

# The Nematic Net

## Net Memes or: How to swap news in the future

by Michel Reimon, 2003, v0.3

There are useful file types for all kinds of computer programs, there are formats for any weird application you can imagine. Except two: a) distributing news and b) sharing files via peer-to-peer-nets. Not to mention something dedicated to distribute news p2p... In this article I want to pick up the concept of memetics and suggest the development of a new file format: the net memes.

### Distributing news

The internet is obviously a wonderful way to spread news of all kind – as well as a very vulnerable one. The idea to use p2p-nets to increase security is all but new and some interesting solutions like Freenet are in development. Most of them have a very sophisticated technological concept – and lack real world, day-to-day usability.

Distributing news is very much different from sharing files on the common mp3-bazaar. One example: A news system has to deliver information about developments the consumer is not even aware of. No one would have searched for „terrorist attack“ on the morning of 9-11. A news distribution system has to be more than an archive, it has to *deliver* news. But clearly it has not to deliver all unsolicited news published by anyone: that would be a spam system. The content has to be filtered, either by human editors or automated as the meta-paper news.google.com does. Every reader then has to find his favorite sources.

One other important difference: Even slight modifications can make the content not only useless, but harmful. And feeding wrong information into the network would certainly occur. Just think about stock market news. A news system will have to take care of this problem (the fakes, not the stock markets).

Peer-to-peer means to store files decentralised. But if you have favorite sources – doesn't it make sense to connect to them directly? That's the quickest and safest way to find all files of a given source in most of cases ... and it's the weak point, against which attacks are easy to launch: Denial of Service attacks (Al Jazeera during the latest gulf war), modified articles (Yahoo News, USA Today), police raids (Independent Media Centers in Genoa and Seattle). It's how the www works and integrating this concept into p2p-nets of course imports the same flaws.

Decentralised storage is superior only when the main source is under attack. So it's a strategy for a backup system then? Well, yes and no. I spent some time trying to develop a design for such a network: clients which regularly connect to chosen servers; those either with fixed IP-addresses or temporarily connected to supernodes which point the clients to their current IP-address; and a gnutella-like p2p-net that kicks in, when a server or supernode can't be located... and some more features. However, after some intense work on it, i believe such a piece of software would be nice, sophisticated and completely useless in real life.

### **If you build it...**

Some media activists would probably experiment with it, but the broad audience wouldn't migrate to such a system, because for their everyday news the www is more convenient. When a source comes under attack a few of it's readers might change to the p2p solution. And someone may even feed the content into the network. But a backup system should already be up and running in the instant when the problem occurs. A powerful p2p net needs plenty of nodes at any time. I doubt that it's possible to build such a news-community before an attack occurs. Not with the current p2p-concepts, however sophisticated you may combine them. It's not a technical problem, it's a social problem. Unfortunately media is not like: If you build it, they will come.

But what if the readers are already there and you build news distribution networks right under their nose? What if you integrate news into existing channels and start with millions of participants from the beginning? What if you turn Kazaa and Grokster and LimeWire and Freenet into tools for publishing and sharing news? And what if you find a way, that allows news to "hop" from one net to the next easily?

Cut.

## Genes, memes and nemes

In 1976 British biologist Richard Dawkins published a book called „The Selfish Gene“. It became one of the most influential books on evolution theory with lots of breathtaking ideas, but here we are interested in only one aspect: memes.

According to Dawkins evolution is a universal process which applies not only to genes. Genes are small bits of information, encoded in DNA, that aim to replicate. Dawkins suggested, that there might exist other replicators: human culture is given from one generation to the next, it's replicated more or less perfect each time. Is there something like a „cultural gene“?

Dawkins introduced the concept of memes (an artificial word). A meme can be a melody, a thought, a slogan, a fashion or a certain method to make pots or bows. Memes ‚hop‘ from one person to another, successful ones reach a lot of people, others disappear after a while. They may develop, change, ‚mutate‘... like ideas and fashions do. Memes are the replicators of human culture.

Isn't that what we are looking for: Replicators of information? News memes? Nemes?

(Note that we are more interested in *replication* than *evolution*. Evolution implies occasional modification, something we have to be careful about. Genes and memes mutate randomly, at least generally. Nemes shouldn't.)

## Nemes, our new replicators

We're talking about file sharing, so a neme is a file. But can it be any file? An article could be plain text or a word or HTML or pdf file. What about sound bites? MPEGs? JPGs and BMPs? We want to distribute those and others.

Of course such files could be nemes, but would they make successful nemes? I doubt it. This file formats haven't been developed to be spread across the internet. Even HTML files are usually stored on a webserver – they are downloaded, viewed and briefly stored as *temporary* internet files. Well, we want *permanent* internet files, don't we?

Common file types were designed to be stored on local harddisks, on diskettes or whatever. People who had to handle them usually knew their name, their content and which folder they were stored in. But in the p2p world that's different. This is why

metadata – information about information – is one of the hottest issues for internet developers.

Then there's the trust issue. When we get a file on a disk from a person we know, we can trust the content to the extent we trust this person. That's already more difficult when we download it from the web, as the attack on yahoo news showed. With p2p it becomes a major problem. When we download an article of investigative journalist Greg P from the harddisk of an unknown person – how can we know that we can trust this source? Obviously we would like an electronic signature of Greg P to come with the article. So we have to find two files: the articles and the signature. They might be on one location or might not. If the signature can't be found, the article is useless. We have to chain them together. Or put them into one file. And when we are going to do this, why not include the metadata as well?

### How should a neme look like?

So a neme is a new file type, designed to exchange news over the internet, especially p2p nets. The basic structure of a neme is

Neme ::= (M I S)

M ist the metadata, I the information, S the signatures (we will see, that one neme can have more signatures). For reasons which will become clear a little bit later I suggest this kind of notation instead of Bacchus Naur. The brackets mark the range of the neme, the three parts have to appear in this order, upper case letters indicate that this parts can be broken into smaller units. Lets have a look on this.

### M - Metadata

I'm not going to describe a full set of Metadata, in fact I haven't developed one. But some things seem to be clear: Metadata will include a title, the byline of the author, space to mention contributors and of course a verbal description of the content. It should also include the time of creation/publishing which would allow to search for nemes published since the last connection. A kind of serial number would be helpful to keep track of an author and to search for specific files: the name of the author and the serial number make for a simple search key.

## I - Information

A basic neme could include only plain text. Actually my first idea of a neme was only plain text with metadata and a signature. But when all the people are buying multimedia PCs, why not use the possibilities of these?

Think of a neme as an envelope. We put information inside, write the metadata on the front, sign the back and have got a neme.

The information sector of a neme can include more than one file. For example it's possible to put a couple of pictures into one neme. From now on they will be distributed together, which could be useful to illustrate the development of a situation. A neme could also contain different files like a picture and an audio file. Neme compatible software, a neme browser, could display the portrait of a person and play an interview.

Files inside a neme can also refer to one another: For example a neme could contain a HTML file which displays two GIF pictures and allows to play an MP3 audio. This four files are all inside the neme, one after the other. The directory is part of the metadata section and created automatically, when the neme is built. So the HTML file now refers to files inside the neme and the neme browser displays them. Our example has the following structure:

$$N ::= (M \ I \ S)$$
$$I ::= i_1 i_2 i_3 i_4$$

or

$$N ::= (M \ i_1 i_2 i_3 i_4 \ S)$$

$i_1$  is the HTML file,  $i_2$  and  $i_3$  represent the GIFs and  $i_4$  is the mp3 sound bite.

The files inside the I section could be packed. In these case a neme would be like a ZIP archive with metadata and signatures. Like an 'intelligent ZIP'.

## S - Signature

To guarantee that a neme was really published by be the author in the byline, we work with hashes and private/public key-signatures. The author creates a hash value of the M and I sections and signs it with his or her private key. How does the reader get the public key?

The most obvious possibility is that the reader obtains the public key from a trusted source, a key server. Publishers for example may provide such a key server for their authors. Software companies may provide key servers for their users... and so on. The advantages and disadvantages of this concept are well known, we will not dive deeper into this issue now, because with nemes we have another interesting possibility to explore: generating reputation.

A reader could stumble across an article of an unknown writer and find it interesting, without knowing anything about the source. He searches the nets for further works of this writer. Let's assume these are about an issue our reader knows quite well, so he can judge that the author knows what he's writing about. The author gains reputation. To be sure that all articles are written by the same person, our reader has to check the signatures and therefore to obtain the public key. He has to find the right key server... does he?

Why not include the public key in every neme, as a part of the signature section? It's public anyway, so we should publish it, shouldn't we? It sounds a little bit strange at the beginning, because if we know the public key from the neme itself, it only proves that the neme was created by the person who created it – a tautology. But consider you have two nemes. You now can check whether these nemes were signed with *the same* private key or not. It tells you nothing about the author, but that's not necessary: anonymity might even be desired. As long as the author keeps his private key secret, you can identify his work.

Nemes may be signed with more than one signature. For example the author and his publisher may like to sign an article. Back to investigative journalist Greg P. He works for *The Paper*. If both, the author and the editor sign an article, the reader can be sure that this piece was approved by the paper. A reader who doesn't know Greg, but has subscribed to *The Paper* and thus gets the neme, can be confident that this article isn't written by a complete weirdo. The reader's software will take the public key of *The Paper* not from the neme itself, but have it stored locally, retrieved from the paper's key server on subscription.

If this reader finds more articles of Greg via this way, he might accept him as a trusted source. Now let's imagine, Greg himself reads a lot of nemes every day. He could sign those he finds interesting and send them to friends. The nemes would spread through the net. A couple of weeks later our reader, who now accepts Greg as a trusted source, might stumble across one of these articles. He doesn't know the author, but he recognizes

Greg's signature. The article is *recommended* by Greg P. If Greg recommends tons of junk, he will soon lose reputation. But if he recommends a lot of brilliant work, Greg's signature might become a quality mark. Young or unknown authors might even send their articles to him and ask him to sign them – to approve them... Greg now does the same thing, his publisher does with his articles.

Nemes can carry any number of signatures, very good articles can be signed by a lot of people. The original author is named in the metadata and this can't be changed without breaking the hash value and the signatures.

## Nemeplexes

You might wish to comment an article, before you sign it. No problem. Here we enter the realms of complex nemes or 'nemeplexes'. They offer some fantastic possibilities, but let's begin simple.

A neme is like a news atom. It contains M, I and S like protons, neutrons and electrons, but all together form an unit and can't be divided without being destroyed. But nemes can be combined to form bigger nemes – like molecules. For example someone might wish to compile all relevant nemes on a topic. Nemes are files and files can be put into the I section of a neme. If one puts three nemes  $N_1$ ,  $N_2$  and  $N_3$  into such a neme molecule or nemeplex,  $N_m$  would look like this:

$$N_m ::= (M_m I_m S_m)$$

or

$$N_m ::= (M_m N_1 N_2 N_3 S_m)$$

or

$$N_m ::= (M_m (M_1 I_1 S_1) (M_2 I_2 S_2) (M_3 I_3 S_3) S_m)$$

$I_1$ - $I_3$  may contain any combination of files.

Note that it is possible to break up the nemeplex and get three undamaged atomic nemes with intact signatures. So it's possible to download the compilation, and keep only one or two interesting nemes. These could then be integrated into new nemeplexes and still keep the signature of the original authors. The nemeplexes are signed by the person who compiles them. The metadata of a nemeplex should include some information of the metadata of the included nemes, to support search routines.

Nemeplexes are nemes themselves, so they can be combined to even bigger nemeplexes, just like Russian wood puppets. An editor could put all articles of the day into one nemeplex, all daily nemeplexes into a monthly one and compile twelve nemes in a whole-year-digest.

Or: An editor wants to compile an electronic book. He asks ten authors to write a chapter each. The authors create nemes, sign them and send them to the editor. He puts them in the right order into a nemeplex, signs this one and the book is finished.

### The evolution of nemes

So, what's about commenting on nemes? Well, it's easy: You write your comment into a file and put it together with the whole original neme into a nemeplex.

$$N_c ::= (M_c \ i_c N_1 \ S_c)$$

or

$$N_c ::= (M_c \ i_c (M_1 \ I_1 \ S_1) \ S_c)$$

This is, if you put the comment in front of the neme. Of course  $N_c ::= (M_c \ N_1 i_c \ S_c)$  would also be possible.

$i_c$  is the actual comment,  $N_1$  is the commented neme. Both together form the information sector of the new neme  $N_c$ . Your signature covers this whole nemeplex. If someone deletes or modifies your comment, your signature breaks. The inner neme  $N_1$  would not be damaged in this case. But if someone manipulates any part of  $N_1$ , of course both signatures  $S_1$  and  $S_c$  break.

### Everyones a publisher: nemezines

Now how could we make use of all these possibilities? Imagine an enhanced version of some p2p software, which includes a neme browser. And imagine a user called alex@nematic.net.

The neme browser allows Alex to define his favourite news sources – other users. The browser tries to find these publishers in the network every fifteen minutes, or every two

hours or every time its started. On successful connection, Alex' browser downloads all new nemes published on the local harddrives of the sources.

Alex is very interested in open source software. His favorite neme sources are:

- some linux distributors
  - a few coders of mozilla, open office and similar projects
  - lawrence@nematic.net, a lawyer publishing legal comments on copyright issues
  - the newsletter of the electronic frontier foundation
- and half a dozen others.

Nemes that are of no interest to Alex get deleted manually. The neme browser displays some metadata, so Alex doesn't have to read all nemes. But what about interesting articles? Shouldn't they be spread? Well, the neme browser has a great button: MyNemezine. Clicking in this one moves nemes from the inbox to a dedicated directory, adding Alex' signature automatically. Other people may download nemes from this folder.

With this simple feature Alex has become a publisher himself, without writing one simple line of text. While he's reading, he's compiling his own online magazine – *Alex's Open Nemezine*. If he is doing a good job, a lot of people may connect to his nemezine. These people do not have to wade through all nemes of all sources Alex is reading, they get only the things he considers interesting.

So Nadine, who is an activist against patents on life forms, but also interested in other patent related issues, connects to Alex. Nadine regularly moves all nemes she considers relevant into her dedicated folder, creating her *Stop BioPatents* Nemezine.

Peter, who is studying genetics, reads a lot of scientific nemezines as well as Nadine's. So a neme containing a legal comment written and originally published by Lawrence may wander via Alex and Nadine all the way to Peter – who would never consider Lawrence as a favorite source, because 99.9 percent of the copyright lawyers articles are of no interest for the biologist. But this nemes are made for spreading, and that is what they do...

The nematic net is no new p2p net, no new topology, no new routing algorithm and a nemezine is not a node in a certain technology. The nematic net is build by many different programs who support the neme filetype, can read the metadata and check the signatures. A neme is only a file and it may spread across all kinds of computer networks, hop from FastTrack to Freenet via Gnutella. Nemes may even spread through the world wide web.

## Nemzilla

Imagine someone writes a webbrowser, that ignores index.htm and checks first if there is a file called index.neme on the server.

index.neme is a neme containing index.htm and a signature. The browser checks the signature and if it's okay, the website is displayed as usual. If not, the browser gives a warning: „The signature is broken, this website may be hacked!“

How does the browser know, what the correct key for a website is? Two possibilities: It retrieves the key from a trusted source – or the key is already stored on the local drive, because the user visits this site regularly, his favorite online-newspaper, for example.

Websites could be uploaded in two different versions: a ‚normal‘, unprotected one and a secure nematic version, where every link leads to another neme. Or if that's too much fuss, webmasters could offer only pages containing crucial information additionally as nemes.

With a simple click the nemes can be saved to a dedicated folder on the users harddrive. If the user allows p2p software to access this folder, he feeds nemes into this nets just by surfing the web. If a newssite is taken down by a DoS attack... the backup system is already in place. And not only one, but many!

## Beyond the newswire

The ability of nemes to add metadata to different already widely used file formats may be used for other purposes than sharing news. Why not put mp3-songs inside nemes and make them easier to find?

Of course there exist various ideas how to add metadata to files, metadata that can be transferred from one p2p net to another. Nemes are probably not the most sophisticated solution to this problem, but one easy to use. Nemes and nemezines could bring interoperability between different nets through a back door. They could be NET MEMes, instead of mere news memes. Actually I suspect, that this might be a

Just think of it: we have dedicated file types for most applications. We have sophisticated formats for all kinds of data. .tiff for high quality pictures, .jpg for compressed ones, .gif for web purposes and dozens of others, and that are only those for pictures. Each of this

formats was developed to fulfill certain requirements. Sharing files is one of the most popular applications right now and there is no file format available that fulfills even the most basic needs like metadata.

mp3 was not developed for sharing music in large, anonymous and sometimes hostile networks. It's just a wonderful compression method that sparked the success of p2p. Now it's time to move forward.

So far, nemes are no more than an idea – the neme-meme. To make them real, the file format has to be defined in detail. That's the first step, and it should be done in an open way by an open community. Everyone's invited.