interNexo

Auto−configuração de roteadores: *oncotô, proncovô, quicossô*

Danton Nunes, InterNexo Ltda., São José dos Campos, SP

danton.nunes@inexo.com.br

Problema focado

interNexo

Rede com segmentos terrestres e "wireless", de topologia complexa e mutável.

Equipamentos de estoque intercambiáveis.

Provisionamento para centenas de roteadores.

A grande idéia:

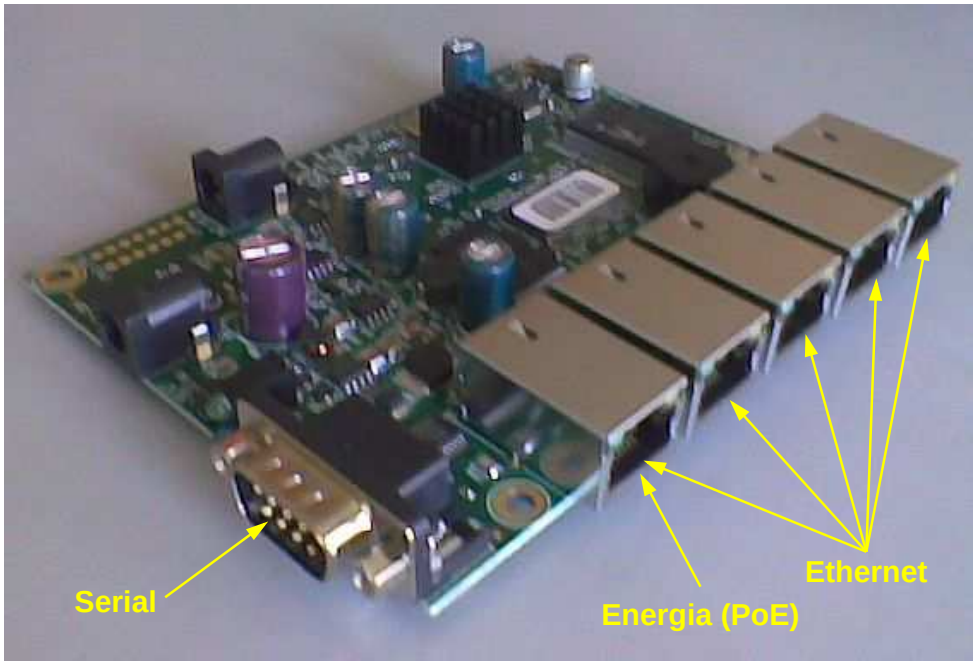
Configuração automática dos roteadores

+

Utilizar um servidor de provisionamento.

O cavalo de batalha

Hardware: RouterBoard 450



Rótulo com o endereço MAC colado na caixa.

O cavalo de batalha

interNexo

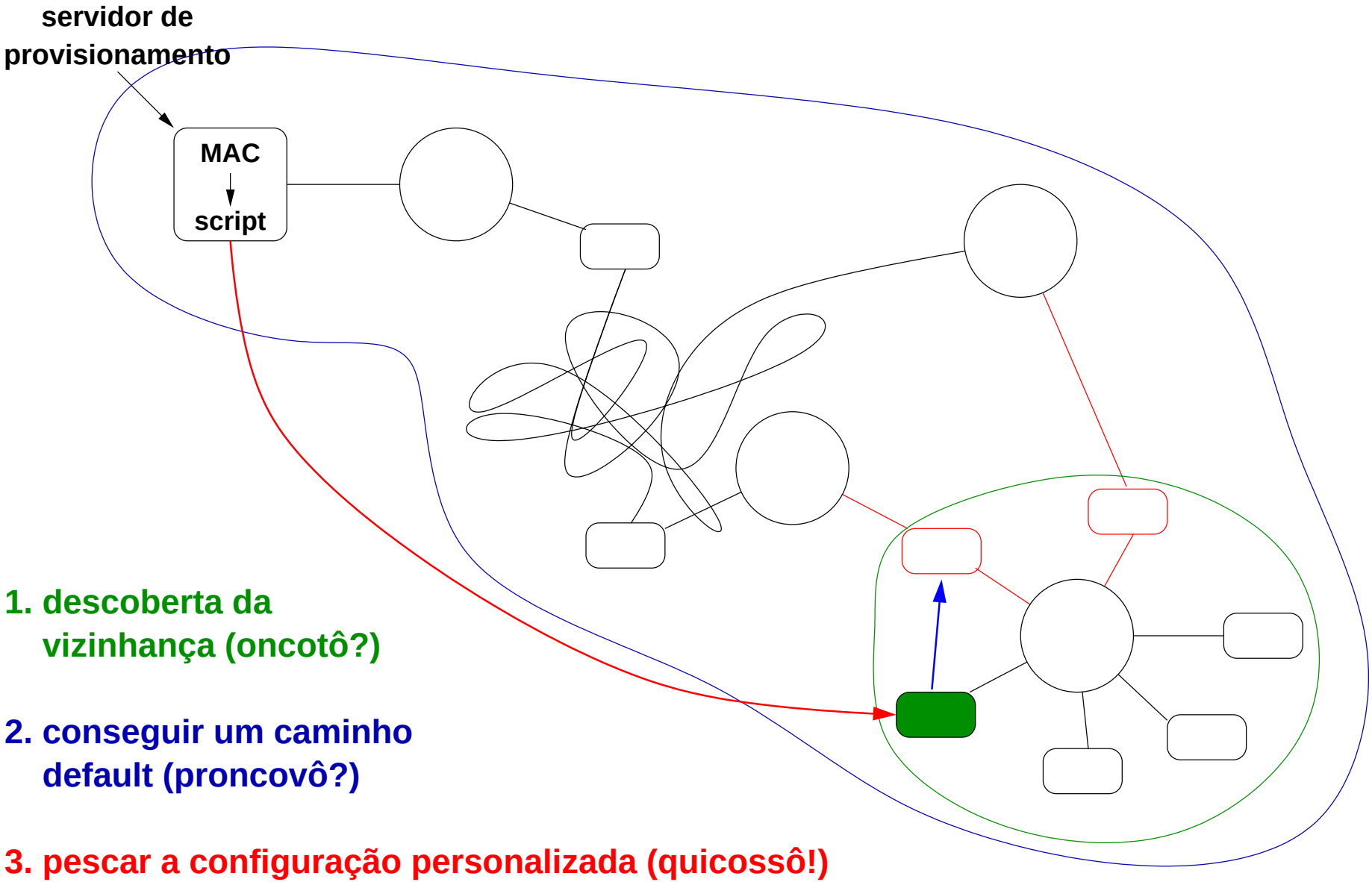
Software: Linux embutido, calcado no Slackware:

* kernel com suporte a IPv4 e IPv6
* iproute2 + iptables
* quagga (zebra + bgpd + acessórios)
* snmpd
* ssh, scp, sshd
* rdisc6 e radvd (cliente e servidor de anúncio de rotas)
* e o bom e velho /bin/bash!

Convenções ao partir

* a porta eth0 esta ligada a uma rede "upstream", i.é, que possui pelo menos um roteador com rota default.
* os endereços MAC são únicos (parece óbvio, mas sabe−se de casos de endereços MAC repetidos).

Arquitetura do sistema



1. descoberta da vizinhança (oncotô?)

2. conseguir um caminho default (proncovô?)

3. pescar a configuração personalizada (quicossô!)

oncotô?

interNexo

Etimologia: do mineirês meridional "onde é que eu estou?"

Funções

* Descobrir os roteadores vizinhos,
* Descobrir o prefixo de IPv6 público da rede local,
* Compor seu próprio endereço IPv6 público.

Alternativas

* Kernel: o Linux pode fazer isto automaticamente

* Userland: utilizar utilitarios do BSD rdisc6.

Optamos por usar o rdisc6 em função de uma diretriz de projeto de passar para a userland o que for possível e manter o kernel mínimo.

oncotô?

interNexo

rdisc6 usa o ICMPv6 para descobrir roteadores próximos e outras coisas interessantes.

usa endereço multicast "all routers"

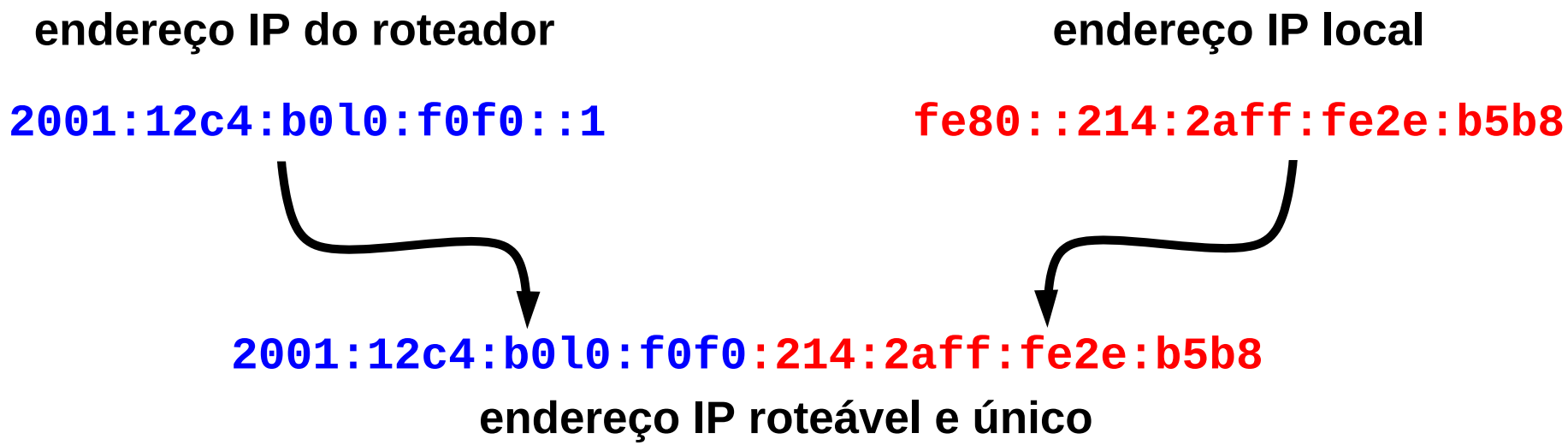
```
$ rdisc6 eth0
Soliciting ff02::2 (ff02::2) on eth0...

Hop limit                 :           64 (      0x40)
Stateful address conf.    :           No
Stateful other conf.      :           No
Router preference         :          low
Router lifetime           :           30 (0x0000001e) seconds
Reachable time            :  unspecified (0x00000000)
Retransmit time           :  unspecified (0x00000000)
 Prefix                   : 2001:12c4:b0l0:f0f0::1/64
  Valid time              :        86400 (0x00015180) seconds
  Pref. time              :        14400 (0x00003840) seconds
 Route                    : ::/0
  Route preference        :       medium
  Route lifetime          :     infinite (0xffffffff)
 Source link-layer address: 00:1E:C9:21:EC:30
 from fe80::21e:c9ff:fe21:ec30
```

encontra um prefixo roteável

e o endereço de um roteador viável.

oncotô?

O endereço IP roteável é formado juntando o prefixo do endereço obtido do rdisc6 com o sufixo do endereço local (que é derivado do endereço MAC).



finalmente o endereço é atribuído pelo método usual:

```
# ip -f inet6 addr add 2001:12c4:b0l0:f0f0:214:2aff:fe2e:b5b8/64 dev eth0
```

proncovô?

interNexo

Etimologia: do mineirês meridional "para onde que eu vou?"

Funções

*** Descobrir o roteador de "upstream",**

*** Configurar uma rota "default" estática,**

Se o rdisc6 retornou mais de um candidato a roteador default, escolhe qualquer um deles, p.ex. o primeiro!

Esta rota default é provisoria mesmo, e só serve para alcançar o servidor de provisionamento.

A rota default é injetada com o bom e velho comando

```
# ip -f inet6 route add default via fe80::21e:c9ff:fe21:ec30
```

resultado do rdisc6

quicossô?

interNexo

Etimologia: do mineirês meridional "que é que eu sou?"

Funções

*** Obter sua identidade e configurações do servidor de provisionamento,**

*** Configurar e partir as demais interfaces,**

*** Iniciar os processos de roteamento dinâmico,**

*** Iniciar o servidor de gerenciamento (snmpd).**

quicossô?

O cliente copia um tarball do servidor de provisionamento e o expande em um ramdisk, montado em /usr/local.

O servidor de provisionamento escolhe o arquivo a enviar em função do sufixo do endereço IP (portanto do MAC).

/usr/local/etc/rc.local contém um script para completar a configuração e disparar os serviços como quagga, snmpd, sshd, etc.

Só nesta fase o equipamento recebe um endereço IPv4!

A memória flash do equipamento não é alterada.

Comentários

O projeto ainda não teve o batismo de fogo, mas já funciona perfeitamente em laboratório.

Estamos elaborando os procedimentos de tratamento de erros, por enquanto o equipamento simplesmente reboota em caso de falha.

Por enquanto a única medida de segurança é o uso de autenticação por chave pública na cópia do tarball.

É possível fazer algo parecido com IPv4, mas é muito mais difícil. O único elemento que tem que manter estado é o servidor de provisionamento (compare, p.ex. o oncotô com DHCP).