0xFF FunkFeuer

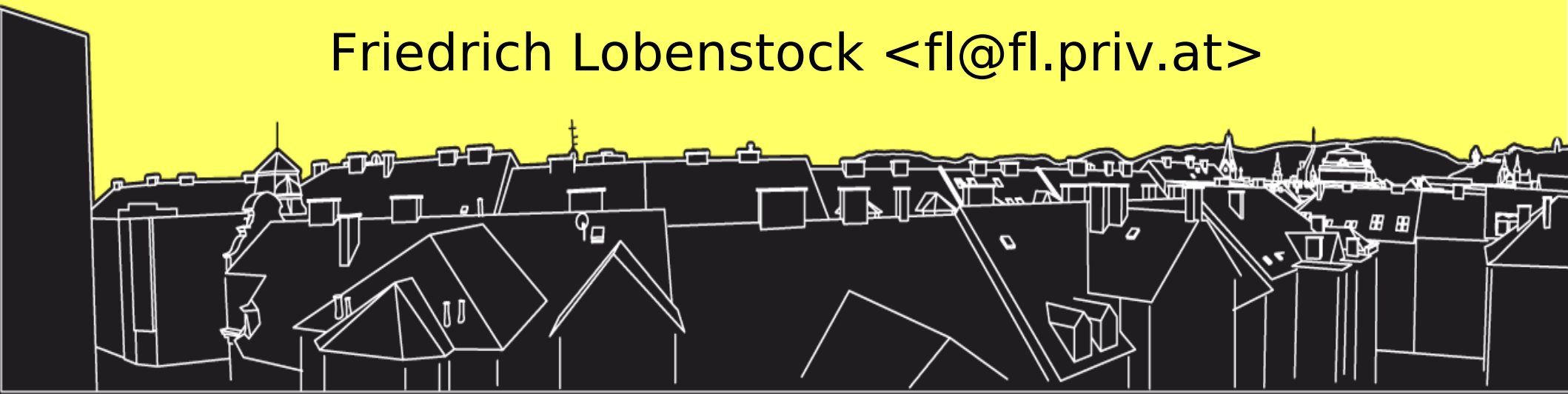
Das freie Funknetz in Graz

Authenticating packets in a provider independent network with IPSec-AH

Othmar Gsenger <otti-ff@graz.funkfeuer.at>

Christian Pointner <equinox@chaos-at-home.org>

Friedrich Lobenstock <fl@fl.priv.at>



Content

- Introduction
- Our Network
- IPsec and Anycast
- Implementation – Firmware
- OLSR+BGP
- Outlook
- The Big Picture

Introduction

- Free network as independent as possible
- Hand out public IP addresses for Internet access
- Support different upstream providers
- Protect against IP address high-jacking

0xFF FunkFeuer

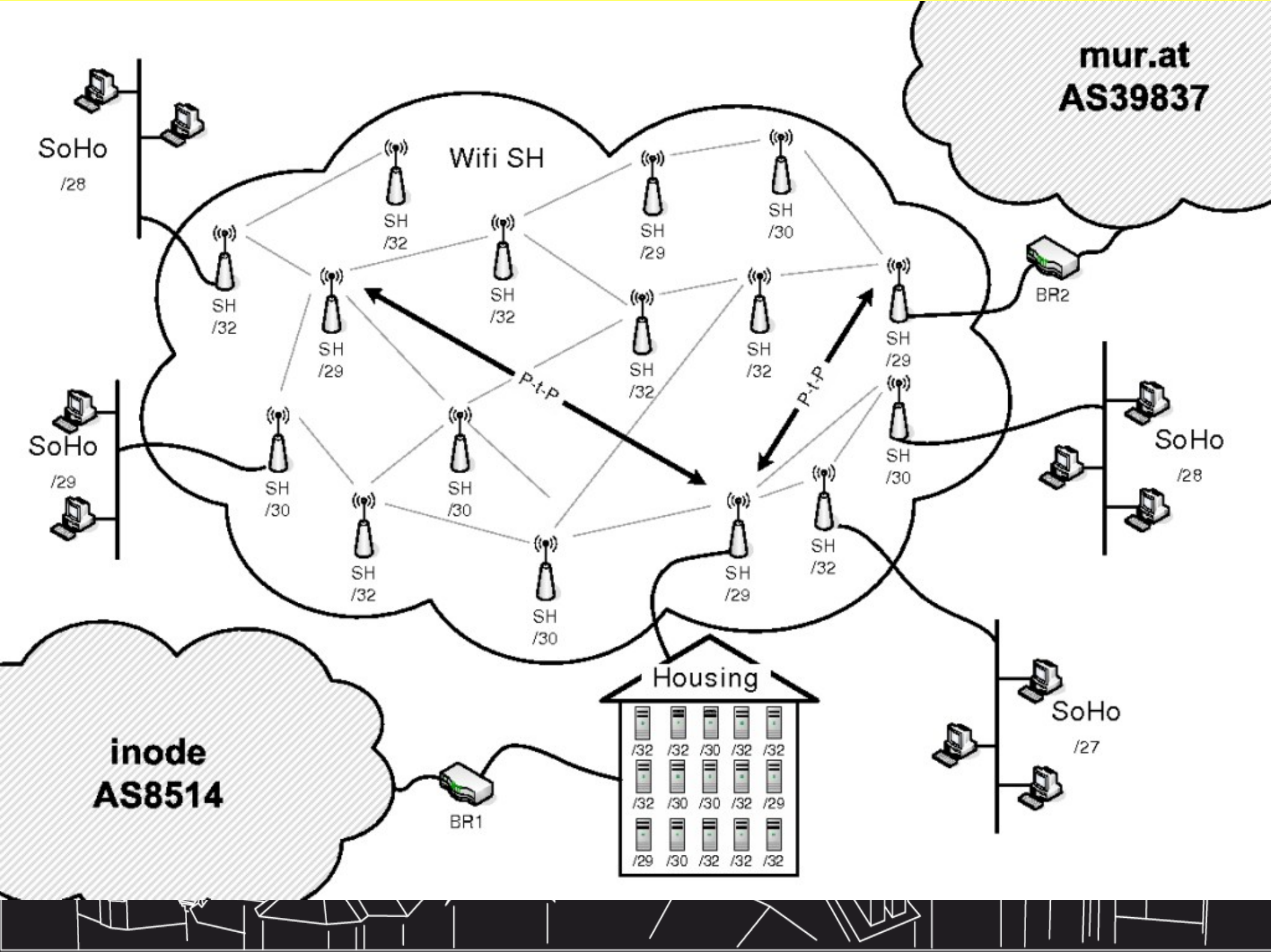
Das freie Funknetz in Graz

IPsec and Anycast

Othmar Gsenger <otti-ff@graz.funkfeuer.at>

Our Network

- One continuous wireless cloud
- multi-home to different upstream providers
- provider independent (PI) address space of public IP addresses

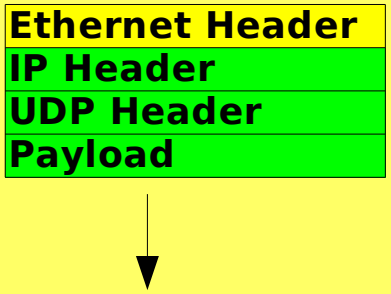

mur.at
AS39837
Wifi SH
SoHo
/28
SH
/32
SH
/32
SH
/32
SH
/29
SH
/30
SH
/29
SH
/32
SH
/32
SH
/32
SH
/29
SH
/30
BR2
P-t-P
P-t-P
SoHo
/29
SH
/30
SH
/30
SH
/32
SH
/30
SH
/29
SH
/32
SH
/29
SoHo
/28
Housing
/32 /32 /30 /32 /32
/32 /30 /30 /32 /29
/29 /30 /32 /32 /32
inode
AS8514
BR1
SoHo
/27

Routing

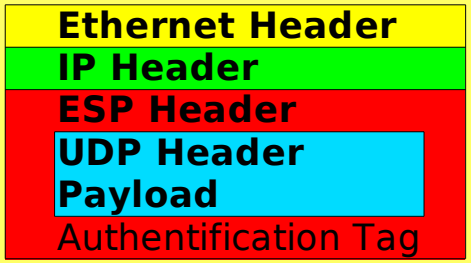
- BGP border routers announce the hole public IP address range
- At each border router there is an OLSR router, which announces the default route
- No NAT or MASQUERADE

What's IPSec? - ESP

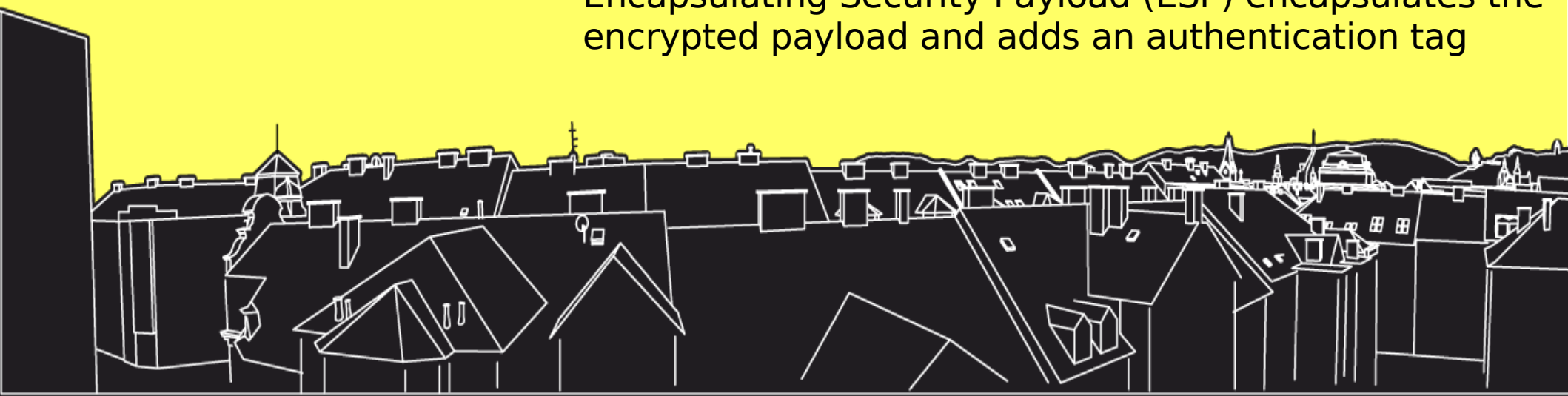
- Security extension for Ipv4 and IPv6



- Adds an additional header after the IP header

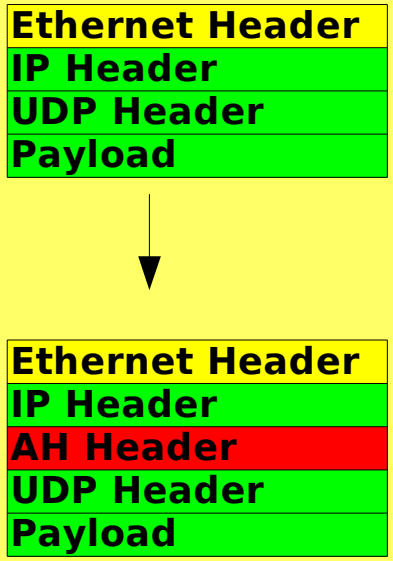


Encapsulating Security Payload (ESP) encapsulates the encrypted payload and adds an authentication tag



IPSec AH

Authentication Header (AH) adds a cryptographic checksum of the green parts, but doesn't encrypt the payload



IPSec tunnel mode?

Transport mode

Ethernet Header
IP Header
UDP Header
Payload

↓

Ethernet Header
IP Header
AH Header
UDP Header
Payload

Ipsec with IPIP tunnel (Tunnel mode)

Ethernet Header
IP Header
IP Header
UDP Header
Payload

↓

Ethernet Header
IP Header
AH Header
IP Header
UDP Header
Payload

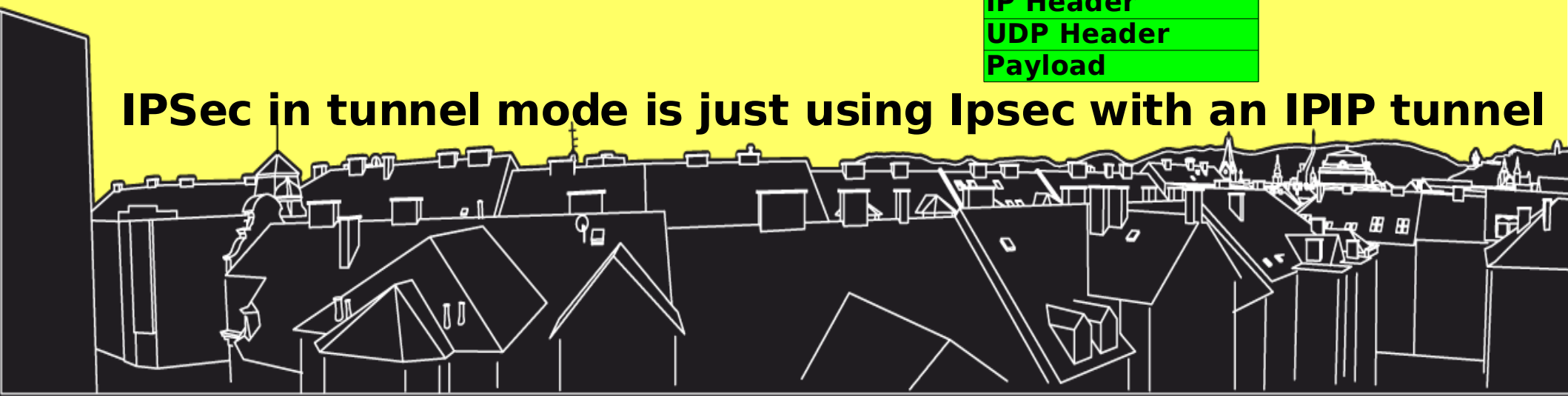
Ipsec with GRE

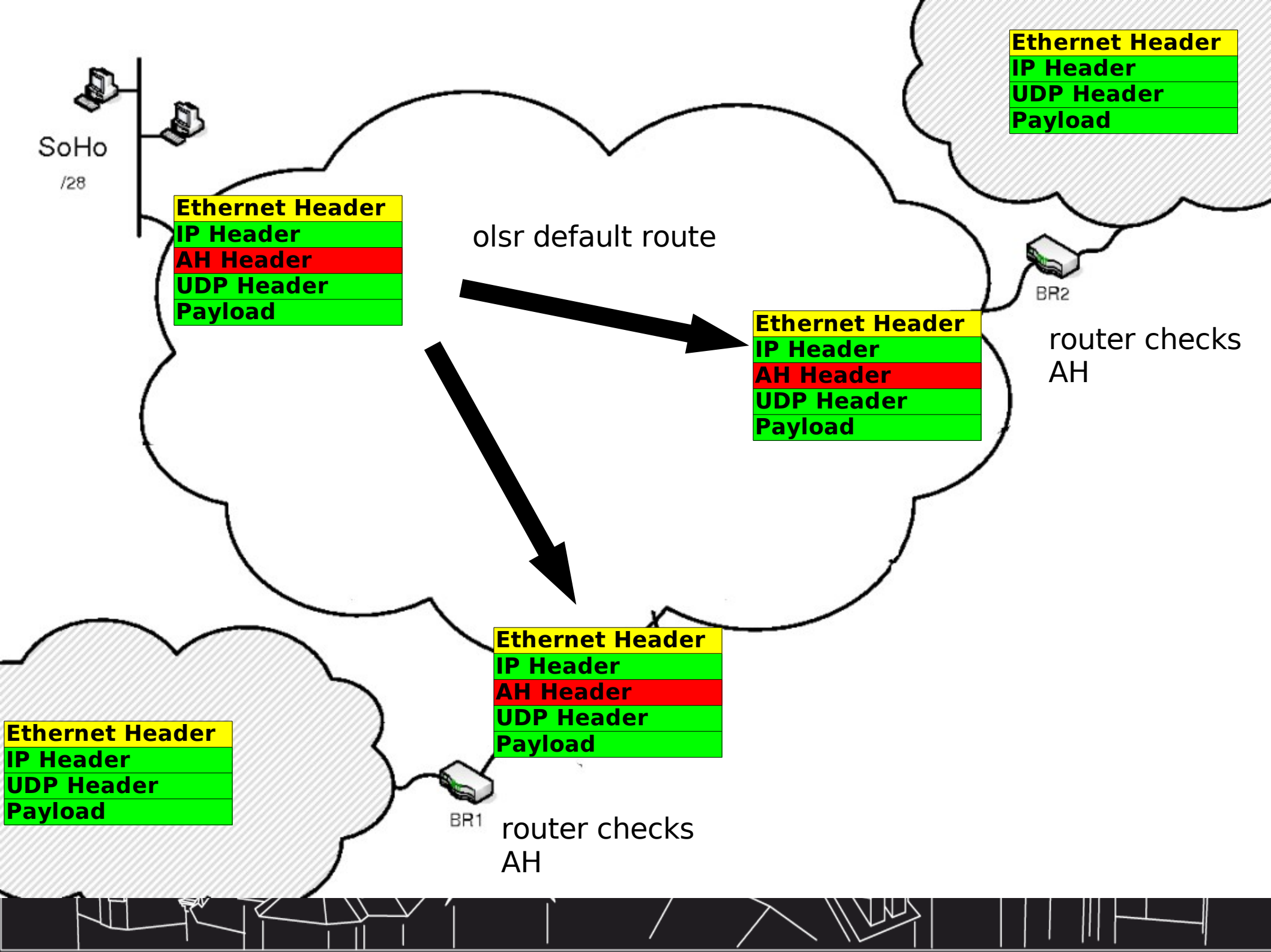
Ethernet Header
IP Header
GRE HEADER
Ethernet Header
IP Header
UDP Header
Payload

↓

Ethernet Header
IP Header
AH Header
GRE Header
Ethernet Header
IP Header
UDP Header
Payload

IPSec in tunnel mode is just using Ipsec with an IPIP tunnel




SoHo
/28
Ethernet Header
IP Header
AH Header
UDP Header
Payload
olsr default route
Ethernet Header
IP Header
UDP Header
Payload
BR2
router checks AH
Ethernet Header
IP Header
AH Header
UDP Header
Payload
Ethernet Header
IP Header
UDP Header
Payload
BR1
router checks AH

Why this isn't working

- We want IPsec to add AH **only** when we use the default route
- So we need a Security Association with all hosts, but the hosts in our routing table

Why this isn't working
What we want to do:

We

Ethernet Header
IP Header
AH Header
UDP Header
Payload

router checks and removes AH

Ethernet Header
IP Header
UDP Header
Payload

Host 1 on the internet

only we and the boarder gateway know the secret

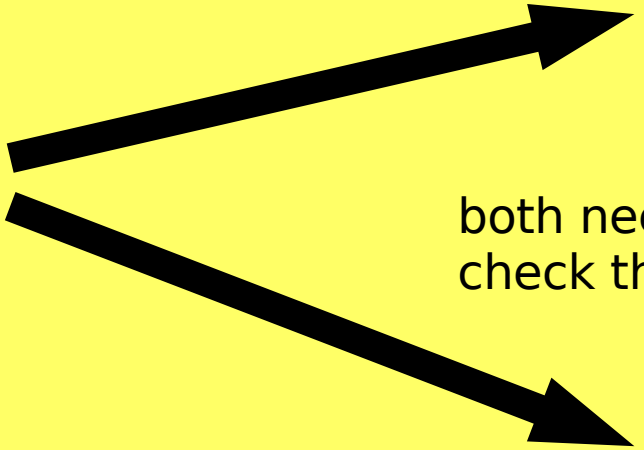
Ethernet Header
IP Header
UDP Header
Payload

host 2 on the internet

Why this isn't working
What it looks like for IPsec

We

Ethernet Header
IP Header
AH Header
UDP Header
Payload



Ethernet Header
IP Header
AH Header
UDP Header
Payload

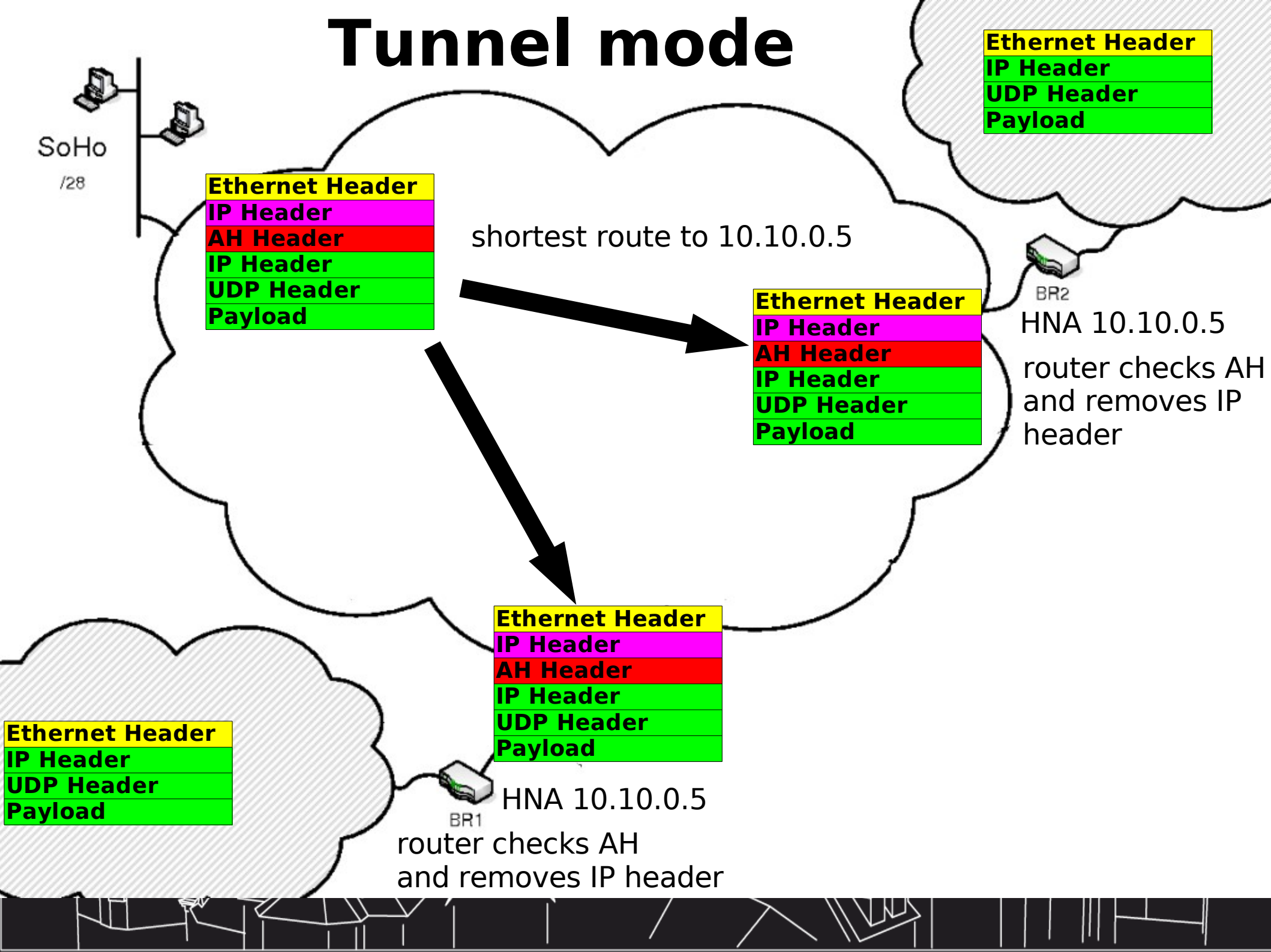
Host 1 on the internet

both need to know the session key to check the ah header

Ethernet Header
IP Header
AH Header
UDP Header
Payload

host 2 on the internet

Tunnel mode


SoHo
/28
Ethernet Header
IP Header
AH Header
IP Header
UDP Header
Payload
shortest route to 10.10.0.5
Ethernet Header
IP Header
UDP Header
Payload
BR2
HNA 10.10.0.5
router checks AH and removes IP header
Ethernet Header
IP Header
AH Header
IP Header
UDP Header
Payload
Ethernet Header
IP Header
UDP Header
Payload
BR1
HNA 10.10.0.5
router checks AH
and removes IP header

Ping ...

IP 193.33.151.42 > 10.10.0.5: AH(spi=0x00000150,seq=0xc69): IP 193.33.151.42 > 129.27.3.16: ICMP echo request, id 36463, seq 0, length 64 (ipip-proto-4)

IP 10.10.0.5 > 193.33.151.42: AH(spi=0x00000150,seq=0x4a): IP 129.27.3.16 > 193.33.151.42: ICMP echo reply, id 36463, seq 0, length 64 (ipip-proto-4)

IP 193.33.151.42 > 10.10.0.5: AH(spi=0x00000150,seq=0xc6a): IP 193.33.151.42 > 129.27.3.16: ICMP echo request, id 36463, seq 256, length 64 (ipip-proto-4)

IP 10.10.0.5 > 193.33.151.42: AH(spi=0x00000150,seq=0x4b): IP 129.27.3.16 > 193.33.151.42: ICMP echo reply, id 36463, seq 256, length 64 (ipip-proto-4)

Difference to signed routing

Signed Routing

- Protects internal routing tables
- Asymmetric Cryptography
- PKI

Authenticated Internet-Traffic

- Protects data sent to the internet
- Symmetric Cryptography
- Upstream Provider creates keys

Possible attacks

Signed Routing

- Inject data when you are on the route path

Authenticated Internet-Traffic

- Manipulate routing table

Combine both if you can!

Provider neutrality

- It's possible that multiple upstream-provider route their IP addresses into the network and protect them
- Every provider may run one or multiple border routers (with anycast IPs)
- Internal IP addressing stays valid and may be done by someone else.

Anycast

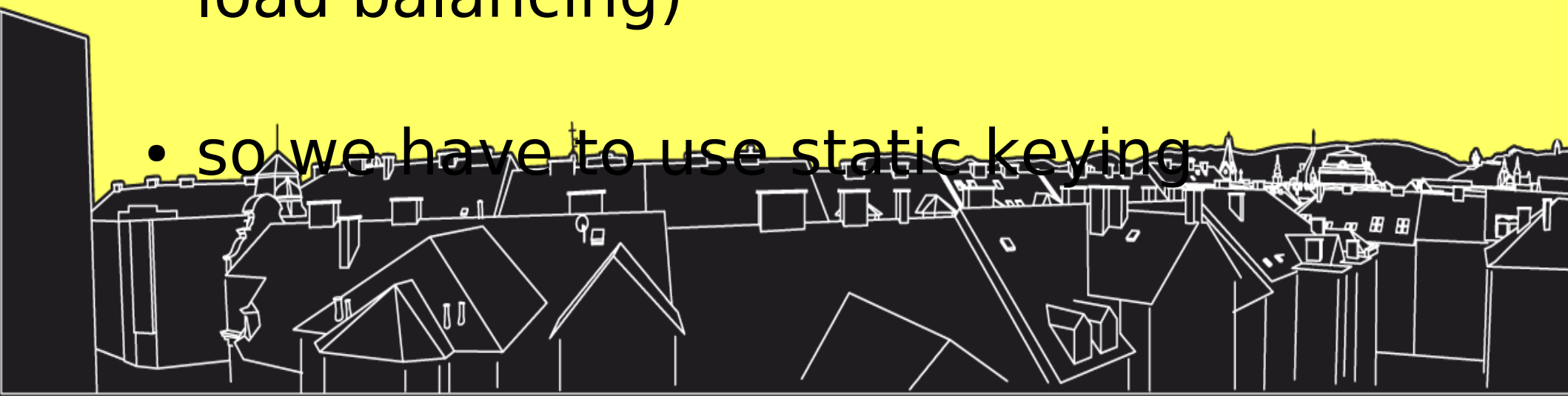
- The border gateways have the same IP address and announce it with OLSR HNA
- shorter route wins

Limits of Ipsec with anycast

- IPSec wasn't designed to allow anycast host.
- Replay protection is done by sequence numbers, but the anycast routers don't know each others sequence counter
- so replay protection doesn't work

Limits of Ipsec with anycast

- IPsec doesn't define a key management, but there is no anycast key management in existence
- synchronization of keys can help, but only for hot standby systems (not for load balancing)
- so we have to use static keying



Links to further infomation

- building hot standby IPsec tunnels with key management
  - isakmpd
  - sasyncd
  - carp
- building real anycast tunnels
  - http://www.anytun.org

0xFF FunkFeuer

Das freie Funknetz in Graz

0xFF FunkFeuer

Das freie Funknetz in Graz

Implementation - Firmware

Christian Pointner <equinox@chaos-at-home.org>

a story about penguins, swans and turtles

ipsec on linux2.4

and linux2.6

openswan™

openswan - ipsec on linux2.4 (openwrt white russian)

- consists of kernel module ond userspace *ipsec* tool
- ipsec interface device
- configuration through */etc/ipsec.conf*

Problem

- ipsec device is bound to existing interface
- bypasses kernel routing table

Solution

- bind ipsec device to a dummy ipip tunnel
- trick ipsec to use routing table, therefore ip tunnel gets bypassed

configuration

```
#  ip tunnel add dummy0 mode ipip local 127.0.0.1 remote 127.0.0.1
#  ifconfig dummy0 193.33.151.42 up
```

/etc/ipsec.conf

```
config setup
        interfaces="ipsec0=dummy0"
        pluto=no
conn ff
        type=tunnel
        left=193.33.151.42
        right=10.10.0.5
        rightsubnet=0.0.0.0/0
        auto=manual
        auth=ah
        ah=hmac-sha1-96
        ahkey=0x00000000000000000000000000000023
        spi=0x150
        authby=never
        ahreplay_window=0
```

kame-tools – ipsec on linux2.6

- consists of userspace tool *setkey* and ike-daemon *racoon*
- *manipulate the kernel SAD and SPD through pf_key*

configuration

/etc/ipsec-tools.conf (debian)

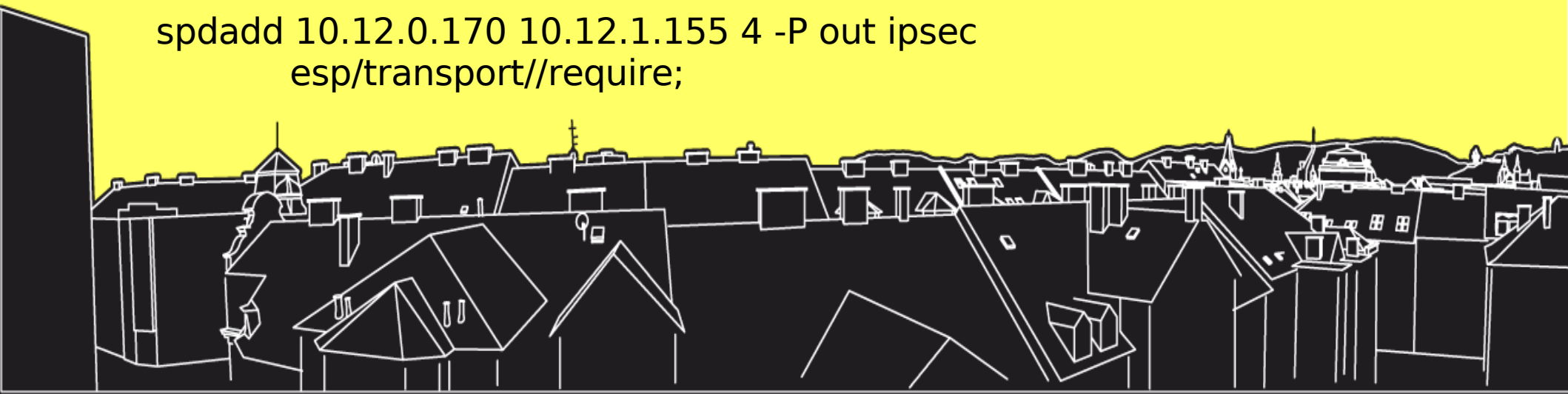
```
add 10.12.0.170 10.12.1.155 esp 0x123 -r 0
  -E aes-cbc 0x00000000000000000000000000000001
  -A hmac-sha1 0x0000000000000000000000000000000000000003;

add 10.12.1.155 10.12.0.170 esp 0x123 -r 0
  -E aes-cbc 0x00000000000000000000000000000002
  -A hmac-sha1 0x0000000000000000000000000000000000000004;

# Security policies

spdadd 10.12.1.155 10.12.0.170 4 -P in ipsec
        esp/transport//require;

spdadd 10.12.0.170 10.12.1.155 4 -P out ipsec
        esp/transport//require;
```



IPSec on Freifunk Firmware

- kernel2.4 -> openswan
- dummy ipip device
- own package consisting of some scripts

0xFF FunkFeuer

Das freie Funknetz in Graz

0xFF FunkFeuer

Das freie Funknetz in Graz

OLSR+BGP4 Outlook

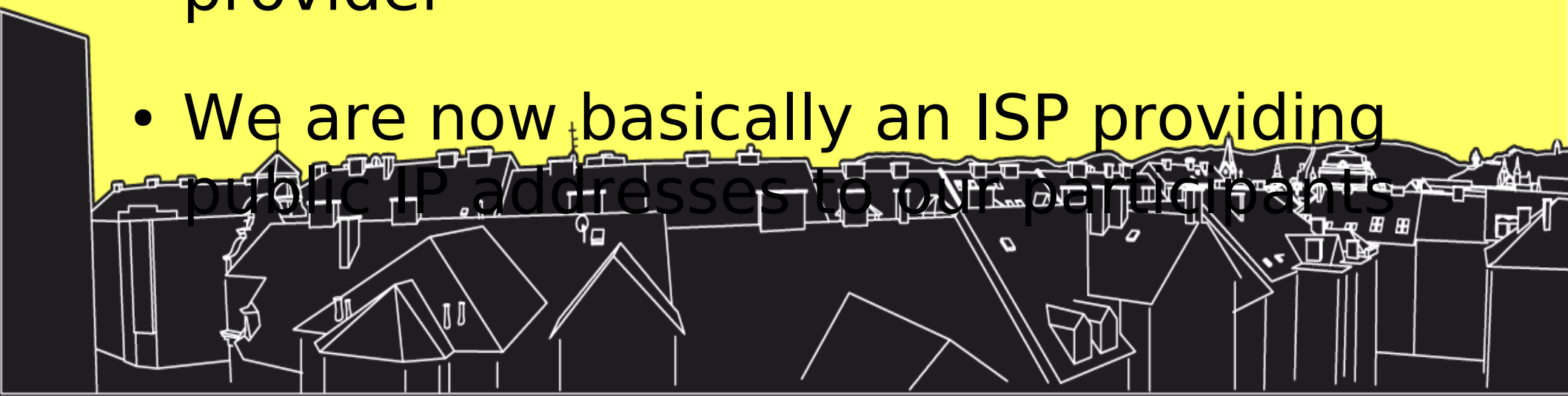
Friedrich Lobenstock <fl@fl.priv.at>

OLSR+BGP4 Outlook

- Why BGP4 (Border Gateway Protocol 4)?
- Why OLSR (Optimized Link State Routing)?
- Why get BGP and OLSR talking?
- OLSR to BGP protocol translation
- Problems
- The Future

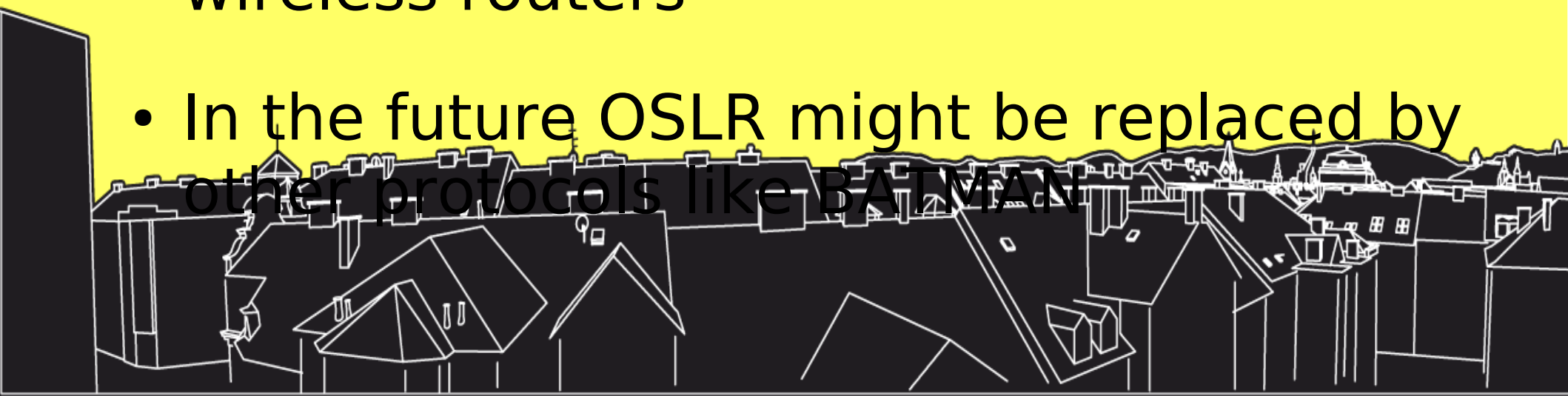
Why BGP4 (Border Gateway Protocol 4)

- The standard routing protocol on the Internet
- BGP4 is essential when multi-homing with a PI(provider independent) address space to more than one upstream provider
- We are now basically an ISP providing public IP addresses to our participants



Why OLSR (Optimized Link State Routing)?

- A routing protocol optimized for ad-hoc wireless LANs - the currently defacto standard
- Currently used because of support in Freifunk firmware for commodity wireless routers
- In the future OSLR might be replaced by other protocols like BATMAN



Why get BGP and OLSR talking?

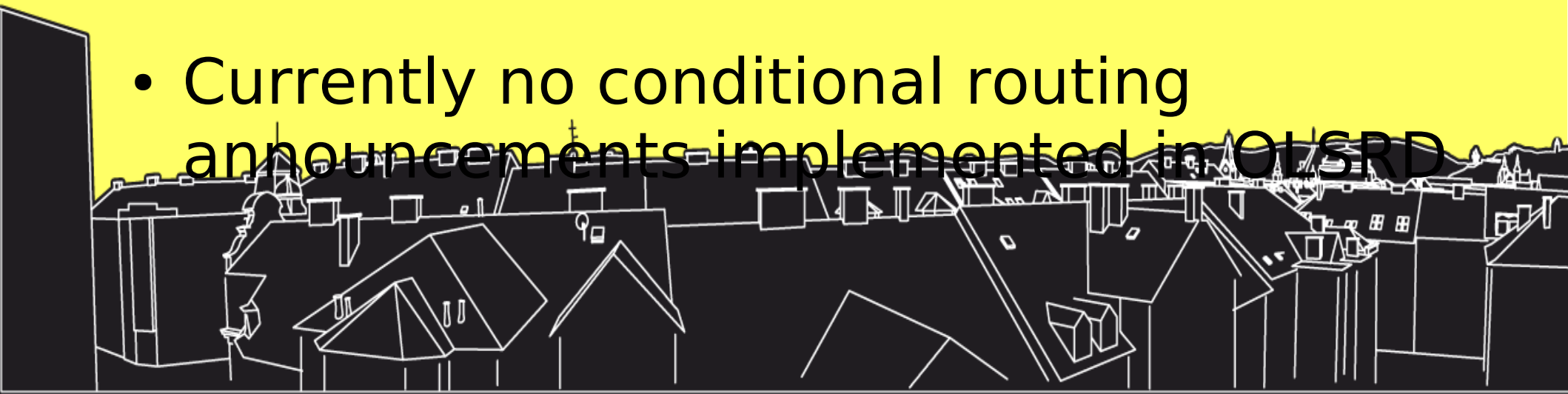
- Border Gateways do not necessarily talk OSLR (i.e. Cisco Routers), but speak BGP
- OLSR announces the gateways anycast IP but doesn't know anything about the conditions of upstreams - this info is in BGP
- Status of eBGP session needs to influence announcement of anycast IP in OLSR

OLSR to BGP protocol translation

- Each Border Gateway needs a companion router running OLSR which is only stable on MIPS platform
- Current plan was to get OLSRD talking to the popular QUAGGA routing daemon
- olsrd_quagga plug-in available for OLSRD which looked like a promising starting point

Problems and Pitfalls

- olsrd_quagga plug-in communication with Quagga had to be fixed in our local Freifunk firmware version
- Nonetheless OLSRD just crashes with this plug-in loaded and debugging led nowhere
- Currently no conditional routing announcements implemented in OLSRD

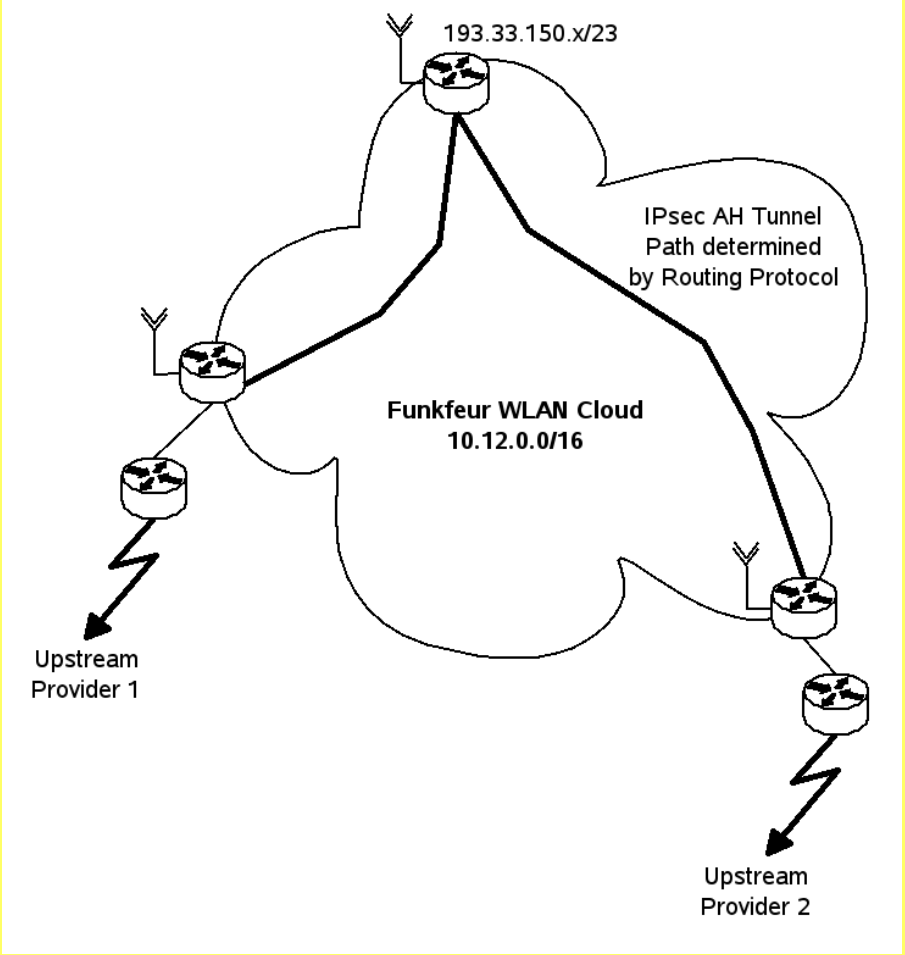


The Future

- Get olsr_quagga plug-in working
- Implement conditional HNA announcements in OLSRD based on routing info from BGP(QUAGGA)
- Motivate other projects like BATMAN to support such a communication with QUAGGA

The Big Picture

- Multi-homed with public IP addresses
- Network provider independet
- IP addresses are protected
- Network stays independent



Questions?