Rede IoT Colaborativa NovaGenesis

Vaner J. Magalhães, Gabriel Scarpioni e Antônio M. Alberti

***Resumo*— Este artigo apresenta uma aplicação do modelo NovaGenesis de Internet do Futuro em uma Rede Colaborativa de dispositivos IoT. Os dispositivos e o núcleo da rede NovaGenesis foram emulados para permitir a coleta de mensagens. Analisando as mensagens obtivemos uma visão de como é construída uma rede de dispositivos IoT onde um nó pode ajudar um outro nó na comunicação com o núcleo da rede, permitido a publicação dos dados sensoriados de forma colaborativa. Mostramos que é possível aplicar as melhorias e novidades de Internet do Futuro em aplicações de IoT reais e complexas.**

***Abstract*— This article presents an application of the NovaGenesis Future Internet model in a collaborative network for Internet of Things devices. All devices and the NovaGenesis network core were defined by software in order to collect all exchanged messages. Analyzing the messages we obtained a vision of how a network of IoT devices is built, where one node can help another node in its communication with the network core, allowing the publication of sensed data in a collaborative way. We show that it is possible to apply the proposed improvements and ideas of Future Internet in real and complex IoT applications.**

I. Introdução

Nos dias de hoje a Internet é utilizada para conectar pessoas, máquinas, prover entretenimento e notícias, etc. Além disso, o número de dispositivos conectados está crescendo rapidamente, o que é chamado de Internet das Coisas. O modelo atual sofre com limitações para atender estes dispositivos, que na maior parte das vezes tem capacidades de memória, processamento e energia restritos, possuem mobilidade e comunicam-se por rádio frequência.

Para satisfazer este novo perfil uso, a Internet atual está passando por aplicação de medidas paliativas de expansão e adaptação, principalmente para prover melhorias em segurança, endereçamento, uso das capacidades computacionais e de energia, mobilidade, escalabilidade, entre outros fatores.

O termo Internet do Futuro foi criado devido a esta necessidade de mudança na Internet [1], [2]. Alguns pesquisadores defendem uma nova Internet evolucionária, onde o modelo atual é evoluído para atender as demandas futuras. Outros pesquisadores clamam por um modelo revolucionário, onde a Internet deve ser recriada do zero, utilizando as tecnologias e conhecimentos atuais.

Neste artigo utilizaremos o modelo de Internet do Futuro NovaGenesis [3], [4]. A NovaGenesis tem uma abordagem revolucionária, redesenhando a Internet a partir da integração de vários ingredientes contemporâneos. Utilizaremos este modelo em uma rede de Internet das Coisas (IoT - *Internet of Things*) [4] para obtermos uma rede de dispositivos confiável, dinâmica, escalável e cooperativa. A NovaGenesis provê melhorias em segurança, endereçamento baseado em nomes, serviços baseados em contratos.

V. J. Magalhães, G. Scarpioni e A. M. Alberti são membros do ICT Lab., Instituto Nacional de Telecomunicações (Inatel), CP 05, Santa Rita do Sapucaí, Minas Gerais, Brasil. E-mails: vaner, gabrield, alberti@inatel.br. 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. Os autores também gostariam de agradecer a FINATEL, FAPEMIG, CNPq e CAPES.

II. Modelo NovaGenesis

NovaGenesis (NG) é um projeto de Internet do Futuro que segue a abordagem revolucionária [3], [4], visando aplicar as melhores tecnologias atuais para se obter uma nova Internet. Todas as entidades que pertencem ao ecossistema NG são nomeadas. Uma entidade pode ser qualquer coisa, por exemplo um *hardware*, um código fonte, um nome de arquivo, uma foto, um serviço, etc. Para facilitar a nomeação, a NG faz uso de Nomes Autoverificáveis (SVN - *Self-Verified Names*), que é um número binário (atualmente 32 bits) gerado pela passagem de uma característica da entidade por uma função *Hash*. O processamento e a troca das informações são feitos baseados nestes SVNes. Todas as entidades que pertencem ao ecossistema são relacionadas através da ligação de seus nomes, criando assim um grafo de nomes ligados. Estas ligações são armazenadas em Tabelas *Hash* Distribuidas (DHT - *Distributed Hash Tables*). A comunicação entre nós na rede NG é feita através da troca de mensagens. Cada mensagem é composta por diversas linhas de comando que seguem um padrão exclusivo da NG. A transmissão de mensagens pode ser feita por qualquer tecnologia da camada de enlace, necessitando somente de um cabeçalho de adaptação. O modelo NG contém muitas outras características e funcionalidades que não serão abordadas neste artigo [3], [4]. Para criarmos a rede colaborativa de dispositivos IoT focaremos somente na Nomeação, nas Mensagens e na camada de adaptação da NG.

A. Nomeação, Nomes Autoverificáveis e Resolução de Nomes

Nomes são uma forma de denotar existências. Eles podem ou não carregar significado. Chamamos de Nome em Linguagem Natural (NLN) um nome que possui significado, e.g. "Gato". Os nomes que não possuem significado para as pessoas são chamados Nomes Planos - *Flat Names*. Eles podem ser gerados a partir da passagem de um NLN por uma função *Hash*. Isso gera um SVN, uma vez que o NLN é embaralhado. A NG utiliza o algoritmo de *Hash* chamado MurMurHash3. A Figura 1a ilustra o procedimento de geração de SVN. Uma vez que a NG engloba os dois tipos de nomes, é capaz de fazer ligações que representam relações entre entidades nomeadas. Uma Ligação de Nome (NB - *name binding*) é um mapeamento entre dois ou mais nomes na forma: $<$ chave; valor(es) $>$. A Figura 1b ilustra um nó de IoT onde "NXP" é o nome do *hardware*, "*Event* OS" é o nome

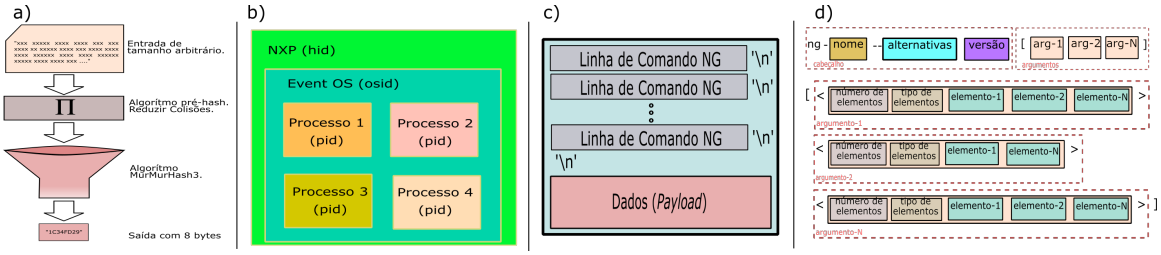

Fig. 1. A geração de nomes autoverificáveis usando função *hash* é ilustrada em a). Em b), é apresentado um exemplo de nomeação para um nó de IoT. Em c) ilustra-se uma mensagem NG. Já em d), é apresentada a estrutura geral de Linha de Comando, bem como de três parâmetros internos.

do Sistema Operacional (OS - *Operating System*) e "Processo 1", "Processo 2", "Processo 3" e "Processo 4" são nomes de processos que estão rodando neste OS. Assim sendo, existe uma relação entre estas entidades nomeadas, que pode ser representada pelos seguintes NBs:

- < **NXP, *Event* OS** > - O *Hardware* "NXP"possui o sistema "*Event* OS".
- < **Event OS, Processo 1** > - O "*Event* OS"possui um processo chamado "Processo 1".

Nestes exemplos, a ligação é feita entre NLN, mas poderia ser feita utilizando SVN:

- < **7E3FFFA6, 027552A1** > - Onde 7E3FFFA6 é o SVN do Identificador do *Hardware* "NXP" e 027552A1 é o SVN do "*Event* OS".

Em algum momento, o *Hardware* "NXP" pode estar relacionado com uma entidade que representa uma localização, como por exemplo coordenadas GPS ou por um endereço MAC: < 00:12:34:AB:CD:EF, NXP >. Neste exemplo, o NXP está localizado no endereço MAC "00:12:34:AB:CD:EF". Se por algum motivo o NXP se movimentar e entrar em outra rede, basta modificar esta única ligação de nome: < EF:CD:AB:34:12:00, NXP >, sem necessidade de atualizar outros nomes da rede. Desta forma, tem-se a independência entre identificares e localizadores, o que é muito importante quando se trata de IoT e dispositivos móveis [5].

B. Linhas de Comando e Mensagens NovaGenesis

Este trabalho é baseado em um programa NG chamado Serviço de Representação/Tradutor Embarcado (EPGS - *Embedded Proxy/Gateway Service*). Todas as tarefas executadas pelo EPGS são compostas por um conjunto de Linhas de Comandos. A Linha de Comando é um texto, feito em uma única linha, onde cada parâmetro é separado por um espaço. Ele pode ser dividido em cabeçalho e argumentos. No cabeçalho tem-se um identificador de arquitetura, o nome, as alternativas e a versão do comando. Todo comando começa com os caracteres "ng" minúsculos, que identificam o comando como sendo NovaGenesis. O próximo parâmetro do cabeçalho é o nome do comando, precedido por um sinal de menos. Em seguida, vem o parâmetro das alternativas do comando, precedido por dois sinais de menos. E por fim, temos a versão. A segunda parte do comando é composta pelos argumentos. Essa seção é encapsulada por colchetes "[...]". Podem existir um ou mais argumentos. Cada argumento é encapsulado por sinais de menor e maior "< ... >". O primeiro parâmetro do argumento indica a quantidade elementos que ele contém. Já o segundo parâmetro indica qual é o tipo dos elementos. E por fim, tem-se os elementos propriamente ditos. A Figura 1d ilustra o formato de uma linha de comando NG. Uma Mensagem transporta todos os comandos e dados necessários para a comunicação. Assim como a Linha de Comando, a Mensagem é em formato textual. A Mensagem pode ser dividida em duas partes: Linhas de Comando e Dados (*Payload*). Cada Linha de Comando ocupa uma linha da mensagem e utiliza o terminador padrão Linux (carriage return), ou seja, o caractere ASCII '\n' ou 0x0A. A segunda parte da mensagem é composta pelos dados (*payload*) a serem transmitidos. Essa parte é opcional. Caso exista, ela é separada dos comandos por um caractere de nova linha, como descrito acima. A Figura 1c ilustra o formato de uma mensagem NG.

C. Proxy/Gateway (PG) NovaGenesis

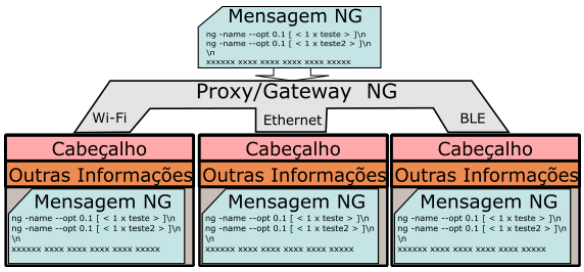

Fig. 2. *Proxy/Gateway* NovaGenesis.

O envio de mensagens é atribuído ao módulo *Proxy/Gateway* do EPGS. Ele deve ser capaz de utilizar diferentes instâncias de camada de enlace (*stack*), como Ethernet, Wi-Fi e *Bluetooth*. O *Gateway* adapta e envia a Mensagem por uma das camadas de enlace. Ele adiciona campos de cabeçalho, informações de fragmentação e tamanho da Mensagem NG. O *Gateway* envia a Mensagem adaptada para o destinatário. A Figura 2 ilustra esse processo, enquanto a Figura 3 ilustra uma mensagem pronta para ser enviada.

Fig. 3. Mensagem da arquitetura Ethernet encapsulando uma mensagem NG.

D. O Serviço de Proxy/Gateway Embarcado - EPGS

O EPGS é uma versão compacta e leve da NG para ser instalada em dispositivos com capacidade limitada de processamento, memória e energia. Ele é capaz de criar SVNes, receber e processar mensagens. Também recebe informações do dispositivo onde está embarcado, trata essas informações e cria mensagens que serão enviadas para serviços NG em nuvem. A versão atual é capaz de enviar ou receber dados via Ethernet ou Wi-Fi. A linguagem de programação C foi escolhida devido ao maior suporte que esta linguagem tem nos fabricantes de dispositivos. Todas as funções dependentes do sistema operacional são encapsuladas para facilitar a adaptação em diferentes dispositivos.

III. A Rede IoT Colaborativa com EPGS

A proposta é expandir o alcance de uma rede de dispositivos utilizando os próprios dispositivos como repetidores dos dados transmitidos, a fim de que um dispositivo emissor localizado a uma longa distância do receptor consiga manter a comunicação mesmo sem visada e utilizando baixa potência.

A. O Estabelecimento da Rede Colaborativa

Primeiramente, o EPGS precisa endereçar e conhecer os seus parceiros. Com isso, ele pode verificar um destinatário e encaminhar mensagens corretamente, caso necessário.

1) O Endereçamento: Para a rede ser colaborativa, todos os nós devem se conhecer e serem capazes de verificar se uma mensagem recebida deve ser consumida, encaminhada ou descartada. Uma linha de comando NG é responsável por carregar as informações sobre o remetente e o destinatário da mensagem. Ela é chamada de linha de comando de roteamento e é mostrada na Figura 4. Ela é composta por três argumentos. O primeiro informa o SVN do domínio a que pertence o dispositivo remetente (15B239D1). O segundo argumento é composto por quatro elementos que são os SVNes dos identificadores do *Hardware*, OS, Processo e Bloco, respectivamente, do emissor da mensagem. O terceiro e último argumento também é composto por quatro elementos com o mesmo significado. Quando a mensagem não tem um destinatário específico (*broadcast*), é usado o nome "*empty*". Analisando o último argumento da linha de comando de roteamento, um nó IoT é capaz de saber se a mensagem foi enviada para ele ou não.

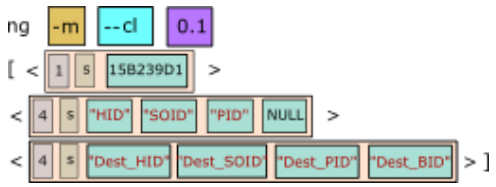

Fig. 4. Linha de comando de roteamento

2) A Descoberta de Parceiros: É a fase mais importante, onde todos os participantes da rede informam seus nomes e endereços e também armazenam as tuplas que receberam dos parceiros. Desta forma, quando receberem outras mensagens poderão identificar quem é o destinatário e fazer o tratamento correto. Quando um EPGS entra na rede, ele deve enviar uma mensagem de *Hello* para todos os participantes (*broadcast*). Os outros elementos, ao receberem o *Hello*, conseguem identificar o novo parceiro. A Figura 5a exemplifica uma mensagem de *Hello*. Os números em hexadecimal representam o cabeçalho Ethernet (14 bytes), cabeçalho de fragmentação (8 bytes) e o cabeçalho de tamanho de mensagem (8 bytes). O cabeçalho de fragmentação NG possui 8 bytes: (i) 4 bytes para identificar a mensagem (número aleatório); (ii) 4 bytes contendo o número do fragmento, caso necessário. O último campo do cabeçalho é composto por 8 bytes que representam o tamanho em bytes da mensagem. Na Figura 5a, o tamanho da mensagem é 0x0000000000000119, que em decimal é 281 bytes.

Após o cabeçalho, vem a mensagem de *Hello*. Ela é composta pela linha de comando de roteamento, onde o destinatário é o endereço de *broadcast* e por isso seu terceiro argumento é todo "*empty*". A segunda linha de comando é o *Hello* propriamente dito. Na versão 0.2, possui dois argumentos: (1) os dois primeiros elementos são sempre NULL e os três últimos informam a arquitetura da camada de enlace, a interface e o identificador do remetente; (2) só existe caso o dispositivo já conheça o Serviço Publica/Assina (PSS - *Publish/Subscribe Service*) do *core* da NG [3]. Neste caso, os quatro elementos são os SVNes do PSS, que serão utilizados para que o EPGS possa publicar um conteúdo aos demais serviços NG na nuvem. Quando um EPGS recebe o *Hello* de um parceiro, ele armazena na tabela *hash* de parceiros os SVNes obtidos na linha de comando de roteamento e as informações de endereço (arquitetura, interface e identificador) obtidos na linha de comando de *Hello*.

3) Consumo, Encaminhamento ou Descarte de Mensagens: O último passo para a Rede colaborativa ser formada é tratar as novas mensagens recebidas analisando a linha de comando de roteamento e buscando na tabela *hash* de parceiros. Ao receber uma mensagem NG qualquer, o EPGS analisa a linha de comando de roteamento e toma uma dentre três decisões possíveis: (i) Consumir; (ii) Encaminhar; e (iii) Descartar. Caso o terceiro argumento da linha de comando de roteamento contenha os SVNes dos identificadores do *Hardware*, do OS, do Processo e do Bloco do dispositivo que está recebendo a mensagem, o EPGS conclui que a mensagem foi enviada para ele e deve ser consumida. Mas se os SVNes não são do dispositivo que está recebendo, ele deve buscar por esses SVNes na tabela *hash* de parceiros que foi preenchida durante a fase de descoberta. Se o registro for encontrado, então o EPGS pega tupla que identifica o destino e encaminha a mensagem para a saída descoberta. Caso os SVNes não sejam encontrados nos parceiros, a mensagem é descartada.

IV. O Experimento

Visamos criar uma rede onde dispositivos IoT possam ofertar seus serviços de sensoriamento (temperatura e população de uma sala) a possíveis clientes assinantes. Além disso, um dos nós sensores não possui visada para o cliente e precisará da ajuda de um nó intermediário para realizar a comunicação. Assim sendo, a rede utilizada (Figura 7) é composta por:

```
a) FF FF FF FF FF FF | AC 22 0B C9 DF 3B 12 34 05 45 07 02 00 00 00 00 00 00 00 00 00 00 01 19
   ng -m --cl 0.1 [ < 1 s 15B239D1 > < 4 s 0BD95286 ED12F3ED CE362E2E 4479BCFE > < 4 s empty empty empty empty > ]
   ng -hello --ihc 0.2 [ < 6 s 51A6873B E4980E41 E86E7B38 Wifi em1 ac:22:0b:c9:df:3b > < 4 s 0BD95286 ED12F3ED DECC2634 82C5E549 > ]
   ng -scn --seq 0.1 [ < 1 s 4B00DED6 > ]

b) FF FF FF FF FF FF | 00 12 34 AB CD EF 12 34 76 B7 76 B7 00 00 00 00 00 00 00 00 00 00 01 05
   ng -m --cl 0.1 [ < 1 s 15B239D1 > < 4 s 7E3FFFA6 027552A1 7CE42568 NULL > < 4 s empty empty empty empty > ]
   ng -hello --ihc 0.2 [ < 5 s NULL NULL Wifi eth0 00:12:34:AB:CD:EF > < 4 s 0BD95286 ED12F3ED DECC2634 82C5E549 > ]
   ng -scn --seq 0.1 [ < 1 s B60D99B7 > ]

c) 00 12 34 AB CD EF | AC 22 0B C9 DF 3B 12 34 05 45 07 02 00 00 00 01 31 20 73 20 32 20 3E 20
   ng -m --cl 0.1 [ < 1 s 15B239D1 > < 4 s 0BD95286 ED12F3ED CE362E2E 4479BCFE > < 4 s 7E3FFFA6 027552A1 7CE42568 NULL > ]
   ng -p --b 0.1 [ < 1 s 2 > < 1 s BEE09CC1 > < 1 s CE362E2E > ]
   ng -p --b 0.1 [ < 1 s 1 > < 1 s BEE09CC1 > < 1 s PGCS > ]
   ng -p --b 0.1 [ < 1 s 2 > < 1 s 2AA3A2EF > < 1 s CE362E2E > ]
   ng -p --b 0.1 [ < 1 s 1 > < 1 s 2AA3A2EF > < 1 s Gateway > ]
   ng -p --b 0.1 [ < 1 s 2 > < 1 s D5546D53 > < 1 s CE362E2E > ]
   ng -p --b 0.1 [ < 1 s 1 > < 1 s D5546D53 > < 1 s Proxy > ]
   ng -p --b 0.1 [ < 1 s 2 > < 1 s 423FCDDC > < 1 s CE362E2E > ]
   ng -p --b 0.1 [ < 1 s 1 > < 1 s 423FCDDC > < 1 s Controller > ]
   ng -message --type 0.1 [ < 1 s 1 > ]
   ng -message --seq 0.1 [ < 1 s 35 > ]
   ng -scn --seq 0.1 [ < 1 s EBF2BD6B > ]

d) AC 22 0B C9 DF 3B | 00 12 34 AB CD EF 12 34 2C D6 72 AE 00 00 00 00 00 00 00 00 00 00 03 60
   ng -m --cl 0.1 [ < 1 s 15B239D1 > < 4 s 7E3FFFA6 027552A1 7CE42568 NULL > < 4 s 0BD95286 ED12F3ED DECC2634 82C5E549 > ]
   ng -p --b 0.1 [ < 1 s 2 > < 1 s 1A53F830 > < 1 s 7CE42568 > ]
   ng -p --b 0.1 [ < 1 s 1 > < 1 s 1A53F830 > < 1 s EPGS > ]
   ng -p --b 0.1 [ < 1 s 2 > < 1 s EC417E01 > < 1 s 7CE42568 > ]
   ng -p --b 0.1 [ < 1 s 1 > < 1 s EC417E01 > < 1 s Term@metro > ]
   ng -message --type 0.1 [ < 1 s 1 > ]
   ng -message --seq 0.1 [ < 1 s 0 > ]
   ng -scn --seq 0.1 [ < 1 s 6B796331 > ]

e) AC 22 0B C9 DF 3B | 00 12 34 AB CD EF 12 34 5A F1 41 BB 00 00 00 00 00 00 00 00 00 00 01 AC
   ng -m --cl 0.1 [ < 1 s 15B239D1 > < 4 s 7E3FFFA6 027552A1 7CE42568 NULL > < 4 s 0BD95286 ED12F3ED DECC2634 82C5E549 > ]
   ng -p --notify 0.1 [ < 1 s 18 > < 1 s 6153124D > < 1 s Temperature.json > < 5 s pub 0BD95286 ED12F3ED A1C31B34 BEAC1234 > ]
   ng -info --payload 0.1 [ < 1 s Temperature.json > ]
   ng -message --type 0.1 [ < 1 s 1 > ]
   ng -message --seq 0.1 [ < 1 s 2 > ]
   ng -scn --seq 0.1 [ < 1 s EB490498 > ]

   { Temperature:32 }

   MAC de destino        SVNes de destino
```

Fig. 5. a) Mensagem de Hello enviada pelo PGCS para toda rede (*broadcast*); b) Mensagem de Hello enviada pelo M-EPGS para toda rede (*broadcast*); c) Mensagem de Exposição de Recursos enviada pelo PGCS para o M-EPGS; d) Mensagem de Exposição de Recursos enviada pelo M-EPGS para o PGCS; e e) Mensagem de Publicação de Dados enviada pelo M-EPGS para o PSS.

```
a) FF FF FF FF FF FF | A1 B2 C3 D4 E5 F6 12 34 15 25 47 F2 00 00 00 00 00 00 00 00 00 00 00 DE
   ng -m --cl 0.1 [ < 1 s 15B239D1 > < 4 s ABCD5286 ED12F3ED CE362E2E NULL > < 4 s empty empty empty empty > ]
   ng -hello --ihc 0.1 [ < 6 s NULL NULL NULL NULL Wifi eth0 a1:b2:c3:d4:e5:f6 > ]
   ng -scn --seq 0.1 [ < 1 s 4B00DED6 > ]

b) 00 12 34 AB CD EF | A1 B2 C3 D4 E5 F6 12 34 14 36 44 B2 00 00 00 00 00 00 00 00 00 00 01 C0
   ng -m --cl 0.1 [ < 1 s 15B239D1 > < 4 s 7E3FFFA6 027552A1 7CE42568 NULL > < 4 s 0BD95286 ED12F3ED DECC2634 82C5E549 > ]
   ng -p --notify 0.1 [ < 1 s 18 > < 1 s 876BC23A > < 1 s Room_1_Access.json > < 5 s pub APP_HID APP_OSID APP_PID APP_BID > ]
   ng -info --payload 0.1 [ < 1 s Room_1_Access.json > ]
   ng -message --type 0.1 [ < 1 s 1 > ]
   ng -message --seq 0.1 [ < 1 s 15 > ]
   ng -scn --seq 0.1 [ < 1 s 8B616B6D > ]

   { Male:1220, Female:456, Total:1676 }

c) AC 22 0B C9 DF 3B | 00 12 34 AB CD EF 12 34 51 3D 51 3D 00 00 00 00 00 00 00 00 00 00 01 C0
   ng -m --cl 0.1 [ < 1 s 15B239D1 > < 4 s 7E3FFFA6 027552A1 7CE42568 NULL > < 4 s 0BD95286 ED12F3ED DECC2634 82C5E549 > ]
   ng -p --notify 0.1 [ < 1 s 18 > < 1 s 876BC23A > < 1 s Room_1_Access.json > < 5 s pub APP_HID APP_OSID APP_PID APP_BID > ]
   ng -info --payload 0.1 [ < 1 s Room_1_Access.json > ]
   ng -message --type 0.1 [ < 1 s 1 > ]
   ng -message --seq 0.1 [ < 1 s 15 > ]
   ng -scn --seq 0.1 [ < 1 s 8B616B6D > ]

   { Male:1220, Female:456, Total:1676 }

   MAC de destino        SVNes de destino
```

Fig. 6. a) Mensagem de *Hello* enviada pelo P-EPGS para toda rede (*broadcast*); b) Mensagem de Publicação de Dados enviada pelo P-EGPS para o M-EPGS, mas com o PSS como destino; e c) Mensagem de Publicação de Dados do P-EGPS encaminhada pelo o M-EPGS para o PSS.

- **Computador Principal (MC)** - Contém o núcleo NG (PGCS e PSS) e também a aplicação cliente das medidas, chamada de IoTTestApp.
- **EPGS Principal (M-EPGS)** - Dispositivo IoT com sensor de temperatura, com EPGS e comunicação Wi-Fi.
- **EPGS Parceiro (P-EPGS)** - Dispositivo IoT com controlador de acesso (número e o sexo das pessoas que acessam uma sala) e EPGS para NG sobre Wi-Fi.

Fig. 7. Cenário do Experimento realizado.

Para simular a cooperação, o P-EPGS foi posicionado sem visada para o MC e, portanto, sem comunicação direta. O M-EPGS foi posicionado para conseguir estabelecer comunicação

tanto com o MC quanto com o P-EPGS.

A. Inicialização e Descoberta dos Dispositivos

O MC contém os processos do *core* NG, como o PGCS (*Proxy Gateway Controller Service*) e o PSS (*Publish Subscribe Service*). Ao ser inicializado o PGCS envia periodicamente sua mensagem de *Hello* versão 0.2, como na Figura 5a. A versão 0.2 do *Hello* contém a tupla que identifica o PSS como segundo argumento.

O M-EPGS é capaz de medir temperatura. Ele está próximo do MC e mantém comunicação direta. Ao receber o *Hello* da Figura 5a, o M-EPGS armazena as informações recebidas do MC em sua tabela *hash* de parceiros. Como o *Hello* recebido é versão 0.2, o M-EPGS consegue saber quem é o PSS. Em seguida começa a enviar suas mensagens de *Hello* versão 0.2 periódicas como na Figura 5b. Continuando, o MC recebe a mensagem de *Hello* do M-EPGS e também armazena na sua tabela *hash*. Esta fase é chamada de descoberta. Agora o MC e o M-EPGS se conhecem e sabem seus endereços e nomes.

B. Exposição dos Recursos

Após a fase de descoberta os dispositivos informam uns aos outros suas características e funcionalidades. É a chamada fase de exposição de recursos. A Figura 5c ilustra a mensagem que o MC envia para o M-EPGS expondo seus recursos. Cada linha de comando "ng -p -b" informa uma ligação entre nomes. A segunda linha deste tipo indica que o MC é um PGCS. Com isso o M-EPGS já sabe quem é o PGCS e utilizará esta informação para fazer suas publicações. Depois é a vez do M-EPGS publicar a exposição de seus recursos. A Figura 5d mostra a mensagem enviada pelo M-EPGS para o PSS do MC fazendo a exposição dos recursos. O M-EPGS informa que ele é um EPGS, bem como um termômetro.

C. Publicação e Assinatura de Contrato de Serviço

O M-EPGS já sabe quem é o PGCS de saída que deve ser usado para enviar mensagens de publicação do serviço de termômetro. Também, o MC pode anunciar para possíveis clientes que o M-EPGS está disponível. O M-EPGS publica para o PSS sua oferta de serviço, informando quais são as características do seu sensor. Caso seja encontrado um cliente, mensagens NG são trocadas entre o EPGS e o PGCS para firmar um contrato de prestação de serviço [6].

D. Publicação dos Dados Sensoriados

Com o contrato feito, o M-EPGS pode começar a publicar as temperaturas medidas para o cliente. Isso é mostrado na Figura 5e, que contém um arquivo chamado Temperatura.json. Esta mensagem é enviada periodicamente ou sempre que a temperatura medida mudar.

E. A Descoberta de um Novo Dispositivo Parceiro

A publicação dos dados sensoriados finaliza o ciclo básico do funcionamento de um nó sensor na rede NG. Agora será introduzido um novo dispositivo na rede, o P-EPGS. Ele será um sensor de presença em uma sala e não conseguirá ter comunicação direta com o PGCS que está no MC. Ao ser ligado, o P-EPGS enviará sua mensagem de *Hello* na versão 0.1 (sem dados do PSS), como mostrado na Figura 6a. O único dispositivo que recebe esta mensagem é o M-EPGS, que armazena endereço MAC (A1:B2:C3:D4:E5:F6) e SVNes (ABCD5286-ED12F3ED-CE362E2E-NULL) do P-EPGS. Estes dados serão utilizados para encaminhar mensagens para o P-EPGS. O M-EPGS então manda seu *Hello* na versão 0.2 (com dados do PSS), como na Figura 5b. O P-EPGS armazena os dados do M-EPGS e também os SVNes do PSS. Após a descoberta, ambos EPGSes fazem suas exposições de recursos, como já foi mostrado na Figura 5d. Assim, o P-EPGS descobre que o M-EPGS é um PG e pode ser utilizado para encaminhar mensagens. Assim, o P-EPGS pode firmar contratos e publicar suas medidas, pois suas mensagens para o PSS são encaminhadas pelo M-EPGS.

V. Conclusão

Com este experimento demonstramos que o modelo revolucionário NovaGenesis pode ser aplicado para construção de uma rede de dispositivos IoT, onde eles podem colaborar uns com os outros para aumentar a cobertura da rede sem aumentar a potência de transmissão e consequentemente economizando bateria. Além disso, a nomeação não vinculada a localização permite que se movam sem o risco de perderem a comunicação. O modelo de contrato permite que outros dispositivos entrem na rede e disponibilizem suas capacidades de forma muito prática e rápida. Todas estas características do modelo NG permitem que seja criada uma rede escalável, econômica e rápida para a Internet das Coisas.

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. Os autores também gostariam de agradecer a FINATEL, FAPEMIG, CNPq e CAPES.

Referências

[1] A. M. Alberti, "A conceptual-driven survey on future internet requirements, technologies, and challenges," *Journal of the Brazilian Computer Society*, vol. 19, no. 3, pp. 291–311, 2013.

[2] C. Partridge, "Helping a future internet architecture mature," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 1, pp. 50–52, Dec. 2013.

[3] A. M. Alberti, M. A. F. Casaroli, D. Singh, and R. da Rosa Righi, "Naming and name resolution in the future internet: Introducing the novagenesis approach," *Future Generation Computer Systems*, vol. 67, pp. 163 – 179, 2017.

[4] A. M. Alberti, D. Mazzer, M. M. Bontempo, L. H. de Oliveira, R. da Rosa Righi, and A. C. Sodré Jr., "Cognitive radio in the context of internet of things using a novel future internet architecture called novagenesis," *Computers & Electrical Engineering*, pp. –, 2016.

[5] W. Ramirez, X. Masip-Bruin, M. Yannuzzi, R. Serral-Gracia, A. Martinez, and M. Siddiqui, "A survey and taxonomy of id/locator split architectures," *Computer Networks*, vol. 60, pp. 13 – 33, 2014.

[6] L. Sun, Y. Li, and R. A. Memon, "An open iot framework based on microservices architecture," *China Communications*, vol. 14, no. 2, pp. 154–162, February 2017.