

Um Modelo para Navegação Web usando Nomeação Auto-Certificável e Publica/Assina

Daniel Lopes Fussia¹, Antonio Marcos Alberti¹

¹ICT Lab. – Instituto Nacional de Telecomunicações (Inatel) – Av. João de Camargo
510 - CEP 37.540-000 – Santa Rita do Sapucaí – MG – Brasil

{daniell, alberti}@inatel.br

Abstract. *This paper investigates a new model for world wide web navigation. The main difference is that instead of using protocols from current Internet architecture, i.e. HTTP/TCP/IP, we employ a future Internet initiative called NovaGenesis. NovaGenesis is being deployed since 2008 and currently has a C++ prototype for Linux. The proposed model employs self-certifying names instead of HTTP hyperlinks. Instead of client/server model, we employ a publish/subscribe approach. Web page contents are stored into network caches, improving performance. We report a successful implementation of NovaGenesis web through laboratory results that validate its advantages in terms of efficiency, flexibility and security.*

Resumo. *Este artigo investiga um novo modelo para navegação na world wide web. A principal diferença é que ao invés de utilizarmos a arquitetura de protocolos da Internet atual, isto é o HTTP/TCP/IP, utilizamos uma arquitetura de Internet do futuro chamada NovaGenesis. A NovaGenesis encontra-se em desenvolvimento desde 2008 e atualmente conta com um protótipo em C++ para o Linux. O modelo proposto utiliza nomes auto-certificáveis no lugar de hyperlinks HTTP. Ao invés do modelo cliente/servidor, utilizamos comunicação pública/assina. O conteúdo das páginas web é armazenado em caches de rede, melhorando o desempenho da transferência de páginas. Reportamos uma implementação do modelo de web NovaGenesis com resultados de laboratório que comprovam suas vantagens em termos de eficiência, flexibilidade e segurança.*

1. Introdução

A *world wide web* é talvez a principal aplicação da Internet. Inventada por Sir Tim Berners-Lee em 1989, a *web* evoluiu muito desde a sua criação. As melhorias cobrem o modelo de comunicação cliente/servidor, as linguagens de programação, a dinâmica de formulários e aplicações, a diversidade de conteúdos e formatos, dentre muitos outros aspectos. Com o advento das chamadas propostas de Internet do futuro, muitos se perguntam como ficará a *web* do futuro? Internet do futuro é o termo usado no meio acadêmico para denotar todos os projetos de evolução (ou revolução) da arquitetura da Internet. Alguns pesquisadores acreditam que a arquitetura da Internet do futuro será uma combinação de propostas evolucionárias, que continuam o processo de adaptação dos protocolos de Internet, tal qual o IPv6 (sem quebra de paradigma). Outros pesquisadores acreditam que as arquiteturas futuras de Internet serão folha em branco (*clean slate*), quebrando os paradigmas atuais e implementando novas e revolucionárias propostas. Neste artigo, reportamos os desafios e resultados obtidos na implementação de um modelo folha em

branco para a navegação *web*. Esse modelo apoia-se sobre a arquitetura *clean slate* chamada NovaGenesis. Propomos e testamos um navegador *web* que segue os paradigmas da NovaGenesis, dentre eles: modelo publica/assina com ciclo de vida integrado de serviços e conteúdos, espaços ilimitados de nomeação e resolução de nomes, nomeação auto-certificável, roteamento e encaminhamento de mensagens usando nomes, auto-certificáveis, armazenamento de conteúdos em *cache* de rede e operação baseada em contratos. Até onde sabemos não existe outro trabalho de *web* para Internet do futuro com essas características.

O modelo da *web* NovaGenesis consiste de: (i) um serviço que publica páginas *web* e seus conteúdos usando uma interface publica/assina distribuída – as páginas são armazenadas em *caches* de rede de forma distribuída usando estruturas de dados *hash table*; (ii) um browser usado para assinar páginas *web* e ligações entre nomes armazenados na rede. O navegador *web* é nomeado como NGBrowser, e é uma aplicação *Peer-to-Peer* (P2P) inserida no contexto da NovaGenesis, permitindo efetuar pesquisas e requisições de conteúdo, bem como páginas *web*, imagens, textos, entre outros objetos de informação. Tais objetos podem ter formatos binários, assim o usuário pode navegar pelo conteúdo com a mesma experiência da Internet atual. A Figura 1 ilustra a arquitetura desenvolvida neste artigo e seus principais serviços e comunicações entre processos.

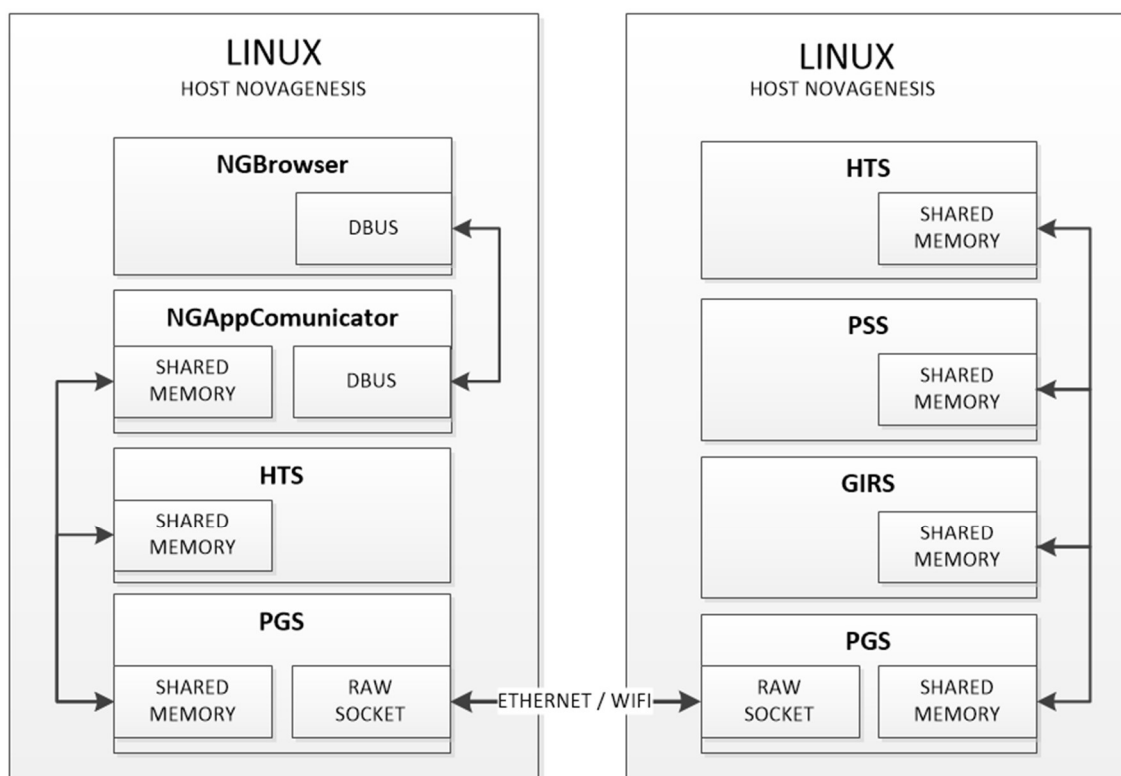


Figura 1 - Modelo da Arquitetura NGBrowser e NovaGenesis.

Diferente do modelo atual HTTP, a NovaGenesis utiliza o modelo publica/assina para o envio e obtenção dos dados na rede. Os *hosts*, processos e conteúdo são todos nomeados por um código *hash* (sequência binária fixada em 128 *bits*) e relacionados através de um grafo de ligações entre nomes que é armazenado de forma distribuída na rede.

Para a navegação *web* na arquitetura NovaGenesis foram desenvolvidas também ferramentas para adequação de páginas *web* já existentes para o novo modelo, chamadas de

NGConverter e NGAppPublisher. Seus papéis são respectivamente os de transformar um *site web* comum em um formato de *site web* NovaGenesis e publicá-los a rede para que o *site* seja posteriormente acessado pelo NGBrowser através de um serviço intermediário chamado NGAppComunicator. A Figura 2 ilustra como esses componentes interagem entre si.

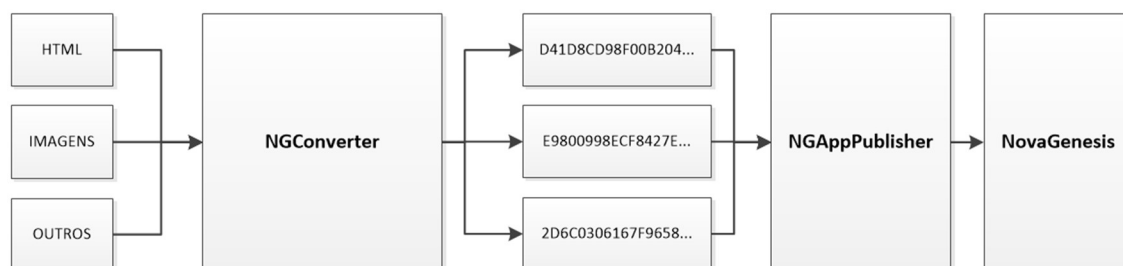


Figura 2 - Fluxo de entrada e saída da conversão e publicação de sites.

É esperado que o NGBrowser seja mais rápido e seguro que um navegador comum e que também faça uma economia de taxa de *bits*. No modelo proposto é possível obter um *hash* de qualquer conteúdo antes que seja efetuado o *download* do conteúdo. Desta forma, é possível se certificar de que o conteúdo solicitado é o mesmo que se encontra no *cache*, assim, os sistemas de *cache* serão mais eficientes descartando a necessidade de realizar um *download* de um arquivo já existente. Embora isso já exista na navegação *web* atual, na NovaGenesis o modelo publica/assina utiliza nomes auto-certificáveis, o que aumenta a segurança no acesso aos objetos de informação [Ghodsí e Koponen e Rajahalme e Sarolahti e Shenker 2013]. Ainda, somente conteúdos autorizados podem ser acessados através dos seus códigos *hash*. Assim, o navegador pode efetuar o *hash* do conteúdo e se certificar de que o conteúdo obtido foi o solicitado. O NGBrowser prova a possibilidade de uma navegação em uma Internet do futuro através do modelo publica/assina, podendo obter mais vantagens como a de possuir um *cache* mais eficiente e uma navegação mais segura do que o HTTP da Internet atual.

O restante desse artigo é organizado como segue. Na Seção 2 é feita uma descrição da arquitetura NovaGenesis e os seus subsídios para um novo modelo de *web*. Na Seção 3 apresentamos algumas outras tecnologias usadas na implementação do novo modelo *web* proposto, tais como *D-Bus* e Qt. Na Seção 4 apresentamos os novos serviços NovaGenesis criados para navegação *web* com nomeação auto-certificável, comunicação via publica/assina, roteamento baseado em nomes auto-certificáveis e *network caching*. Na Seção 5 reportamos os resultados dos testes feitos no ICT Lab. do Inatel. Por fim, na Seção 6 concluímos o artigo.

2. NovaGenesis

A NovaGenesis (NG) é uma arquitetura convergente que integra a troca, processamento e armazenamento de informações. Pode ser vista como uma arquitetura de Internet do futuro integrada a computação em nuvem. O projeto surgiu em 2008, mas foi em 2012 que houve a implementação de uma primeira prova de conceito. A NovaGenesis visa a integração dos princípios fundamentais de *design* que estão sendo considerados na Internet do futuro em um modelo único, convergente e auto similar. A NovaGenesis adota vários paradigmas como o de auto-organização de baixo para cima (de pequenos programas para grandes sistemas distribuídos), nomeação de serviços e conteúdos, exposição de

recursos, virtualização de substrato, nomes auto certificáveis (SCN - *Self-Certifying Names*), modelo de comunicação publica/assina, ciclo de vida de entidades física e virtuais, sejam processos, *hosts* ou objetos de informação.

A NovaGenesis não implementa qualquer protocolo TCP/IP ou aplicações de rede conhecidas, baseando-se atualmente em quatro processos principais: HTS, PSS, GIRS e PGS. O *hash table service* (HTS) é um resolvidor distribuído de nomes e *cache* de rede, capaz de resolver nomes entre qualquer *namespace* usado e armazenar ligações entre nomes, bem como conteúdos em tabelas *hash*. O *generic indirection resolution service* (GIRS) é responsável por escolher o HTS que irá armazenar uma dada ligação entre nomes, armazenando o endereço dos HTS que guardam determinados pares < chave, valor(es) >. O *publish/subscribe service* (PSS) implementa uma interface publica/assina distribuída usada pelos demais serviços NovaGenesis para disponibilizar ligações entre nomes e conteúdos na rede [Pedro e Junior e Amorim 2013], [Alberti e Fernandes e Casaroli e Oliveira e Júnior e Singh 2014]. Por fim, o *proxy/gateway service* (PGS) encapsula mensagens NovaGenesis diretamente sobre um *Network Socket* (NS) do tipo *Raw* que recebe e envia datagramas que não incluem cabeçalhos de *link*[die.net 1996], ou seja, não é necessário utilizar quaisquer protocolos da camada de rede já existentes. Tipicamente, as mensagens NovaGenesis são encapsuladas diretamente sobre tecnologias de enlace (tais com Ethernet e Wi-Fi). Porém, a versão corrente da arquitetura ainda carece de um protocolo de transporte como o TCP. A Figura 3 ilustra a arquitetura NovaGenesis para rede local.

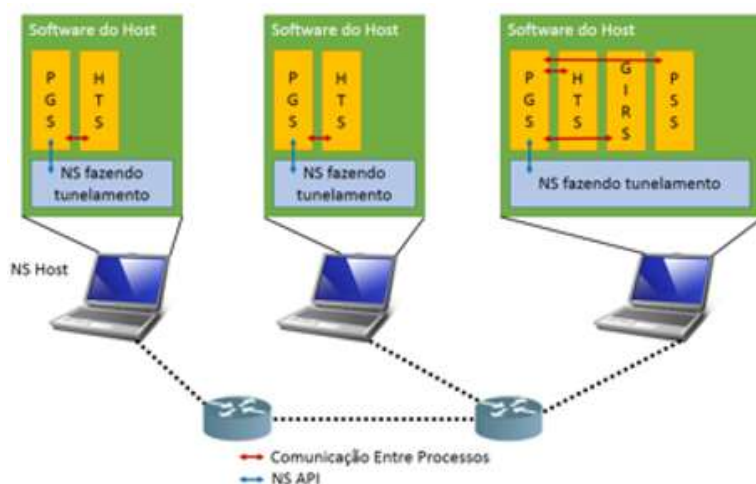


Figura 3 – Arquitetura NovaGenesis ilustrada para uma rede local.

2.1. Hash Table Service (HTS)

O serviço de tabela *hash* é o processo responsável por armazenar (em *cache*) conteúdos, informações sobre *hosts*, processos ou relacionamentos entre eles. Os relacionamentos são efetuados através de uma chave associada a vários valores, onde as chaves ou valores assumem uma sequência em *bits* no formato de *String* (sequência de caracteres terminado com '\0'). Cada valor pode ser associado a outras chaves que por sua vez possuem outros valores. Desta forma os relacionamentos podem ser comparados a grafos cíclicos. Os arquivos também podem fazer parte dessa associação, de forma indireta, na qual um arquivo é representado por um SCN obtido através do *hash* do próprio arquivo e o valor é o caminho onde o arquivo é armazenado no sistema operacional.

2.2. Generic Indirection Resolution Service (GIRS)

O serviço de resolução de indireções genéricas é um processo que seleciona qual HTS irá guardar uma ligação entre nomes no formato < chave, valor(es) >, tal como descrito na subsecção anterior. Seu papel é escolher a instância de HTS para cada par chave/valor, balanceando a carga, distribuindo igualmente as informações disponíveis entre as *hash tables*.

2.3. Publish/Subscribe Service (PSS)

O serviço de publica/assina é responsável por gerar permissões de acesso a ligações entre nomes publicadas. Portanto, ele faz o *rendezvous* entre serviços que publicam e assinam pares chave/valor. Todos os processos NovaGenesis devem utilizar o PSS para publicar/assinar qualquer conteúdo e/ou ligações entre nomes.

2.4. Proxy/Gateway Service (PGS)

O serviço *proxy/gateway* é o processo responsável por estabelecer a comunicação entre *hosts*. Cada PGS é executado em um *host* e efetua o protocolo de descobrimento da NovaGenesis para mapear e rotear dados entre eles utilizando as tecnologias de enlace.

3. NovaGenesis Web

Para uma navegação *web* mínima, considere que há uma comunicação ponto-a-ponto entre dois *hosts* diferentes. O cliente requisita o conteúdo desejado a um servidor e o servidor fornece o conteúdo ao cliente, que por sua vez exibe o conteúdo requisitado. Fazendo uma comparação com a Internet atual e a NovaGenesis, a Internet atual precisa dos seguintes componentes para esse cenário: servidor DNS para resolução de nomes em IPs, servidor de página (hospedagem), protocolo de comunicação entre o cliente e servidor (HTTP) para efetuar a entrega do conteúdo solicitado sobre o TCP/IP [Fielding and Reschke 2014]. Já a NovaGenesis utiliza dos processos PSS/GIRS/HTS para realizar as tarefas que seriam do DNS e servidor de página. O modelo publica/assina substitui o protocolo de comunicação (HTTP) para transferência de conteúdo. Na Internet atual, uma página *web* comum utiliza da linguagem de programação HTML para criar *sites* interativos que podem ser interpretados por navegadores [W3C 2014]. A linguagem de programação HTML não é fortemente ligada ao HTTP ou TCP, portanto ela foi adotada neste artigo para ser utilizada na NovaGenesis da mesma forma que é utilizada na Internet atual. Porém, os Identificadores Uniforme de Recurso (URI), utilizados para encontrar um recurso na Internet, são modificados para acomodar a arquitetura de rede adotada.

3.1. NGConverter

O NGConverter é uma ferramenta utilizada para converter um *site web* utilizado na Internet atual em uma página para ser utilizado na NovaGenesis. Esta conversão se resume em cinco etapas que são executadas em ordem conforme ilustra a Figura 5. Cada página *web* tem um descritor que é extraído da página e formatado em JSON, conforme ilustra a Figura 6. O principal objetivo do NGConverter é preparar o *site* de tal forma que simplifique o método de publicação executado pelo NGAppPublisher, discutindo logo em seguida. No final da execução desta ferramenta, é esperado que todos os arquivos do *site* utilizem apenas SCNs, bem como que todo o conteúdo de seus arquivos dependentes também esteja nomeado de forma auto-certificável. Ainda, é gerada uma lista de palavras

chaves de cada página *web*. Todos os arquivos são convertidos e armazenados em uma única pasta, mesmo que um *site* contenha subpastas. Também não existem arquivos com nomes diferentes para um mesmo conteúdo, somente um SCN é utilizado em diferentes *sites*, uma tremenda vantagem de se usar os SCNs.

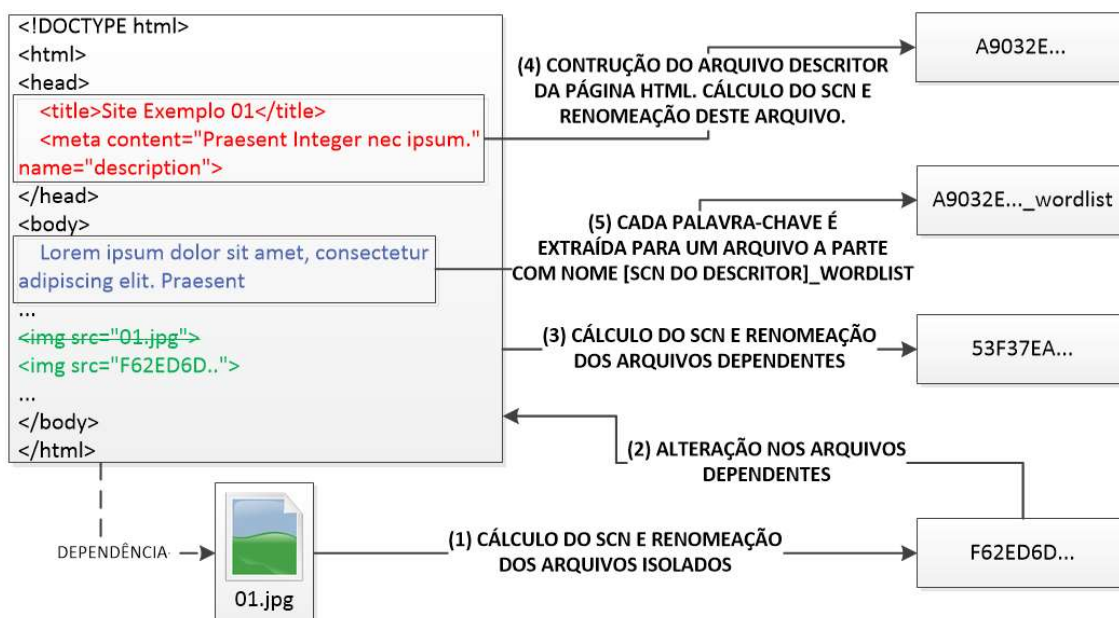


Figura 4 - Processo de conversão de site.

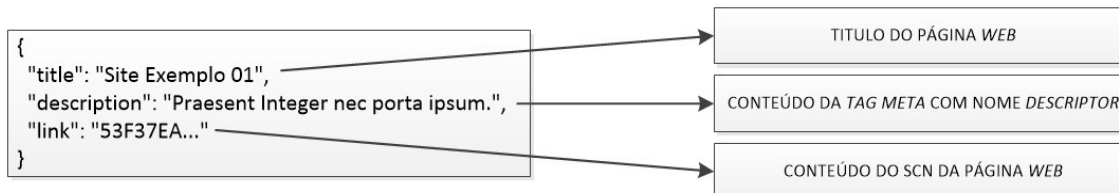


Figura 5 - Conteúdo de um descritor.

3.1.1 Exemplo de conversão de um único *site*

No lado esquerdo da Figura 7 demonstra-se um exemplo de uma estrutura de *site web* simples que poderia ser utilizado tanto na Internet atual, quanto em novas arquiteturas de rede. Esta estrutura serve de entrada para o NGConverter, que após convertido, resulta na parte direita da Figura 7. É esperado que após a conversão do *site*, esse tenha 2 arquivos a mais para cada arquivo HTML do *site* original. Eles representam o descritor e a lista de palavras-chaves do arquivo HTML. Em uma rápida análise a Figura 7, é possível observar o arquivo com extensão “_wordlist”, que representa a lista de palavras-chaves e o descritor deste HTML, representado pelo mesmo nome do arquivo da lista de palavras-chaves, porém sem a extensão “_wordlist”.

```
[fedora@localhost HTMLFiles]$ tree
.
├── site01
│   ├── 01.jpg
│   ├── 02.jpg
│   ├── 03.jpg
│   └── index.html
└──
1 directory, 4 files
[fedora@localhost HTMLFiles]$

[fedora@localhost NGFiles]$ tree
.
├── 09AA661FE5C84F646707B3B60ED673D5
├── 9789EF75DEAA6C6DB34C22EAA96D4A97
├── A1E0DBC95052460F910AC8EE7ECC3849
├── AA6C9E8353DDCE9E934EE2C77B7D4A72
├── AA6C9E8353DDCE9E934EE2C77B7D4A72_wordlist
└── D4CA335CAC00AEF6605746358777EAA6
0 directories, 6 files
[fedora@localhost NGFiles]$
```

Figura 6 - Demonstração do antes e depois da conversão do Site Exemplo 01.

A Figura 8 reporta as atividades realizadas pelo NGConverter durante a conversão do *Site Exemplo 01*. Através dele é possível notar as tarefas descritas na Figura 5, onde os arquivos isolados neste exemplo representam as imagens: 01.jpg, 02.jpg e 03.jpg.

```
[fedora@localhost HtmlToHash]$ ./NGConverter.sh
index: ./HtmlToHash/SiteFiles/NGFiles/site01/03.jpg => A1E0DBC95052460F910AC8EE7ECC3849
index: ./HtmlToHash/SiteFiles/NGFiles/site01/02.jpg => D4CA335CAC00AEF6605746358777EAA6
index: ./HtmlToHash/SiteFiles/NGFiles/site01/01.jpg => 09AA661FE5C84F646707B3B60ED673D5
replace: 03.jpg => A1E0DBC95052460F910AC8EE7ECC3849 at ./HtmlToHash/SiteFiles/NGFiles/site01/index.html
replace: 02.jpg => D4CA335CAC00AEF6605746358777EAA6 at ./HtmlToHash/SiteFiles/NGFiles/site01/index.html
replace: 01.jpg => 09AA661FE5C84F646707B3B60ED673D5 at ./HtmlToHash/SiteFiles/NGFiles/site01/index.html
index: ./HtmlToHash/SiteFiles/NGFiles/site01/index.html => 9789EF75DEAA6C6DB34C22EAA96D4A97
building-descriptor: 9789EF75DEAA6C6DB34C22EAA96D4A97(index.html)
index-meta: ./HtmlToHash/SiteFiles/NGFiles/site01/9789EF75DEAA6C6DB34C22EAA96D4A97_metafile => AA6C9E8353DDCE9E934EE2C77B7D4A72
index-wordlist: ./HtmlToHash/SiteFiles/NGFiles/site01/9789EF75DEAA6C6DB34C22EAA96D4A97_wordlist => AA6C9E8353DDCE9E934EE2C77B7D4A72_wordlist
moving: ./HtmlToHash/SiteFiles/NGFiles/site01/* => ./HtmlToHash/SiteFiles/NGFiles/site01/..
delete: ./HtmlToHash/SiteFiles/NGFiles/site01
[fedora@localhost HtmlToHash]$
```

Figura 7 – Conversão do Site Exemplo 01.

3.2. NGAppPublisher

O NGAppPublisher é um processo que possui os protocolos NovaGenesis e assim se comunica com outros processos da rede. Dessa forma é possível realizar as operações de publicação e assinatura de ligações entre nomes e/ou objetos de informação. Este processo é temporário e é executado somente com o objetivo de ler uma pasta com os arquivos preparados pelo NGConverter e publicá-los na rede NovaGenesis. A associação efetuada na publicação é simples, do qual cada SCN é associado ao arquivo respectivo arquivo. Isto, torna possível o acesso direto ao arquivo apenas utilizando o SCN. Outra etapa realizada, é a associação das palavras-chaves extraídas das páginas para o respectivo SCN do descritor. Desta forma, é possível buscar uma palavra-chave e descobrir qual página ela está inserida. Cada palavra-chave pode ser associada a mais de um descritor, resultando na descoberta de mais de uma página que está inserida a mesma palavra-chave, por ex.: comparado ao sistema de busca do Google que retorna ao usuário as páginas encontradas através da palavra-chave buscada.

3.3. NGAppCommunicator

O NGAppCommunicator é um processo que também possui os protocolos da NovaGenesis e assim como o NGAppPublisher também pode realizar as operações de publicação/assinatura. A diferença é que esse é executado durante o tempo em que o navegador *web* é usado. Sua finalidade é expor uma API de acesso simples ao NGBrowser, retirando toda a complexidade do protocolo na aplicação visual. A exposição desta API é efetuada através do *D-Bus*, portando mais de uma aplicação visual diferente pode utilizar essa funcionalidade. O principal objetivo do NGAppCommunicator é intermediar a comunicação entre a aplicação visual e a arquitetura NovaGenesis, abstraindo funcionalidades simples e usuais as diferentes aplicações visuais com essa finalidade. Neste artigo, dois métodos foram criados: (i) o SearchByMurmur para solicitar acesso direto ao conteúdo; e o (ii) SearchByLiteral para solicitar todos os descritores de uma ou mais palavras-chaves requisitadas pelo usuário. Um método deve ser implementado no requisitante – chamado de “*complete*” – para receber uma lista de SCNs como resposta ao invocar os métodos SearchByLiteral ou SearchByMurmur. Os parâmetros para as chamadas aos métodos SearchByLiteral e SearchByMurmur, bem como o retorno através do “*complete*” são formatados usando JSON com um *array* de *strings*.

O NGAppCommunicator implementa um sistema de *cache* que verifica a existência de arquivos em disco local para efetuar o pedido de arquivos a rede NovaGenesis. Esta funcionalidade se baseia da seguinte forma: quando uma lista de SCNs é recebida como resultado de qualquer busca, esta lista é comparada com a que existe em disco. Caso exista, não é efetuada a assinatura do arquivo. O mesmo acontece quando ocorre um pedido de assinatura de um arquivo através do *browser*, sendo que ele já foi assinado anteriormente. Também implementa o sistema de segurança, em que é checado o SCN do conteúdo com o SCN solicitante, caso sejam incoerentes o arquivo é descartado.

3.4. NGBrowser

O NGBrowser é um *software* aplicativo visual que o usuário utiliza para navegar na *web* NovaGenesis. A comunicação com a NovaGenesis é realizada através do NGAppCommunicator que emprega *D-Bus*. As URLs de acesso ao conteúdo definem o processo decisório que escolhe qual método do NGAppCommunicator será chamado. Neste artigo, foram definidas somente dois tipos de URLs, as que começam com “*ngs://*” (NovaGenesis *Search*) e as que começam com “*ngu://*” (NovaGenesis *Unique*), outros tipos serão definidos futuramente para agregar várias funcionalidades ao usuário. O NGBrowser implementa a chamada recursiva de dados dependentes para exibir a página, por ex.: caso um HTML dependa de um arquivo CSS, é efetuada uma assinatura do conteúdo CSS através da URL NGU e se houver uma imagem atribuída dentro deste assinado, uma nova assinatura desta imagem será requisitada novamente através da URL NGU. As chamadas recursivas são identificadas somente após o *download* do conteúdo ser concluído.

3.4.1 NovaGenesis Search (*ngs://*)

As URLs do tipo “*ngs://*” são utilizadas para pesquisa e são compostas por palavras-chaves em linguagem natural podendo conter espaço entre elas. As palavras-chaves são escolhidas pelo usuário com o desejo de encontrar páginas *web* que as contenham. Exemplo: “*ngs://carro vermelho*”, traz como resultados páginas que contenham as palavras-

chaves carro ou vermelho. Na versão utilizada pelo artigo não existe uma ordem de classificação para exibir os resultados encontrados. Estas URLs invocam o método Search-ByLiteral do NGAppComunicator, que quando finalizado, invoca o método “complete”, passando uma lista de SCNs dos descritores das páginas. O navegador por sua vez abre cada descritor e formata o conteúdo para uma página comum, exibindo em cada linha o título, descrição da página *web* e um link de acesso a esta página encontrada.

3.4.2 NovaGenesis Unique (ngu://)

As URLs do tipo “ngu://” são utilizadas para requisitar um único conteúdo da rede, seja uma foto, página, *scripts*, *stylesheet*, etc. Qualquer conteúdo em formato de arquivo. O método “ngu://” é uma URL que referência a um SCN. É uma URL geralmente utilizada para um acesso direto ao arquivo, podendo ser usada em *tags* HTML como “” ou “” que exigem em seus parâmetros *src* ou *href*, respectivamente. Por definição, as *tags* que requisitam um conteúdo direto não necessitam da palavra “ngu://”, somente o SCN já é o suficiente para o NGBrowser identificar a requisição de único arquivo. O método “ngu://” pode ser digitado através da barra de endereços do navegador, ex.: “ngu://EC9E8A5F6354AD70812196ABB89ECAEB”, que carrega o conteúdo por trás desse código *hash* hexadecimal.

4. Resultados Experimentais

A arquitetura utilizada para este experimento foi a mesma ilustrada na Figura 1, onde o meio de comunicação adotado foi a tecnologia *Ethernet*. Foram construídos 11 *sites* com uma página contendo palavras aleatórias e 3 fotos. Cada *site* teve 2 fotos repetidas dos outros 10 *sites*, seguindo a mesma estrutura do tópico “4.1.1”. Os próximos tópicos estão relacionados a preparação das páginas *web*, inicialização da NovaGenesis, publicação das páginas *web* e por fim a navegação utilizando as URLs “ngs://” e “ngu://”.

4.1 Preparação das páginas *web*

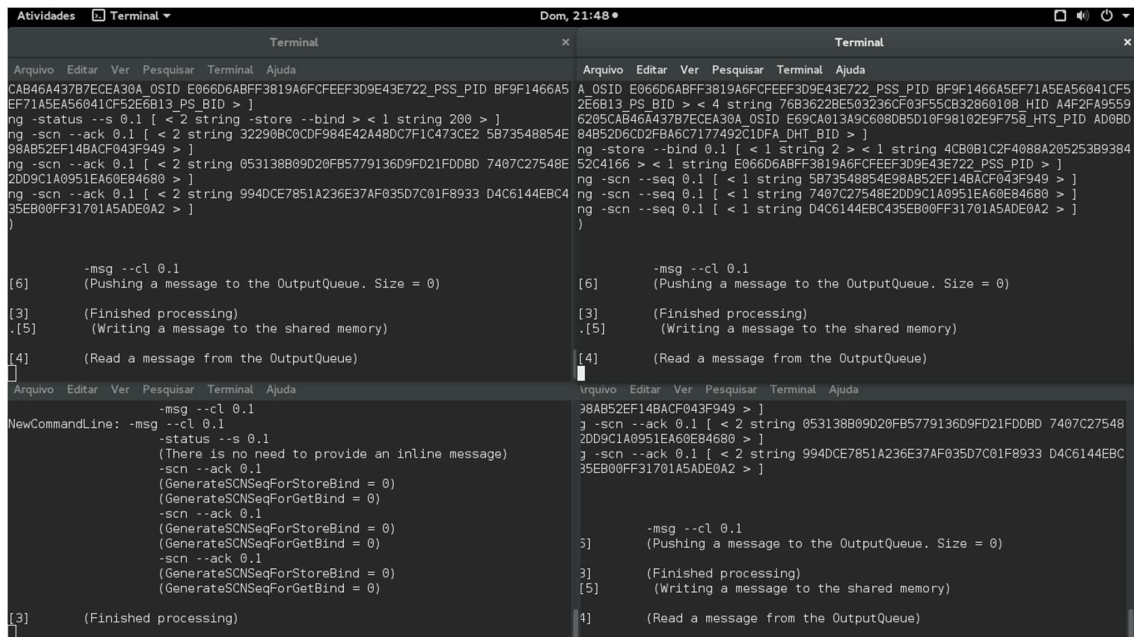
Com os *sites* de exemplo construídos, inicia-se a conversão através da ferramenta NGConverter. É esperado que tanto a pasta de origem quanto a pasta resultante da conversão contenham 45 arquivos no total. Pois, o conversor irá retirar 2 arquivos duplicados de cada *site* e gerar mais 2 arquivos de cada HTML, totalizando a mesma quantidade. A Figura 9 demonstra a pasta resultante da conversão.

```
[fedora@localhost NGFiles]$ ls
09AA661FE5C84F646707B3B60ED673D5 6D6D20BE51BEB54C11E65DEEF2FF9A7 _wordlist AA6C9E8353DDCE9E934EE2C77B7D4A72 _wordlist
08CE2FE16AA7672442E512FD6E916C80 6E85E2B913D720870CFD3EC062A03F44 B5126FF4D7B6B68EF7742C62F3425714
165E75746BCCF09B8A7A642C0C5B80E2 770516326045414C546BC749AB606B18 BF2CCF196994DC679D8738B9DFED3C10
1714334085B9CFA5568A1A51D1AF22F 770516326045414C546BC749AB606B18 _wordlist C1531F43B9614AA701DCCCE9005B6670E
1901D6A78A566307854C48034EFC0232 7AC9FD032A95B24DACC8195B180986BE C268098BCA38F4A931F08B087E561A16
1A1304B6422F1B5552B986C620881873 7AC9FD032A95B24DACC8195B180986BE _wordlist C268098BCA38F4A931F08B087E561A16 _wordlist
1A1304B6422F1B5552B986C620881873 _wordlist 8323387C1F15CE9F5B2B7641C0C95090 CB8D5B23A0F9265053FF6443AB22C5C2
248729544FCFA33366A812F638929310 8D0C6C88974868EF41EC3D54EBFA0397 CE58C9AF2DC32A87EA63EBD8809B8B9
36790CE5E0F96B3684D1B5CFE081B590 9789EF75DEAA6C6DB34C22EAA96D4A97 CE58C9AF2DC32A87EA63EBD8809B8B9 _wordlist
41267271432A3054FBF5A552C8495300 A11EBEE0D02B627C445F5F89D8E2C5EE D4CA335CAC00AEF6605746358777EAA6
55EEFD6DA282317A24FB805225980DAB A1E0DBC95052460F910ACBEE7ECC3849 D861C0EFE444E9113992962F81BAFC72
575B5B52C48F73EBBF8F9DD1094391C5 A7AF50D17127D19D386CA68680685EB8 DF0069473EB0FE0468F5D7214B850480
5E8199D17420DB3D315A4E42483FD0CF A86EB041E0772B80618DC30C722A36FB DF0069473EB0FE0468F5D7214B850480 _wordlist
5E8199D17420DB3D315A4E42483FD0CF _wordlist A86EB041E0772B80618DC30C722A36FB _wordlist EC77EEEA4FB2D4EFA19151178B74B525
6D6D20BE51BEB54C11E65DEEF2FF9A7 AA6C9E8353DDCE9E934EE2C77B7D4A72 EC77EEEA4FB2D4EFA19151178B74B525 _wordlist
[fedora@localhost NGFiles]$ ls | wc -l
45
[fedora@localhost NGFiles]$
```

Figura 8 - Resultado da conversão dos 11 *sites*.

4.2 Inicialização da NovaGenesis

Para que a NovaGenesis esteja funcionando em dois computadores diferentes, primeiro é necessário executar seus principais processos no *host* que assumirá o papel de servidor de páginas. Desta forma os processos estarão aptos a se comunicarem entre si, inclusive pelas interfaces de rede. Logo em seguida, em outro *host*, já se pode iniciar o *host* cliente que se conecta automaticamente a NovaGenesis. A Figura 10 demonstra os quatro principais processos da NovaGenesis em execução: PGS, GIRS, PSS, HTS da esquerda para a direita e de cima para baixo. Nesta tela, eles aguardam o cliente se conectar.



```
Terminal 1 (PGS):
CAB46A437B7E3A OSID E066D6ABFF3819A6FCFEF309E43E722 PSS_PID BF9F1466A5
EF71A5EA56041CF52E6B13 PS BID > ]
ng -status --s 0.1 [ < 2 string -store --bind > < 1 string 200 > ]
ng -scn --ack 0.1 [ < 2 string 32290BC0CF984E42A48DC7F1C473CE2 5B73548854E
98AB52EF14BACF043F949 > ]
ng -scn --ack 0.1 [ < 2 string 053138B09D20FB5779136D9FD21FDD0B 7407C27548E
2DD9C1A0951EA60E84680 > ]
ng -scn --ack 0.1 [ < 2 string 994DCE7851A236E37AF035D7C01F8933 D4C6144EBC
35EB00FF31701A5ADE0A2 > ]
]

Terminal 2 (GIRS):
A_OSID E066D6ABFF3819A6FCFEF309E43E722 PSS_PID BF9F1466A5EF71A5EA56041CF5
2E6B13 PS BID > < 4 string 768362226553236CF03F55CB32860108_HID A4F2FA9559
6205CAB46A437B7E3A OSID E69CA013A9C080B5D10F98102E9F758 HTS_PID AD6BD
84B5206CD2FBA6C7177492C1DFA DHT BID > ]
ng -store --bind 0.1 [ < 1 string 2 > < 1 string 4CB081C2F4088A205253B9384
52C4166 > < 1 string E066D6ABFF3819A6FCFEF309E43E722 PSS_PID > ]
ng -scn --seq 0.1 [ < 1 string 5B73548854E98AB52EF14BACF043F949 > ]
ng -scn --seq 0.1 [ < 1 string 7407C27548E2DD9C1A0951EA60E84680 > ]
ng -scn --seq 0.1 [ < 1 string D4C6144EBC35EB00FF31701A5ADE0A2 > ]
]

Terminal 3 (PSS):
-msg --cl 0.1
(Pushing a message to the OutputQueue. Size = 0)
[6]
(Finished processing)
[3]
(Writing a message to the shared memory)
.[5]
(Read a message from the OutputQueue)
[4]

Terminal 4 (HTS):
-msg --cl 0.1
(There is no need to provide an inline message)
-status --s 0.1
(GenerateSCNSeqForStoreBind = 0)
(GenerateSCNSeqForGetBind = 0)
-scn --ack 0.1
(GenerateSCNSeqForStoreBind = 0)
(GenerateSCNSeqForGetBind = 0)
-scn --ack 0.1
(GenerateSCNSeqForStoreBind = 0)
(GenerateSCNSeqForGetBind = 0)
[3]
(Finished processing)
```

Figura 9 - Quatro principais processos da NovaGenesis.

4.3 Publicação das páginas web

Toda a publicação é efetuada através do NGAppPublisher que se conecta à rede NovaGenesis. O processo NGAppPublisher utiliza os arquivos convertidos pelo NGConverter para iniciar as publicações. Na solução proposta nesse artigo, as páginas são publicadas no *cache* de rede da NovaGenesis (chamado HTS), ao invés de ficarem armazenadas em servidores *Web* acessíveis via HTTP. A Figura 11 reporta o processo NGAppPublisher comunicando-se com os outros processos NovaGenesis. A Figura 12 reporta a publicação da foto “02.jpg” do *site* Exemplo 01. A Figura 13 reporta a associação das palavras-chaves “vel”, “nunc”, “turpis”, “condimentum” e “pretium” ao descritor do *site* Exemplo 01 com SCN “AA6C9E8353DDCE9E934EE2C77B7D4A72”. Após o termino das publicações e o encerramento do processo NGAppPublisher, o HTS contém todos os arquivos dos *sites* gerados. Então, a Figura 14 reporta os arquivos publicados. Nota-se que o número de arquivos publicados (35 arquivos) é menor do que quando se considera os arquivos temporários, pois todos os arquivos com extensão “_wordlist” não foram publicados, mas sim lidos para criar a associação entre a palavra-chave e o descritor da página HTML em questão.

```

Arquivo Editar Ver Pesquisar Terminal Ajuda
> < 1 string 7B249B2B1577D901D2EA9C3065D5AD97_Core_BID > ]
ng -pub --bind 0.1 [ < 1 string 2 > < 1 string 55EEFD6DA282317A24FBB05225980DAB
> < 1 string 64EA88E95728B6E95E4ACEEDAE1FECD9_App_PID > ]
ng -pub --bind 0.1 [ < 1 string 2 > < 1 string 55EEFD6DA282317A24FBB05225980DAB
> < 1 string A4F2FA95596205CAB46A437B7ECEA30A_OSID > ]
ng -pub --bind 0.1 [ < 1 string 9 > < 1 string 55EEFD6DA282317A24FBB05225980DAB
> < 1 string 76B3622BE503236CF03F55CB32860108_HID > ]
ng -message --type 0.1 [ < 1 string 1 > ]
ng -message --seq 0.1 [ < 1 string 42 > ]
ng -scn --seq 0.1 [ < 1 string 7BC4D0B3878B525845DF8F5BF5D03723 > ]

There is a payload of 10385 bytes
)

[6]      -msg --cl 0.1
        (Pushing a message to the OutputQueue. Size = 0)

[3]      (Finished processing)
.        (Waiting 2.443373076 sec)
.....[5]      (Writing a message to the shared memory)

[4]      (Read a message from the OutputQueue)

```

Figura 10 – Log de execução do NGAppPublisher.

```

ng -store --bind 0.1 [ < 1 string 2 > < 1 string D4CA335CAC00AEF6605746358777EAA
6 > < 1 string D4CA335CAC00AEF6605746358777EAA6 > ]
ng -info --payload 0.1 [ < 1 string D4CA335CAC00AEF6605746358777EAA6 > ]

```

Figura 11 - Mensagem NovaGenesis de publicação de arquivo.

```

ng -pub --bind 0.1 [ < 1 string 2 > < 1 string vel > < 1 string AA6C9E8353DDCE9E
934EE2C77B7D4A72 > ]
ng -pub --bind 0.1 [ < 1 string 2 > < 1 string nunc > < 1 string AA6C9E8353DDCE9
E934EE2C77B7D4A72 > ]
ng -pub --bind 0.1 [ < 1 string 2 > < 1 string turpis > < 1 string AA6C9E8353DDC
E9E934EE2C77B7D4A72 > ]
ng -pub --bind 0.1 [ < 1 string 2 > < 1 string condimentum > < 1 string AA6C9E83
53DDCE9E934EE2C77B7D4A72 > ]
ng -pub --bind 0.1 [ < 1 string 2 > < 1 string pretium > < 1 string AA6C9E8353DD
CE9E934EE2C77B7D4A72 > ]

```

Figura 12 - Associação de palavras-chave a um descritor.

```

[fedora@localhost HTS]$ ls
09AA661FE5C84F64670783B60ED673D5  55EEFD6DA282317A24FBB05225980DAB  9789EF75DEAA6C6DB34C22EAA96D4A97  C268098BCA38F4A931F0BB087E561A16
08CE2FE16AA7672442E512FD6E916C80  57585852C48F73EBB8BF9DD1094391C5  A11EBEE0DD2B627C445F5F89D8E2C5EE  CBB05B23A0F9265053FF6443AB22C5C2
165E75746BC0CF0988A7A642C0C588DE2  5E8199D17420DB3D315A4E42483FD0CF  A1E0D8C905052460F910AC8EE7ECC3849  CE58C9AF2DC32A87EA63E8D88098BB9
1714334D85B9CFEA5568A1A51D1AF22F  6D6D620BE51BE54C11E65D0EEF2FF9A7  A7AF50D17127D19D386CA686806B5E88  D4CA335CAC00AEF6605746358777EAA6
1901D6A7BA566307854C48034EFC9232  6E85E2B913D720870CFD3EC062A03F44  A86EB041E0772BB0618DC30C722A36FB  D861C0EFE444E9113992962F81BAFC72
1A1304B6422F1B5552B986C620881873  770516326045414C546BC749AB606B18  AA6C9E8353DDCE9E934EE2C77B7D4A72  DF0069473EB0FE0468F5D72148850480
248729544FCFA33366A812F63B929310  7AC9FD032A95824DACC8195B180986BE  B5126FF4D7B6B68EF7742C62F3425714  EC77EEEA4FB2D4EFA19151178B748525
3679DEC5E0F96B36B4D1B5CFE081B590  8323387C1F15CE9FB52B7641C0C95090  BF2CCF196994DC679D873889DFED3C10  HTS.ini
41267271432A3054FBF5A552C8495300  8DDC0C88974B68EF41EC3D54EBFA0397  C1531F43B9614AA701DCCE900586070E
[fedora@localhost HTS]$ ls | wc -l
35

```

Figura 13 - HTS após o término das publicações.

4.4 Navegação

Para iniciar a navegação na NovaGenesis é preciso iniciar o processo NGAppCommunicator que irá intermediar a comunicação entre o NGBrowser e a NovaGenesis. Quando o processo estiver concluído, será exibida a mensagem: “*Buffer Empty, nothing to subscribe*”. A Figura 15 demonstra essa imagem.

```
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda
ng -msg --cl 0.1 [ < 1 string 3028B54759987327147FD54D3770C558 App2_PID > < 1 st
ring DF566188EDB1621D98E4AD521B9C2E1D_Core_BID > < 1 string DF566188EDB1621D98E4
AD521B9C2E1D_Core_BID > ]
ng -run --evaluate 0.1 [ ]
ng -scn --seq 0.1 [ < 1 string E2C42A0013E5BE8DDF94ABF36F335943 > ]
)

-msg --cl 0.1

-msg --cl 0.1
NewCommandLine: -msg --cl 0.1
-run --evaluate 0.1
(The app is already aware of this PSS)
(Try to Run expose files)
(Buffer EMPTY, nothing to subscribe)
(Subscribing Checking: Phase 2)
-scn --seq 0.1

[3] (Finished processing)
```

Figura 14 - NGAppCommunicator em espera por navegadores.

Logo, o NGBrowser é iniciado e os testes são realizados. Inicia-se por uma pesquisa pela palavra-chave “01” com a intenção de encontrar o “*Site Exemplo 01*” e depois a pesquisa pelas palavras-chaves “ipsum vac”, para encontrar todos os *sites* publicados. As Figuras 16 e 17 demonstram o resultado das pesquisas. É possível observar o NGAppCommunicator exibindo uma nota de que os arquivos foram assinados “*WORD was subscribed*”. Em ambos os testes, as exibições dos resultados aconteceram de forma correta no NGBrowser.

```
NewCommandLine: -msg --cl 0.1
-run --evaluate 0.1
(The app is already aware of this PSS)
(Try to Run expose files)
(WORD was subscribed)
(Subscribing Checking: Phase 2)
-scn --seq 0.1
```

Figura 15 - Assinatura encontrada e obtida com sucesso.

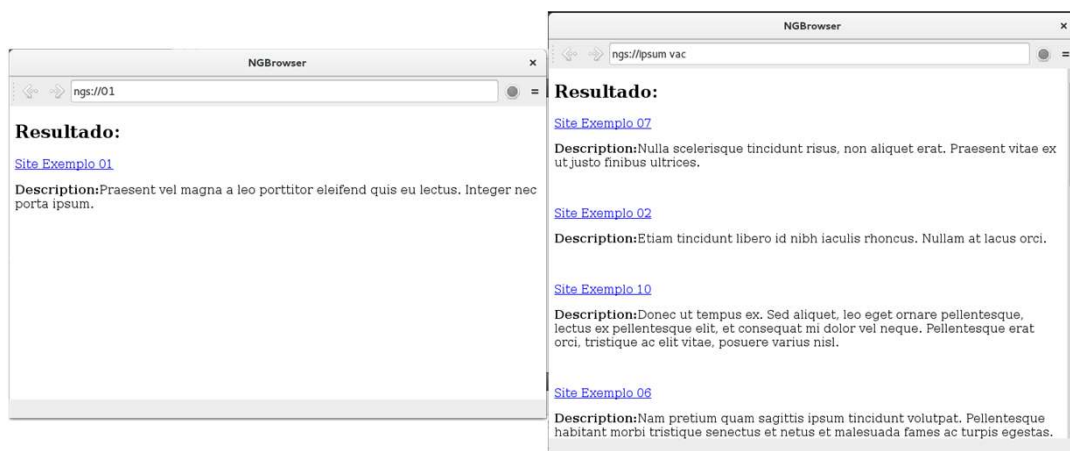


Figura 16 - Resultado do NGS de um *site* (esquerda) e de vários *sites* (direita).

Em seguida, a Figura 18 reporta outros dois testes. Primeiro, com um clique sobre o *link* de um *site* exemplo resultado do NGS e segundo a exibição de um conteúdo através da URL NGU.



Figura 17 - Exibição da página HTML (esquerda) e o acesso com o NGU (direita).

4.5. Eficiência do *Cache*

Para validar o ganho de eficiência do *cache* NovaGenesis, foi realizada uma comparação entre o NGBrowser e um navegador comum, ignorando o *overhead* de ambos os protocolos das redes. A Figura 19 representa o acumulo em *bytes* trafegados começando do *site* 1 até 11.

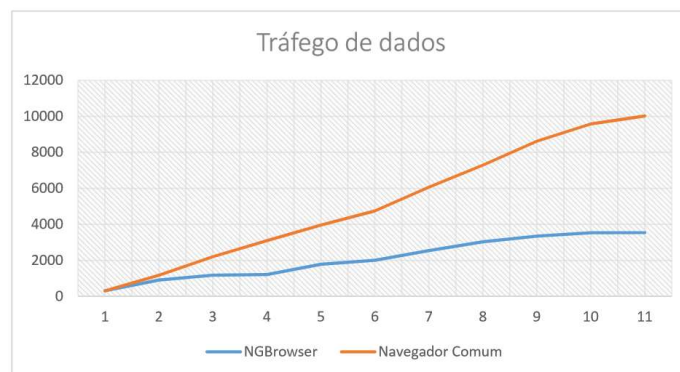


Figura 18 - Eficiência do *cache* do NGBrowser.

5. Conclusão

Este artigo reporta que é possível haver a navegação através de *websites* utilizando o modelo publica/assina. Este modelo se mostrou eficaz, dispensando a necessidade da implementação do protocolo HTTP. Ferramentas podem ser desenvolvidas para facilitar a migração de *websites* de forma automática. O NGAppCommunicator é um processo que pode servir de exemplo para outros projetos de Internet do Futuro com o âmbito de fornecer facilidades de acesso a diferentes redes através de *APIs*. Dentre as vantagens, estão a redução da complexidade de implementações de acesso a rede que os navegadores possuem e compartilhamento de um mesmo sistema de *cache*.

O uso de nomeação auto-certificável torna o sistema de *cache* eficiente devido a possibilidade de consultar conteúdos e certificá-los mesmo antes de realizar um *download*. Desta

forma é possível obter uma economia de dados significativa, conforme reporta a Figura 19. Nela constata-se uma economia de aproximadamente 6 *KBytes*. Este valor depende da quantidade de repetições de conteúdo copiadas em diferentes *sites*. O sistema de *cache* dos navegadores atuais só são eficientes caso o conteúdo que os veiculam tenham a mesma URL [Ilya 2014]. Os nomes auto-certificáveis ainda permitem que os dados sejam transferidos com verificação de integridade, pois conteúdos que tenham códigos *hash* diferentes são descartados pelo NGAppCommunicator. Esse artigo contribuí com a comunidade mediante a implementação deste NGBrowser. Futuramente, novos recursos serão implementados nesta aplicação para que ela alcance todas funcionalidades existentes hoje na Internet atual. Também, serão feitos testes em escalas maiores para aprofundar a análise de desempenho.

6. Agradecimentos

Este trabalho foi parcialmente financiado pela Finep, com recursos do Funttel, contrato No 01.14.0231.00, sob o projeto Centro de Referência em Radiocomunicações (CRR) do Instituto Nacional de Telecomunicações, Inatel, Brasil.

7. Referências

- Pedro, E. e Junior, F. e Amorim, G. (2013), “Estudo e Análise da Proposta NovaGenesis para a Internet do Futuro”, NovaGenesis, Trabalho de Conclusão de Curso, p. 42-77.
- Alberti, A. e Fernandes, V. e Casaroli, M. e Oliveira, L. e Júnior, F. e Singh, D. (2014), “A NovaGenesis Proxy/Gateway/Controller for OpenFlow Software Defined Networks”, NovaGenesis Architecture, p. 2-4.
- Fielding, Reschke (2014) “HTTP/1.1 Message Syntax and Routing”, <http://tools.ietf.org/html/rfc7230#page-5>, Março 2016.
- W3C (2014) “HTML5”, <https://www.w3.org/TR/html/introduction.html#a-quick-introduction-to-html>, Março 2016.
- Ilya, G. (2014) “Armazenar HTTP em cache”, <https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching>, Março 2016.
- die.net (1996) “Linux man page”, <http://linux.die.net/man/7/raw>, Março 2016.
- Ghods, A. e Koponen, T. e Rajahalme, J. e Sarolahti, P. e Shenker, S. (2013), “Naming in Content-Oriented Architectures”, Introduction, p. 1-2.