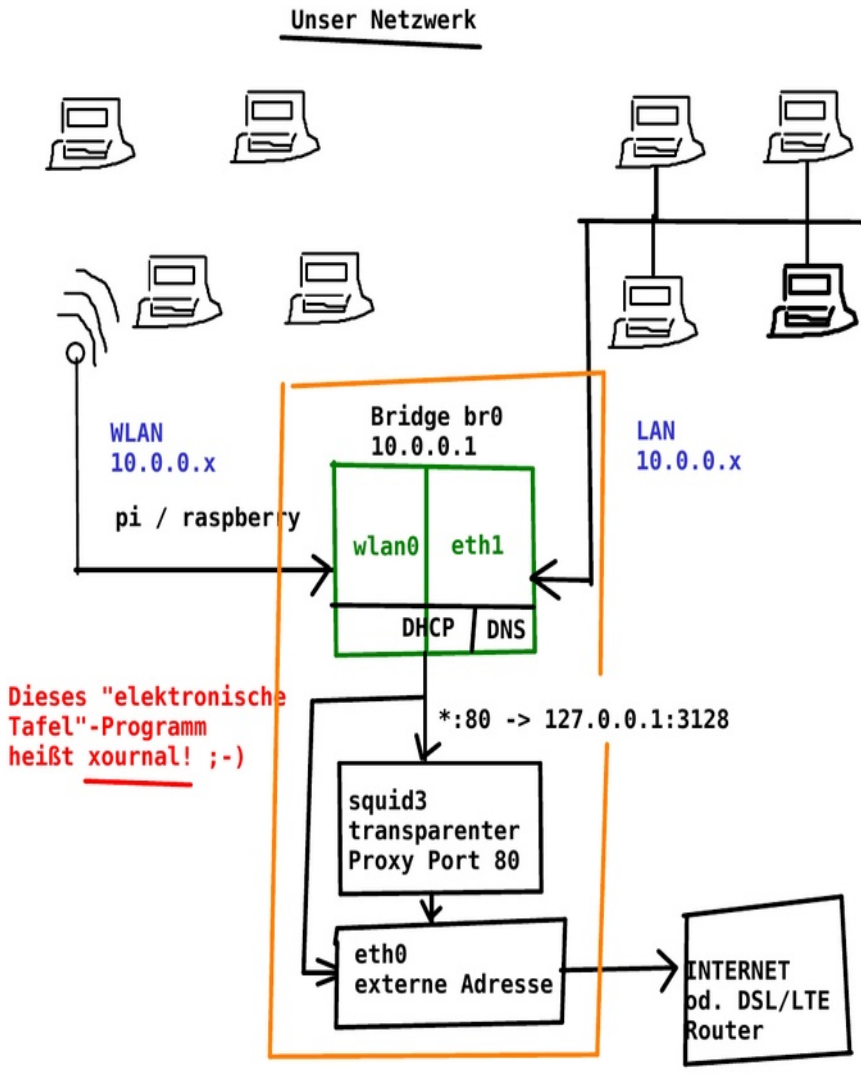
Transparenter Proxy, DHCP, Masquerading und Accesspoint mit Raspberry Pi

Ziel: Den Netzteilnehmern per WLAN und LAN einen Internet-Zugang zur Verfügung stellen, ohne dass diese etwas konfigurieren müssen. WWW-Daten zwischenspeichern und ggf. kontrollieren.



1 Internet-Zugang mit Masquerading für den RasPi-Router

Hier: Per DHCP (Client) holt sich RasPi eine gültige IP-Adresse für sich selbst. Das ist die einzige Adresse, die für das Internet sichtbar ist.

/etc/network/interfaces für die externe Anbindung per DHCP:

```
auto eth0

iface eth0 inet dhcp
```

Die Clients im internen Netz haben „private“ IP-Adressen, die per **Masquerading** auf die Adresse des RasPi umgesetzt werden!

S.a. `/etc/init.d/firewall`, die beim Hochfahren ausgeführt werden muss:

```
#/bin/bash

# Das hier wäre mit VLAN-Tagging (eine physikalische NW-Karte, zwei getrennte Netze)
# ETH_EXTERN=eth0.99
# ETH_INTERN=eth0.666

# Extern: Netzwerkkarte eth0, Intern: Bridge aus WLAN und 2. Netzwerkkarte
ETH_EXTERN=eth0
ETH_INTERN=br0

# Besonderheit dieser WLAN-Karte: Die Transmit-Queue l?uft voll -> Hässliche Meldungen
# bei dmesg und sporadisch Ausfälle, wird hier behoben

echo 10 > /sys/class/net/wlan0/tx_queue_len

# Forwarding zwischen Netzwerkinterfaces einschalten
echo 1 > /proc/sys/net/ipv4/ip_forward

# Weiterleitung von Netzwerkpaketen konfigurieren, erst mal alles auf 0
iptables -F FORWARD

# Generell Forwarding erlauben
iptables -P FORWARD ACCEPT

# Bei weitergeleiteten Paketen (postrouting) wird die Herkunfts-Adresse
# auf die ds Pi gesetzt! Umgekehrt (Pakete zurü ck) entsprechend wieder auf die
# Originale.
iptables -t nat -F POSTROUTING
iptables -t nat -A POSTROUTING -o $ETH_EXTERN -j MASQUERADE

# Transparenter Proxy redirect: Wenn auf einen Port 80 eines externen Webservers
# Daten geschickt werden, dann wird dies auf den lokalen Port 3128 "umgebogen"
iptables -t nat -F PREROUTING
iptables -t nat -A PREROUTING -i $ETH_INTERN -p tcp --dport 80 -j REDIRECT --to-ports 3128
```

2 Transparenter http-Proxy mit squid3

Der Redirect im vorigen Abschnitt muss bereits eingerichtet sein, d.h. die Clients verbinden sich

(unbemerkt) mit dem lokalen Squid auf dem RasPi Port 3128, statt mit dem „richtigen“ Webserver im Internet Port 80!

Der Squid erkennt nun anhand der http-Anfrage („Host:“-Feld), mit welchem Server der Client Kontakt wollte, und erledigt das Abholen und lokale Zwischenspeichern der Daten!

/etc/squid3/squid.conf (nur die relevanten Optionen):

```
# Squid normally listens to port 3128, transparent is transparent. ;-)
http_port 3128 transparent


# Access Control-List, damit das ganze 10-er Netz den Proxy nutzen kann,
# Achtung: Wegen des Redirects scheinen die Verbindungen von localhost zu kommen,
# der muss also auch freigeschaltet werden
acl localnet src 10.0.0.0/24    # Kindernet
http_access allow localnet
http_access allow localhost
# Große Archive auch cachen (300MB, „Windows-Updates“…)
maximum_object_size 307200 KB
```

Im Abschnitt `refresh_pattern` kann z.B. erzwungen werden, dass bestimmte Dateien auch im Cache verbleiben, wenn sie vom Server aus als „immer neu holen“ deklariert wurden, oder wenn die Clients „Reload“ drücken.

3 Netzwerk-Bridge zwischen lokalem LAN (eth1) und WLAN (wlan0)-Adapter → beide ins 10-er Netz mit der gleichen IP-Adresse

Hierfür braucht man das Paket „bridge-utils“ mit dem Programm „brctl“.

```
sudo apt-get update
sudo apt-get install bridge-utils
```

/etc/network/interfaces folgender Eintrag:

```
auto br0
# Bridge setup
iface br0 inet static
        bridge_ports wlan0 eth1
        address 10.0.0.1
        broadcast 10.0.0.255
        netmask 255.255.255.0
```

Nachträglich kann man auch Geräte in eine Bridge übernehmen: `brctl br0 addif wlan0`

und anschauen, welche Geräte drin sind: `brctl show`

Achtung: Steht eins der Geräte in der Bridge nicht sofort zur Verfügung, kommt es NICHT automatisch wieder in die Bridge, sondern muss entweder manuell, oder über einen Automatismus hinzugefügt werden. Hier kommt der nächste Abschnitt ins Spiel.

4 Accesspoint einrichten mit hostapd

Modi einer WLAN-Karte:

managed	-	über Accesspoint (normal)
ad-hoc	-	direkt von einer WLAN-Karte zur anderen („Wifi Direct“)
master	-	Gerät wird zum Accesspoint, aber noch ohne Intelligenz!

Man kann diese Modi auch manuell setzen mit `iwconfig wlan0 mode MODUS`

Wichtige Voraussetzung: Der WLAN-Adapter muss den „master“-Modus kennen.

Weitere Voraussetzung, leider nur durch Ausprobieren möglich: Er muss den Modus nicht nur kennen, sondern auch können.

Mit lsusb kann man die ID und Modelname von WLAN-Adaptern herausfinden, mit ein bisschen googlen findet man heraus, ob diese **hostapd**-fähig sind.

Hostapd einrichten, dieser managt die Verbindungen, „akzeptiert“ Clients oder weist diese zurück, steuert die Sendeleistung, sendet beacons / essid broadcasts / carrier etc.

sudo apt-get install hostapd

Konfigurieren über /etc/hostapd/hostapd.conf:

```
# Basic configuration
ctrl_interface=/var/run/hostapd
ctrl_interface_group=0
interface=wlan0
ssid=pi
wpa=3
wpa_passphrase=raspberry
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP CCMP
rsn_pairwise=CCMP
channel=11
beacon_int=100
dtim_period=2
max_num_sta=255
rts_threshold=2347
fragm_threshold=2346
country_code=DE
ieee80211d=1
# Sobald der Master-Modus läuft, wlan0 in die Bridge übernehmen!
bridge=br0
# WPA and WPA2 configuration
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wmm_enabled=0
eapol_key_index_workaround=0
eap_server=0
# Hardware configuration
# driver=rtl871xdrv
driver=nl80211
ieee80211n=1
hw_mode=g
```

5 IP-Adressen vergeben mit DHCP

Hierfür ist `/etc/dhcp/dhcpd.conf` zuständig, es muss der isc-dhcp-server installiert sein

(sudo apt-get install isc-dhcp-server).

```
ddns-update-style none;
# Hier läuft auch ein lokaler Nameserver-Cache, der Anfragen direkt beantworten kann
# (Konfiguration von bind9, nächstes Kapitel)
option domain-name-servers 10.0.0.1;
default-lease-time 600; # 10 Minuten
max-lease-time 7200;   # 2 Stunden
# Der DHCP-Server kann Anfragen ENDGÜLTIG ABLEHNEN.
authoritative;
# Diese Adressen werden frei vergeben
subnet 10.0.0.0 netmask 255.255.255.0 {
  range 10.0.0.101 10.0.0.250;
  option routers 10.0.0.1;
}

# Feste Adressen für bestimmte Rechner (per MAC-Adresse)
host referent {
  hardware ethernet 50:b7:c3:4f:98:48;
  fixed-address 10.0.0.10;
  option routers 10.0.0.1;
}

host octopi {
  hardware ethernet 00:87:33:06:8e:3e ;
  fixed-address 10.0.0.100;
  option routers 10.0.0.1;
}
```

Um zu wissen, auf welchen interfaces er laufen soll, muss man dem DHCP-Server dies in der /etc/default/isc-dhcp-server mitteilen:

```
# br0 besteht aus wlan0 und eth1
INTERFACES="br0"
```

Dann neu booten, oder:

```
sudo /etc/init.d/isc-dhcp-server restart
```

6 Caching only name server einrichten

`sudo apt-get install bind9`

In der Datei `/etc/bind/named.conf.options` sind einige Änderungen einzutragen. Wenn man vom Provider immer einen festen Nameserver bekommt, sollte dieser in „forwarders“ stehen (die IP, nicht der Name!!!)

```
options {
        directory "/var/cache/bind";

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        forwarders {
                193.84.28.4; // Hier den Nameserver des Providers eintragen
        };
        dnssec-validation auto;
        listen-on { 127.0.0.1; 10.0.0.1; }; // Hier die lokale Interface-Adresse eintragen!
        auth-nxdomain no;    # conform to RFC1035
        listen-on-v6 { none; }; // oder any, wenn auch per IPV6 Namensabfragen erlaubt sind
};
```

`sudo /etc/init.d/bind9 restart`