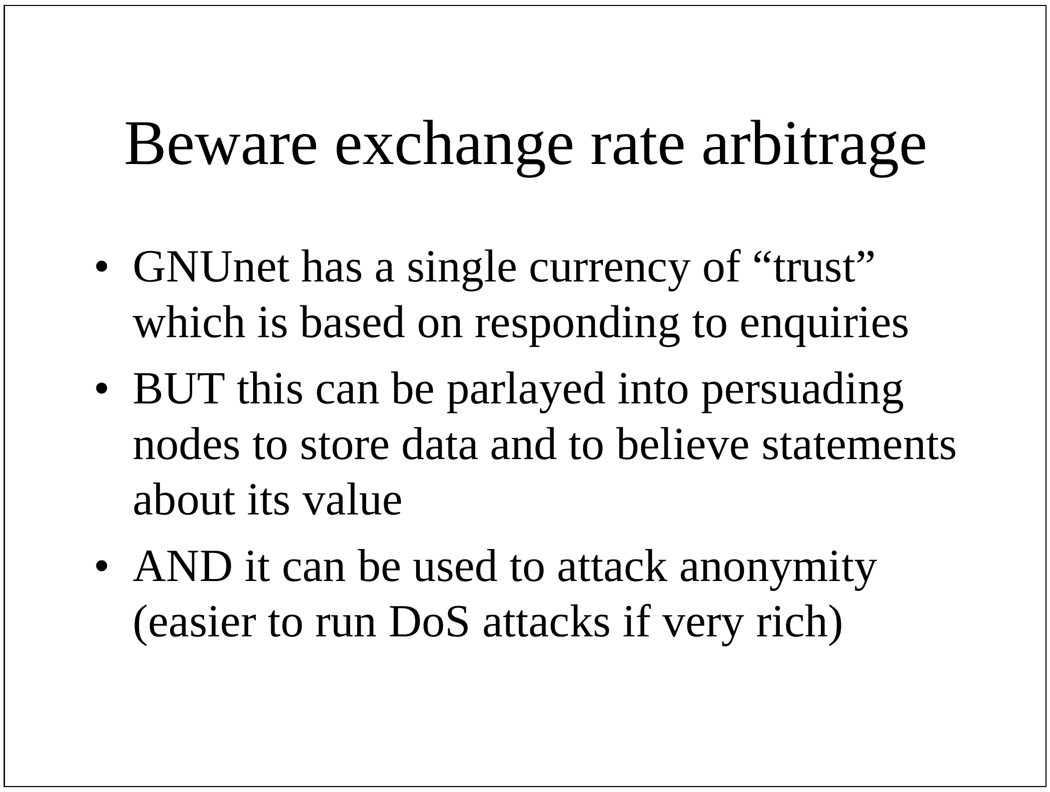
The slide above was presented when I gave the talk -- but since then it has been gently explained to me that I had significantly misunderstood GNUnet’s economy and that in fact the system suffers from a significant deflation problem!! These comments attempt to correct my misapprehension:

I was assuming that I could test out the policies of neighbours by using throw away identities. These identities would determine when queries would be relayed “for free” and when ones lack of credit would cause them to be discarded. This is certainly possible, but not in fact very useful.

My cunning plan was to then use a fake identity to request material via the neighbour (at a quiet time, when lack of credit would be irrelevant) and I would satisfy the request myself. Thereby I would increase my reputation with my neighbour as a prompt provider of scarce material.

However, what I had missed was that a neighbour (assumed to be honestly running the standard GNUnet economic model), will only raise the reputation of a supplier if *they themselves initiated the request*. ie: all these requests from fake identities would not affect the scores held by the neighbour and hence no advantage would be gained by my deception.

BUT consider what happens when a node makes no requests of its own (perhaps the owner goes on holiday, or to sleep for a few hours). The node will reduce the scores of other nodes as they make requests upon it. When the node eventually gets around to making requests (the following morning perhaps) then it will have set every other node’s reputation to a rock bottom zero. So no inflation, but this type of deflation (and consequent amnesia about past performance) doesn’t seem entirely desirable either !



## Beware exchange rate arbitrage

- GNUnet has a single currency of “trust” which is based on responding to enquiries
- BUT this can be parlayed into persuading nodes to store data and to believe statements about its value
- AND it can be used to attack anonymity (easier to run DoS attacks if very rich)

GNUnet establishes trust by the willingness of other nodes to supply it with answers to queries. However, its only defence against flooding attacks that may damage anonymity is to reduce the trust of those who implement them. Viz: if you’re prepared to assist by supplying material then, later on, you may consume these resources in an attack on someone’s anonymity.

Further, since nodes may decide to store material because it appears to be popular, someone who has built up trust reserves may be able to issue requests to others to persuade them that worthless material should be kept (and by implication, valuable information discarded).

Andrei Serjantov points out a related anonymity attack. If you satisfy a request for something rare to a neighbour, then you can determine if they were the original requestor by determining if they have raised your reputation and will now answer queries for you. If they were merely passing on a request then your reputation with them will be unchanged and your query will fail to be satisfied.

As a general rule, it doesn’t seem wise to use behaviour in one sphere to influence ones judgement about likely behaviour in another.

## What’s in your corner store?

- GNUnet white paper suggests that large nodes should hold common content and small nodes the uncommon stuff
  - requests for uncommon material will be higher priced and hence worth responding to
- Not what I see -- my corner store has common things only and for specialist foods I try the hyper-market.

GNUnet nodes need to build up trust in others by being able to respond to their requests. It is suggested (GNET v0.5.2, Grothoff et al) that rare material should be stored by a low capability site (though they only say that this is a “should” and that it is up to the nodes to decide). This *might* be a rational strategy for a node to implement because requesters will be sending queries marked with a high priority. However, it’s unclear that requestors can accurately judge rarity (so may misprice their requests) and perhaps they will merely wait for a “quiet” time and then get their queries answered “for free”. viz:the node storing the material may not get a fair recompense (and may indeed only get that recompense from a handful of participants).

Even assuming the queries for rare material are fairly priced, I am still unconvinced it is rational for a node to store low-demand high-price material. I suspect that, like my corner store, nodes will go for the high-demand material because that’s what sells day-to-day and serving it will ensure they always have some trust reserves available to be used, rather than awaiting a significant windfall that might turn up just once in a blue moon.

I also wonder if it might be an effective strategy for building trust “on the cheap” to answer requests for the “front part” of files very quickly but not in fact be able to supply the rest. viz: one would try to be able to give a partial answer to anything … since that will still appear to be useful to other nodes (who would continue to request it) even when the rest of file has long disappeared from the network!

## Don’t mix aims

- Are you storing material for closed user groups or for the mass market
- If you let RIAA download from your system then they’ll know what is present
- You can use H(H(H(A))) to obscure till your heart’s content but so can they. Hence what is being stored becomes transparent!

Various storage schemes allow material to be stored encrypted (so that the provider of the space does not know what they hold). If the decryption keys are available to the attacker they can demonstrate the nature of the content. Appealing to the court that you “didn’t know” may show innocence at the level of a mathematical proof, but may not convince a judge (or jury) with deeply held feelings about the wickedness of the material being stored.

In the GNUnet scheme, data is split into small (1K) chunks and distributed onto servers. The data (and possibly other keywords) is hashed a couple of times and the results stored in indirection nodes which are then placed on servers. Queries take the form of an extra level of hashing (so that nodes cannot cheat and supply garbage). So that nodes remain unaware of what they are storing, GNUnet proposes a scheme of storing a “one time pad” as well as the file so that both must be retrieved to make sense of the data. This is fine for private data… but where any member of the public can access the file, they can clearly determine what the chunk contents look like and hence can inspect a node to determine if they are hosting this material. The hashing may obscure things in one direction… but not in the other!

Schemes to tackle this problem do exist (see for example A.Serjantov, “Anonymizing Censorship Resistant Systems”, IPTPS 2002) and PET builders should consider using systems with working deniability properties.

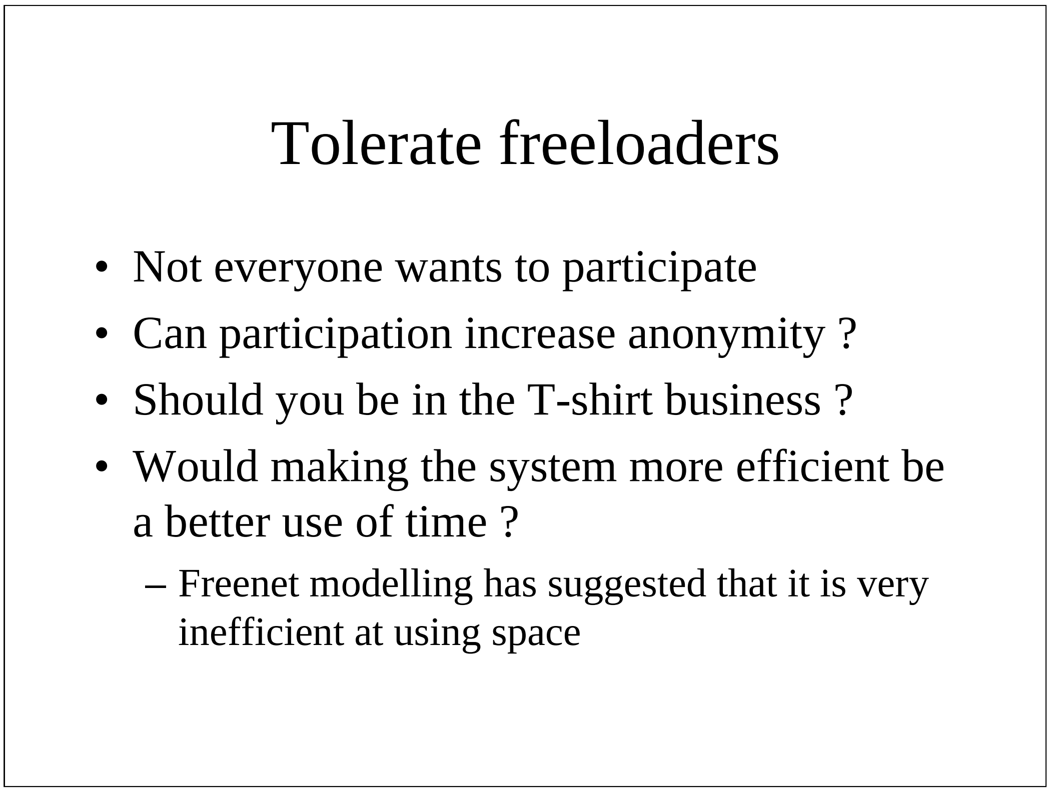
## Be nice to the network

- What’s all this with UDP ?
  - Very clever people have worked a long time to tweak TCP to make it an efficient and globally friendly way of moving data around.
  - Routers understand “flows”.
- Using UDP efficiently will mean recreating TCP (which seems like a waste of effort) and the routers will still not help you.

The Internet is optimised for transporting TCP flows from point to point (and to a lesser extent, UDP flows). Packets that fit a relatively simple model of data transport will typically be handled by customised hardware rather than processed by a router’s central processing unit.

Systems that do not fit this model, perhaps by delaying packets and then sending them out as a burst, will have performance problems and will degrade the performance of the network for others as well.

Equally, many systems that use UDP for its apparent simplicity look likely to end up reinventing TCP (or some major parts of it) so as to provide end to end guarantees of reliability. If your system needs TCP-like properties then TCP should be used!



## Tolerate freeloaders

- Not everyone wants to participate
- Can participation increase anonymity ?
- Should you be in the T-shirt business ?
- Would making the system more efficient be a better use of time ?
  - Freenet modelling has suggested that it is very inefficient at using space

“Freeloaders” are often seen as a problem, yet reserving your system only for the dedicated few may condemn it to long-term obscurity. There are valuable lessons to be learnt from the success of the 1980s BBS systems that didn’t impose download quotas, or indeed on the quality of the uploads they received when they required contributions before shipping material.

Many privacy enhancing systems need lots of traffic to be able to deliver privacy to participants. Freeloaders may be a better solution to this than having to create fake “cover traffic”.

If Freeloaders remain a problem then perhaps some out-of-the-box thinking would be desirable. Why aren’t you selling them “cool” T-shirts rather than trying to get them to pay for their connection? Would it be better to spend your time improving your system’s overall efficiency rather than designing complex systems to prevent a few people getting a “free lunch” ?

For the Freenet modelling results I had in mind (which indicated that in practice data did not move far from its initial placement and hence that new joiners didn’t provide much new capacity), see: Palomino, M.A.: “Adaptable Network Architectures”, 4th CaberNet Plenary Workshop, Pisa, Italy, 2001 (though I’m currently unable to locate an online copy to verify my recollection of the paper’s conclusions).

## Open Design Questions

- Storage
  - – store *n* times or distribute *n* of *m* shares?
- Naming
  - – pure names imply an indexing system
  - – impure names need continuity of filing scheme
- Indexing
  - – why would anyone want a Google-free system?

There are a number of fundamental design issues that we don't yet seem to have a consensus upon as a community.

Is this because there isn't a right answer ?

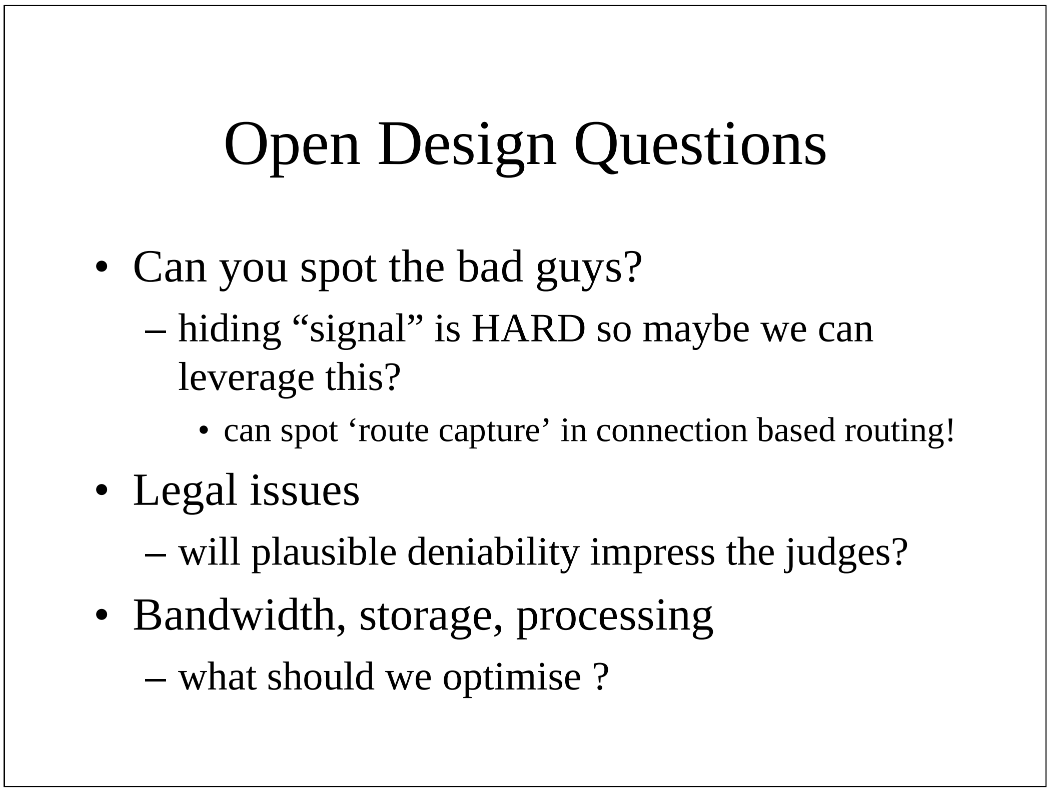
Or because no-one yet knows what the right answer is ?

Or because we're not critical enough of systems that fail to implement what we all know to be the right answer ?

eg:

When we distribute material into a PET storage system should we be keeping several copies for reliability, or should we be using an "n of m" secret sharing system ? The latter requires us to predict necessary redundancy beforehand but if we can do this then will it be more secure against attack ? or indeed will it use less disk space overall for the same level of resilience ?

Should the names of things within our system be meaningless or should they be more human friendly (indicating content). Should we be naming by using hashes of the content or settling for pure names with no semantic content at all? Where the names don't mean much how are we to index the contents, and how do we find things (why should anyone today want to use a PET that didn't have a Google within it?) and how do we deal with things that are incorrectly labelled ?



## Open Design Questions

- Can you spot the bad guys?
  - – hiding “signal” is HARD so maybe we can leverage this?
    - can spot ‘route capture’ in connection based routing!
- Legal issues
  - – will plausible deniability impress the judges?
- Bandwidth, storage, processing
  - – what should we optimise ?

Most systems try to prevent bad behaviour. But maybe it would be better design to ensure that we can detect it happening and then punish it. For example, if nodes in an onion routing system make their own decisions about where to send traffic then there is a risk that a compromised node will send the traffic only to other compromised systems and thereby the whole route will be exposed. However Marc Rennhard (“MorphMix: Peer-to-peer based anonymous Internet usage with collusion detection”, Swiss Federal Institute of Technology &c Zurich, Technical Report TIK-Nr. 147, Aug 2002) shows how this collusion can be detected by analysis of the nodes that are chosen (the compromised systems show up “too often”).

Many papers about plausible deniability make claims about how their system escapes legal responsibility. Few papers discuss which jurisdiction this claim is made within -- indicating that perhaps this is guesswork rather than informed legal opinion.

We may not be able to optimise all of bandwidth usage, storage requirements and processing load at the same time. Do we actually understand which of these are rare on today’s Internet ? The advent of 140G disks and 3GHz processors may suggest that the scarce resource is bandwidth, but DWDM and similar technologies may be changing that as well.

## Learn from Usenet

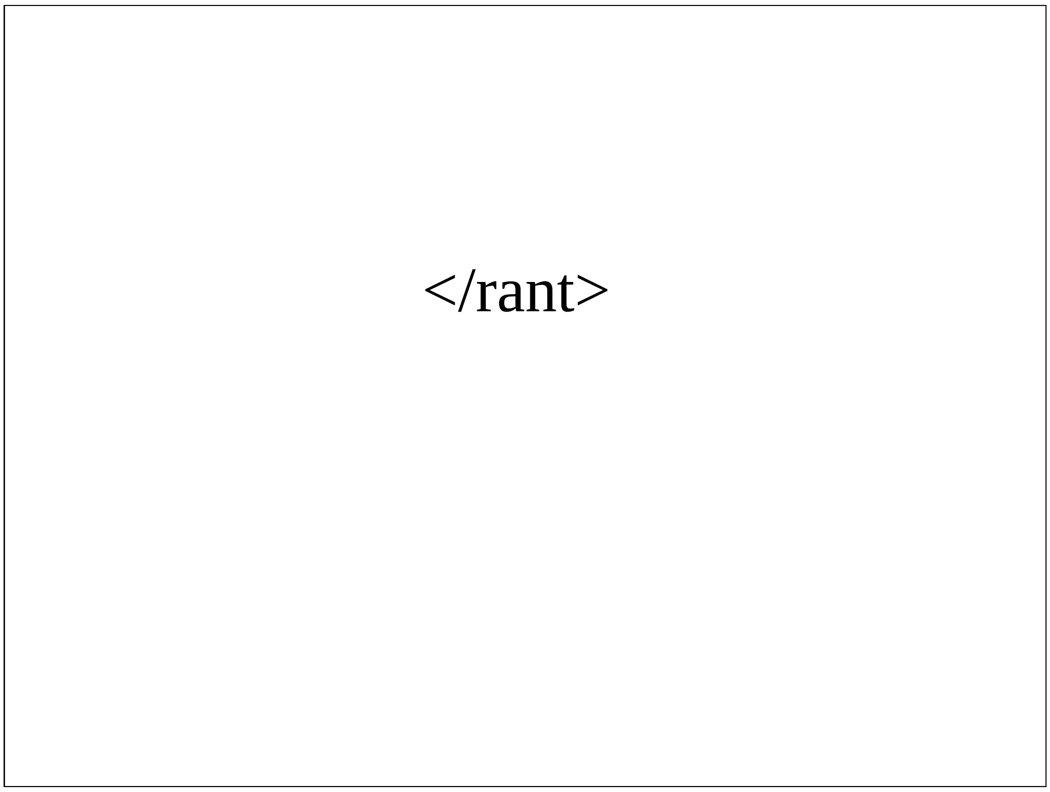
- Imminent death of the Net has been predicted many times!
  - Usenet has scaled from just a few articles to 1,000,000 articles a day (~80 Gbytes)
  - no longer “hundreds of thousands” of equal servers (Freenix 1000th = 0.08%)
  - propagation times no longer hours but seconds
  - only(!) 10% spam [down from 30%]

For a working Peer-to-Peer system that offers significant anonymity guarantees for readers and authors, Usenet is somewhat ignored by the academic community.

Despite many predictions that Usenet would fail to scale any further, it has continued to grow at exponential rates. It’s worth noting that the cynical phrase “Imminent death of the Net expected, Film at 11” that captures this doomsaying was coined by Brad Templeton as long ago as the early 1980s, when the expected death-inducing volume would have been minuscule compared to current levels.

However, Usenet has changed its nature considerably (yet again!) over the past few years with the emergence of a small number of pay-to-use sites dominating the “full feed” arena. It is no longer a network of hundreds of thousands of similar servers (if indeed it ever was). At the same time, propagation times are unrecognisable to those who once saw contributors from the Australian outback chipping into article threads a week or more after everyone else.

Usenet has also dealt with its abuse problems relatively successfully (there are lessons to be learned here), although some are still put off by the poor “signal to noise” ratio in many newsgroups. However, this is arguably a reflection of the wide usage rather than the system per se. Perhaps other systems should be aspiring to encounter similar problems!



</rant>

Somewhere in my talk (and I cannot now recall where) I gave a quick plug for John Douceur's paper on "The Sybil Attack", IPTPS 2002, LNCS 2429, pp 251-260. This sets out something we all know, that we cannot tell if two participants in our system are in fact two faces of the same person. John shows that attempts to challenge the participants to demonstrate their distinctness are doomed to fail (short of imagining a global PKI).

Fully understanding this lesson (and facing up to its implications) is vital to any design where your system aims for privacy (or indeed reliability) rely on the other nodes not ganging up on you.

## Chaffinch

Get your own back :-)

http://www.chaffinch.info

This talk has taken a swipe at several other systems…. Those who want to get their own back should have a look at Chaffinch (a system for the plausibly deniable transfer of messages) designed by myself & George Danezis

There’s doubtless lots of bad things to be said about this system as well :-)