802.11a Bridge Link Howto Using Voyage, Madwifi, and OpenVPN

1. Overview

1.1. The motivation behind this project was two-fold. One was to build a low-cost 5.8 Ghz link using readily available hardware, and open source software. The other requirement that makes this approach unique is to be able to provide layer 2 bridging that would pass any and all MAC traffic in both directions over the link. This avoids problems with proxy arp and 3-way addressing that normally prevents layer 2 traffic from passing properly.

1.2. The solution is to use a vpn bridging tunnel. While this may seem like overkill to accomplish the goal, OpenVPN is much easier to configure than past vpn implementations, and it is open source. Bridge configuration of the vpn allows layer 2 traffic to traverse the tunnel transparently, providing for the use of any IP range at both ends.

1.3. The downside to this approach is an increase in complexity, and throughput is reduced somewhat, depending on the hardware configuration. Also, because you are using the bridging mode rather than the routed mode, you are sending more bits across the link, so it is not as efficient as a routed solution in general. Bench tests indicate a steady throughput of better than 5-7 Mbps using the hardware described.

1.4. These instructions assume a medium level of familiarity with linux and debian, therefore every step and keystroke is not spelled out.

2. Required Items:

2.1. Mainboard WRAP 2C http://www.pcengines.ch/wrap.htm
2.2. Radio Ubiquity SR5 http://www.ubnt.com/sr5/index.htm
2.3. OS base Voyage Linux http://www.voyage.hk/software/voyage.html
2.4. Storage 128Mb CompactFlash http://www.computergiants.com (I used Kingston)

3. Software Build

3.1 Prebuilt image

3.1.1. The fastest way to get up and running is to download a package of an existing build. There are currently two versions, one for the server end and one for the client or far end. The only differences are the configuration files for networking, VPN and a few other misc. items covered later. If not using the packages, continue reading.

3.2. Voyage linux

3.2.1. The version used for this project was 0.2pre1. Download and install on a linux box. Debian Sarge 3.1 was used originally, but Voyage is fairly distro-agnostic for this part. Follow the Voyage instructions on how to install from there onto a compactflash (CF). I used a Lexar USB style card reader/writer, and a Kingston 128MB flash card.

3.2.2. After installation to the CF, you want to remount the flash and reconfigure /etc/network/interfaces to suit the situation. By default, eth0 is setup for DHCP. The wireless

interface will be ath0 using the Atheros-based SR5 card. For the purposes of this document, the following assumptions are made regarding the network configuration:

a. Network subnet: 192.168.1.x mask 255.255.255.0

b. Gateway 192.168.1.1

c. DHCP is being served by some other device with addresses from 192.168.1.100 to 254

d. 192.168.1.50 to 99 is reserved and/or available for Open VPN purposes

e. Because the VPN cannot travel on the same subnet it is tunneling, 192.168.192.x was selected as the wireless IP range.

3.2.3. After remounting the flash drive, cd (your mount)/etc/network, and edit the interfaces file to look like the following:

On the Server Unit-

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.1.90
netmask 255.255.255.0
broadcast 192.168.1.255
gateway 192.168.1.1
auto ath0
iface ath0 inet static
address 192.168.192.90
netmask 255.255.255.0
broadcast 192.168.192.255
wireless_mode Master
wireless_essid bridge1 # or whatever ssid you selected
wireless_rate auto
wireless_channel 161
wireless_enc off
up /usr/bin/athctrl -d 9
```

On the Client Unit-

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.1.91
netmask 255.255.255.0
broadcast 192.168.1.255
gateway 192.168.1.1
auto ath0
iface ath0 inet static
address 192.168.192.91
netmask 255.255.255.0
broadcast 192.168.192.255
wireless_mode managed
wireless_essid bridge1
wireless_rate auto
wireless_enc off
up /usr/bin/athctrl -d 9
```

3.2.4. The ethernet port on my WRAP board had a tendency to go to sleep. I fixed this by creating a script in /etc/network/if-up called wol that contains the following:

```
#!/bin/sh
#
# command to disable wake-on-lan on eth0
#
ethtool -s eth0 wol d
```

3.2.5. Edit /etc/network/options to show ip_forward=yes instead of no.
3.2.6. Edit /ro/etc/resolv.conf to show nameserver 192.168.1.1
3.2.7. Now you can unmount the flash card, and put into the WRAP board. Either connect to the serial port or ssh into the WRAP.
3.2.8. As always, issue a remountrw to enable writing to the device. Then do apt-get update to retrieve the available files.

3.3. OpenVPN

3.3.1. Do apt-get install openvpn. when asked whether or not it should install the tun device, say yes.
3.3.2. Follow OpenVPN instructions for generating key files. I have mine stored in /etc/openvpn/keys/.
3.3.3. Create the file /etc/openvpn/bridge-start on the Server Unit and put the following in it:

```
#!/bin/bash
# Define Bridge Interface
br="br0"

# Define list of TAP interfaces to be bridged,
# for example tap="tap0 tap1 tap2".
tap="tap0"
# Define physical ethernet interface to be bridged # with TAP interface(s) above.
eth="eth0"
eth_ip="192.168.1.90"
eth_netmask="255.255.255.0"
eth_broadcast="192.168.1.255"
dgw="192.168.1.1"

for t in $tap; do
openvpn—mktun—dev $t
done
brctl addbr $br
brctl addif $br $eth

for t in $tap; do
brctl addif $br $t
done
for t in $tap; do
ifconfig $t 0.0.0.0 promisc up
done
ifconfig $eth 0.0.0.0 promisc up
ifconfig $br $eth_ip netmask $eth_netmask broadcast $eth_broadcast
for t in $tap; do
iptables -A INPUT -i $t -j ACCEPT
done
iptables -A INPUT -i $br -j ACCEPT
iptables -A FORWARD -i $br -j ACCEPT
route add default gw $dgw
```

3.3.4. Do the same for the Client Unit, changing eth_ip to 192.168.1.91

3.3.5. Create a file called /etc/openvpn/bridge-stop on both units and put the following in it:

```
#!/bin/bash
```

```
# Define Bridge Interface
br="br0"

# Define list of TAP interfaces to be bridged together
tap="tap0"
ifconfig $br down
brctl delbr $br

for t in $tap; do
openvpn—rmtun—dev $t
done
```

3.3.6. Create a file called /etc/openvpn/server-bridge.conf on the Server Unit and put the following in it:

```
port 1194
proto tcp
dev tap0
ca /etc/openvpn/keys/ca.crt
cert /etc/openvpn/keys/server.crt
key /etc/openvpn/keys/server.key  # This file should be kept secret
dh /etc/openvpn/keys/dh1024.pem
ifconfig-pool-persist /var/log/ipp.txt
persist-tun
persist-key
server-bridge 192.168.1.90 255.255.255.0 192.168.1.50 192.168.1.60
ping 11
comp-lzo
persist-key
persist-tun
status /var/log/openvpn-status.log
verb 3 #can change to 0 after everything is working - defines logging level
```

My file was built by modifying the example file that comes with openvpn, which is heavily commented. I left those comments out here in the interest of space, but be sure to read and understand them. Also, be sure to change the filename references to match the ones you generated in the key directory.

3.3.7. Create a file called /etc/openvpn/client-bridge.conf on the Client Unit and put the following in it:

```
client
dev tap0
proto tcp
remote 192.168.192.90 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ping 10
ping-restart 60
ca /etc/openvpn/keys/ca.crt
cert /etc/openvpn/keys/client1.crt
key /etc/openvpn/keys/client1.key
comp-lzo
verb 3
```

Again, change the referenced names of the files in the key directory to match the ones you generated.

3.3.8. Edit /etc/init.d/openvpn so it will run the bridge scripts at the proper time. Insert the following line right befor the line that says "$DAEMON—writepid /var/run/openvpn.$NAME.pid \"in the start_vpn() section:

/etc/openvpn/bridge-start

Also, insert this line right after the line that says "&& rm /var/run/openvpn.$NAME.status" in the stop_vpn() section:

/etc/openvpn/bridge-stop

3.3.9. I discovered an issue related to OpenVPN. If the certs in the /etc/openvpn/key directory have a date stamp that is more recent than the current machine time, it will not work. If you don't have a battery attached to the WRAP board (most folks don't) it will have an old date when it first comes up. This becomes a chicken-and-egg sort of problem, because the far end of the bridge link can't connect to the internet to run ntpdate if the bridge/tunnel is down. A less than elegant solution for now was to put a date command somewhere at the head of the /etc/init.d/openvpn script that inserts a date into the system that is later than the file dates in question. Then the tunnel can load, and ntpdate can set the correct time.

3.3.10. Madwifi doesn't reassociate when it loses it's connection, so you could lose the tunnel under the wrong circumstances. I created a script called /etc/openvpn/keepup to recover should this happen:

```
#!/bin/bash
#
while [ 1 ]; do
ping -q -c 1 192.168.1.90 > /dev/null #pinging the other end
if [ "$?" -ne 0 ]; then
echo "link down, resetting..."
iwpriv ath0 reset 1
/etc/init.d/openvpn reload
sleep 5
fi
sleep 15 # or whatever interval you feel comfortable with
done
```

Crude, but effective.

3.4. SNMP (Optional)

3.4.1. Do `apt-get install snmpd` (not necessary, but nice to have if you are using a traffic graphing package like Cacti or MRTG). Follow snmp documentation for setup.

3.5. THTTPD (Optional)

3.5.1. I didn't really feel it was necessary to provide a full-blown web interface, as I am quite comfortable with the command line and SSH. However, it seemed like it would be nice to have a web interface just for a quick status of things. I downloaded wewimo, a great little script created by Jan Krupa (krupaj@mobilnews.cz) and modified it to work with a madwifi-centric installation. Some parts of the script do not function, and have been commented out. The contents of the script are included at the end of this howto.

3.5.2. Do `apt-get install thttpd`

3.5.3. Edit /etc/thttpd/thttpd.conf to suit your installation. Mine looks like this (again, comments stripped for brevity):

```
port=80
user=www-data
logfile=/dev/null
throttles=/etc/thttpd/throttle.conf
cgipat=/cgi-bin/*.pl
```

3.5.4. Create a directory called /var/www/cgi-bin and place wistat.pl in it. You should now be able to access it by using the URL http://192,168.2.90/cgi-bin/wistat.pl

4. Misc Items

4.1. This Howto was done from memory, so a disclaimer regarding mistakes applies. Script contents were copied from the working units, so they should be OK.

4.2. Be patient when booting up, as it takes a while. A minute or so was normal on the bench. When these units were placed in the field, for some reason it took considerably longer, 2-3 minutes for the pair to link up. I haven't had a chance to find out why, it may not happen to you. Since they are up and working, it's hard to justify rebooting just to see if it will happen again.

4.3. I had to move ntpdate in the bootup order for it to work. It normally tries to run before the tunnel is up, which will fail on the far end.

4.4. Logging has not been addressed. Given the limited space in /var/log, it should be turned off or limited. It's on my todo list.....

4.5. Make sure you are feeding enough power to the WRAP board. If you believe the specs, there can be a fairly large current draw when the SR5 is running full boat. I am using a pair of 14 volt, 2 amp supplies that are doing great. I had a smaller 12 volt supply on the bench, and was getting some strange symptoms. One was that the flash device wouldn't boot about half the time.

5. Wistat.pl contents:

```
#!/bin/sh
      #
# wistat.pl - a web-based status interface for the Voyage/Madwifi/OpenVPN bridging system
# Modified by Brian Anderson
# Based on Wewimo by Jan Krupa (krupaj@mobilnews.cz)
# Modified for use with Voyage/madwifi
# http://www.mobilnews.cz/honza/
# WISP compatibility by Vic
# Rate add-on by Sameleon
# CZFree DNS add-on by Woody
# Uptime add-on by Tibor Stefanovic
# Optimalization by Passi

      # ------------------------------------------------------------------------
# -- Set your variables here -----------------------------------------------
# Paths
PATH_IWCONFIG="/sbin/iwconfig"
PATH_IFCONFIG="/sbin/ifconfig"
PATH_IWLIST="/sbin/iwlist"
PATH_ARP="/usr/sbin/arp"
PATH_HOST="/usr/bin/host"
PATH_HOSTNAME="/bin/hostname"
```

```
PATH_UPTIME="/usr/bin/uptime"
PATH_ATH_WLANS="/proc/sys/net/wlan"
PATH_WIRELESS="/proc/net/wireless"
PATH_EXEC="/cgi-bin/wistat.pl"

# Vbridge. conf is a file created that is intended for more options,
# but currently only contains the following:
# remote_eth 192.168.2.xx #put ethernet address of remote end
#
# It it just used to provide a convenient hyperlink to the other end's web interface.
PATH_FBCONFIG="/etc/Vbridge.conf"
# If you don't want to show all ifaces, you can disable it here
#DISABLE_IFACES="wlan0,wlan1"
DISABLE_IFACES="none"

# Error display suppresion. To see errors, change to "/dev/stderr"
ERR_DEST="/dev/null"
# Designated ethernet interface
E_IFACE="br0"

# If you don't want to show all MAC addresses in master mode, you can disable it here
#DISABLE_MACS="00:11:22:33:44:55,12:34:56:78:90:AB"
DISABLE_MACS="none"
# Master mode - display properties
MASTER_DISPLAY_HOST=1
MASTER_DISPLAY_IP=1
MASTER_DISPLAY_MAC=1
MASTER_DISPLAY_SIGNAL=1
MASTER_DISPLAY_RATERX=0
MASTER_DISPLAY_RATETX=0
MASTER_DISPLAY_RX=0
MASTER_DISPLAY_TX=0

# If you are running wistat.pl as CGI script change this value to 1
CGI_SCRIPT=1
# If you want to display uptime change this value to 1 (not yet fully implemented -
# it will show other information, too)
DISPLAY_UPTIME=1
# If you want to add automatic refresh change this value to seconds (0 = no refresh)
WWW_REFRESH=0
# If you want to add CZFree DNS add-on change this value to 1
CZF_DNS=0
# If you are using old version of grep which doesn't support grep -m,
# change this value to 0
GREP_M=1
# If you are running wistat.pl in WISP (Linux distribution) change this value to 1
WISP_RUN=0
WISP_PATH_ETHERS="/etc/ethers"
WISP_HOSTNAME="ComputerName"

# -- Do not change anything below this line if you are not sure what you are doing ;) --
# ------------------------------------------------------------------------

# -- Language definition ---------------------------------------------------
LNG_NAME="English"
LNG_FREQ="Frequency"
LNG_BITRATE="Bit Rate"
LNG_UNKNOWN="unknown"
LNG_MODE_UNKNOWN="Unknown"
LNG_SIGNAL="Signal"
LNG_SNR="SNR"
LNG_DATARX="RX"
LNG_DATATX="TX"
```

```
LNG_RATERX="RX Rate"
LNG_RATETX="TX Rate"
LNG_MASTER_CONNECTED_CLIENTS="Connected clients"
LNG_MASTER_HOST="Host"
LNG_MASTER_IP="IP"
LNG_MASTER_MAC="MAC"
LNG_MASTER_SIGNAL="Signal"
LNG_MASTER_CLIENTS_COUNT="Active / Total Clients"
# ------------------------------------------------------------------------

# -- Color definitions ---------------------------------------------------
COL_BG="#DDDDFF" # this is a particularly ugly color. My apologies :)
COL_TEXT="#000000"
COL_TEXT_INACTIVE="#808080"
COL_LINK="#0000FF"
COL_SBAR1="#00C000"
COL_SBAR2="#C00000"
COL_SBAR1_INACTIVE="#80C080"
COL_SBAR2_INACTIVE="#C08080"
COL_CL_BG1="#DDDDDD"
COL_CL_BG2="#FFFFFF"
COL_DATA="navy"
# ------------------------------------------------------------------------

if [ $WISP_RUN = 1 ]; then
HOSTNAME=$WISP_HOSTNAME
GREP_M=0
else
HOSTNAME=`$PATH_HOSTNAME`
fi
UPTIME=`$PATH_UPTIME`
RADIO_MAXSIGNAL=94
WISTAT_VERSION="0.1.12"
WISTAT_TITLE="Vbridge Link System"
if [ $CGI_SCRIPT = 1 ]; then
echo "Content-Type: text/html"
echo
fi
echo "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\">"
echo "<html>"
echo "<head>"
if [ $WWW_REFRESH != 0 ]; then
echo "<META HTTP-EQUIV=\"Refresh\" CONTENT=\"$WWW_REFRESH\">"
fi
echo "<META HTTP-EQUIV=\"Pragma\" CONTENT=\"no-cache\">"
echo "<title>$WISTAT_TITLE</title>"
echo "<STYLE TYPE="text/css"><!--BODY{font-family:sans-serif;--></STYLE>"

echo "</head>"
echo "<body text=$COL_TEXT bgcolor=$COL_BG link=$COL_LINK vlink=$COL_LINK>"
# Comment out next line if you don't have a logo file to display
echo "<img src="/bwbridge.gif" align=right>"
echo "<font size=+2><b><CENTER>$WISTAT_TITLE</CENTER></b></font><br>"
echo "<b>Hostname: </b> <FONT COLOR=$COL_DATA>$HOSTNAME</FONT><br>"
if [ $DISPLAY_UPTIME = 1 ]; then
echo "<font size=-1><b>Uptime:</b> <FONT COLOR=$COL_DATA>$UPTIME</FONT></font><br>"
fi
echo "<br><hr size=1><br>"

# Retrieve the web accessible IP address of the other end of the bridge from the
# vbridge config file
other_end=`cat $PATH_FBCONFIG 2>$ERR_DEST | grep ^remote_eth | gawk '{print $2}'`
mate_link="http://$other_end$PATH_EXEC"
```

```
# Ethernet interface
e_ipaddr=`$PATH_IFCONFIG $E_IFACE 2>$ERR_DEST | grep "Mask" | gawk -Faddr: '{ printf
"%s", $2 }' | gawk -F\  '{ printf "%s", $1 }'`
e_rx=`$PATH_IFCONFIG $E_IFACE 2>$ERR_DEST | grep "RX bytes" | gawk -FRX\ bytes: '{ printf
"%s", $2 }' | gawk -F\  '{ printf "%u", ($1/1024) }'`
e_tx=`$PATH_IFCONFIG $E_IFACE 2>$ERR_DEST | grep "TX bytes" | gawk -FTX\ bytes: '{ printf
"%s", $2 }' | gawk -F\  '{ printf "%u", ($1/1024) }'`
echo "<font size=+2><b>Ethernet ($E_IFACE) </b></font><br>"
echo "<b>IP Address: </b><FONT COLOR=$COL_DATA>$e_ipaddr</FONT><br>"
echo "<b>$LNG_DATARX:</b> <FONT COLOR=$COL_DATA>$e_rx</FONT> Kb     "
echo "<b>$LNG_DATATX:</b> <FONT COLOR=$COL_DATA>$e_tx</FONT> Kb <br>"
echo "<br><hr size=1><br>"

for A_IFACE in `ls -1 $PATH_ATH_WLANS`; do
if [ a`echo $DISABLE_IFACES | grep $A_IFACE` = "a" ]; then
ifmode1=`$PATH_IWCONFIG $A_IFACE 2>$ERR_DEST | grep "Mode" | gawk -FMode: '{ printf "%s",
$2 }' | gawk -F\  '{ printf "%s", $1 }'`
m_freq=`$PATH_IWCONFIG $A_IFACE 2>$ERR_DEST | grep "Freq" | gawk -FFrequency: '{ print $2
}' | gawk '{ print $1, $2 }'`
m_ipaddr=`$PATH_IFCONFIG $A_IFACE 2>$ERR_DEST | grep "Mask" | gawk -Faddr: '{ printf
"%s", $2 }' | gawk -F\  '{ printf "%s", $1 }'`
m_rx=`$PATH_IFCONFIG $A_IFACE 2>$ERR_DEST | grep "RX bytes" | gawk -FRX\ bytes: '{ printf
"%s", $2 }' | gawk -F\  '{ printf "%u", ($1/1024) }'`
m_tx=`$PATH_IFCONFIG $A_IFACE 2>$ERR_DEST | grep "TX bytes" | gawk -FTX\ bytes: '{ printf
"%s", $2 }' | gawk -F\  '{ printf "%u", ($1/1024) }'`
m_bitrate_tmp=`$PATH_IWCONFIG $A_IFACE 2>$ERR_DEST | grep "Bit Rate"`
m_bitrate=`echo $m_bitrate_tmp | gawk -FBit\ Rate '{ printf "%s", $2 }' | gawk -F\  '{
printf "%s", substr($1, 2, (length($1)-1)) }'`
# Managed mode
if [ $ifmode1 = "Managed" ]; then

m_signal1=`cat $PATH_WIRELESS 2>$ERR_DEST | grep $A_IFACE | gawk -F\  '{ if (match($3,
/\./) == 0) { printf "%s", $3 } else { printf "%s", substr($3, 1, (length($3)-1)) } }'`
m_signal2=`expr $RADIO_MAXSIGNAL - $m_signal1 2>$ERR_DEST`
m_signal1_img=`echo $RADIO_MAXSIGNAL $m_signal1 | gawk -F\  '{ printf "%u", (100 / $1 *
$2) }'`
m_signal2_img=`echo $RADIO_MAXSIGNAL $m_signal2 | gawk -F\  '{ printf "%u", (100 / $1 *
$2) }'`
m_snr1=`cat $PATH_WIRELESS 2>$ERR_DEST | grep $A_IFACE | gawk -F\  '{ if (match($4, /\./)
== 0) { printf "%s", $4 } else { printf "%s", substr($4, 1, (length($4)-1)) } }'`
m_snr2=`cat $PATH_WIRELESS 2>$ERR_DEST | grep $A_IFACE | gawk -F\  '{ if (match($5, /\./)
== 0) { printf "%s", $5 } else { printf "%s", substr($5, 1, (length($5)-1)) } }'`
m_snr=`expr $m_snr1 - $m_snr2 2>$ERR_DEST`
echo "<font size=+2><b>Wireless ($A_IFACE)</b></font><br>"
echo "(Managed Mode, $LNG_FREQ <FONT COLOR=$COL_DATA>$m_freq</FONT>, $LNG_BITRATE <FONT
COLOR=$COL_DATA>$m_bitrate Mbps</FONT>)<br>"
echo "<b>IP Address: </b><FONT COLOR=$COL_DATA>$m_ipaddr</FONT><br>"
echo "<table width=280 cellspacing=0 cellpadding=0 border=0><tr>"
echo "<td width=170><b>$LNG_SIGNAL:</b> <FONT
COLOR=$COL_DATA>$m_signal1/$RADIO_MAXSIGNAL</FONT></td>"
echo "<td width=110>"
echo "<table width=100 cellspacing=0 cellpadding=0 border=0><tr>"
echo "<td width=$m_signal1_img height=12 bgcolor=$COL_SBAR1><font
size=1> </font></td>"
echo "<td width=$m_signal2_img bgcolor=$COL_SBAR2><font size=1> </font></td>"
echo "</tr></table>"
echo "</td>"
echo "</tr></table>"

echo "<b>$LNG_SNR:</b> <FONT COLOR=$COL_DATA>$m_snr</FONT><br>"
echo "<b>Link Quality: </b> <FONT COLOR=$COL_DATA>"
if [ "$m_snr" -lt "10" ]; then echo "  Poor<br>"; fi
if [ "9" -lt "$m_snr" -a "20" -gt "$m_snr" ]; then echo "  Good<br>"; fi
```

```
if [ "19" -lt "$m_snr" -a "30" -gt "$m_snr" ]; then echo "  Very Good<br>"; fi
if [ "28" -lt "$m_snr" ]; then echo "   Excellent<br>"; fi
echo "</FONT>"
echo "<b>$LNG_DATARX:</b> <FONT COLOR=$COL_DATA>$m_rx Kb</FONT>"
echo "<b>$LNG_DATATX:</b> <FONT COLOR=$COL_DATA>$m_tx Kb</FONT>"
echo "<br><br>"
echo "Bridge Paired Device: <a href="http://$other_end/cgi-bin/wistat.pl">$other_end</a>"
echo "<br><br>"
echo "<hr size=1><br>"

# Master mode
elif [ $ifmode1 = "Master" ]; then

echo "<font size=+2><b>Wireless ($A_IFACE)</b></font><br>"
echo "(Master Mode, $LNG_FREQ <FONT COLOR=$COL_DATA>$m_freq</FONT>)<br>"
echo "<b>IP Address: </b><FONT COLOR=$COL_DATA>$m_ipaddr</FONT><br>"
echo "<b>$LNG_DATARX:</b> <FONT COLOR=$COL_DATA>$m_tx Kb</FONT>" # switched - you can see
it from CLIENT side
echo "<b>$LNG_DATATX:</b> <FONT COLOR=$COL_DATA>$m_rx Kb</FONT><br>"
echo "<br>"
echo "<b>$LNG_MASTER_CONNECTED_CLIENTS:</b><br>"
echo "<table width=100% cellspacing=0 cellpadding=2 border=0>"
echo "<tr>"
if [ $MASTER_DISPLAY_HOST = 1 ]; then echo "<td align=left><b>$LNG_MASTER_HOST</b></td>";
fi
if [ $MASTER_DISPLAY_IP = 1 ]; then echo "<td align=left><b>$LNG_MASTER_IP</b></td>"; fi
if [ $MASTER_DISPLAY_MAC = 1 ]; then echo "<td align=left><b>$LNG_MASTER_MAC</b></td>";
fi
if [ $MASTER_DISPLAY_SIGNAL = 1 ]; then echo "<td
align=left><b>$LNG_MASTER_SIGNAL</b></td>"; fi
if [ $MASTER_DISPLAY_RATERX = 1 ]; then echo "<td align=right><b>$LNG_RATERX</b></td>";
fi
if [ $MASTER_DISPLAY_RATETX = 1 ]; then echo "<td align=right><b>$LNG_RATETX</b></td>";
fi
if [ $MASTER_DISPLAY_RX = 1 ]; then echo "<td align=right><b>$LNG_DATARX</b></td>"; fi
if [ $MASTER_DISPLAY_TX = 1 ]; then echo "<td align=right><b>$LNG_DATATX</b></td>"; fi
echo "</tr>"
tmp_color=0
count_clients_total=0
count_clients_active=0

for A_MAC in `$PATH_IWLIST $A_IFACE ap 2>$ERR_DEST | grep level | gawk '{print $1}'`; do
if [ a`echo $DISABLE_MACS | grep -i $A_MAC` = "a" ]; then
#am_file=$PATH_ATH_WLANS/$A_IFACE/$A_MAC
count_clients_total=`expr $count_clients_total + 1 2>$ERR_DEST`
tmp_color_text=$COL_DATA
tmp_color_bar1=$COL_SBAR1
tmp_color_bar2=$COL_SBAR2

if [ $GREP_M = 1 ]; then
am_ip=a`$PATH_ARP -n -i $A_IFACE 2>$ERR_DEST | grep -m 1 -i $A_MAC | gawk '{ printf "%s",
$1 }'`
else
am_ip=a`$PATH_ARP -n -i $A_IFACE 2>$ERR_DEST | grep -i $A_MAC | gawk '{ printf "%s", $1
}'`
fi
if [ $WISP_RUN = 1 ]; then
am_host=`cat $WISP_PATH_ETHERS 2>$ERR_DEST | grep -i $A_MAC | gawk -F\  '{ printf "%s",
$2 }'`
else
am_host=$LNG_UNKNOWN
fi
```

```
if [ $am_ip = "a" ]; then
am_ip=" "
tmp_color_text=$COL_TEXT_INACTIVE
tmp_color_bar1=$COL_SBAR1_INACTIVE
tmp_color_bar2=$COL_SBAR2_INACTIVE
else
count_clients_active=`expr $count_clients_active + 1 2>$ERR_DEST`
am_ip=`echo $am_ip | cut -c 2-`
if [ $WISP_RUN != 1 ]; then
#  am_host1=`$PATH_HOST $am_ip 2>$ERR_DEST | gawk -F\  '{ printf "%s", $5 }'`
#  am_host2=`$PATH_HOST $am_ip 2>$ERR_DEST | grep "Name:" | gawk -F\  '{ printf "%s", $2
}'`
#  am_host=`echo $am_host1$am_host2 2>$ERR_DEST | gawk -F. '{ printf "%s", $1 }'`
if [ $CZF_DNS = 1 ]; then
if [ "`echo \"$am_host\" | gawk '\$1 ~ /^wlan[0-9]/ { yes = \"1\" }; END { print yes }'`"
== "1" ]; then
am_host=`echo $am_host1$am_host2 | gawk -F. '{ printf "%s", $3 }'`
fi
fi
fi
am_host=`echo $am_host | gawk '{ if (match($1, /\(/) == 0) { printf "%s", $1 } else {
printf "unresolved" } }'`
fi
# am_rx=`cat $am_file 2>$ERR_DEST | grep "rx_bytes" | gawk -F= '{ printf "%u", ($2/1024)
}'`
# am_tx=`cat $am_file 2>$ERR_DEST | grep "tx_bytes" | gawk -F= '{ printf "%u", ($2/1024)
}'`
am_silence=`$PATH_IWLIST $A_IFACE ap 2>$ERR_DEST | grep "level" | gawk -Flevel= '{ printf
"%d", $3 }'`
am_signal=`$PATH_IWLIST $A_IFACE ap 2>$ERR_DEST | grep "level" | gawk -Flevel= '{ printf
"%d", $2 }'`
am_signal1=`expr $am_signal - $am_silence 2>$ERR_DEST`
# am_rate_rx=`cat $am_file 2>$ERR_DEST | grep "last_rx:" | gawk -F\  '{ printf "%s", $4
}' | gawk -F= '{ printf "%.1f", ($2/10) }'`
# am_rate_tx=`cat $am_file 2>$ERR_DEST | grep "tx_rate" | gawk -F= '{ printf "%.1f",
($2/10) }'`
am_signal2=`expr $RADIO_MAXSIGNAL - $am_signal1 2>$ERR_DEST`
am_signal1_img=`echo $RADIO_MAXSIGNAL $am_signal1 | gawk -F\  '{ printf "%u", (100 / $1 *
$2) }'`
am_signal2_img=`echo $RADIO_MAXSIGNAL $am_signal2 | gawk -F\  '{ printf "%u", (100 / $1 *
$2) }'`
if [ $tmp_color = 0 ]; then
echo "<tr bgcolor=$COL_CL_BG1>"
else
echo "<tr bgcolor=$COL_CL_BG2>"
fi
if [ $MASTER_DISPLAY_HOST = 1 ]; then echo "<td><font
color=$tmp_color_text><b>$am_host</b></font></td>"; fi
if [ $MASTER_DISPLAY_IP = 1 ]; then echo "<td><font color=$tmp_color_text><a
href="$mate_link">$am_ip</a></font></td>"; fi
if [ $MASTER_DISPLAY_MAC = 1 ]; then echo "<td><font
color=$tmp_color_text>$A_MAC</font></td>"; fi
if [ $MASTER_DISPLAY_SIGNAL = 1 ]; then
echo "<td>"
echo "<table width=180 cellspacing=0 cellpadding=0 border=0><tr>"
echo "<td width=70><font color=$tmp_color_text>$am_signal1/$RADIO_MAXSIGNAL<br>$am_signal
dbm</font></td>"
echo "<td width=110>"
echo "<table width=100 cellspacing=0 cellpadding=0 border=0><tr>"
echo "<td width=$am_signal1_img height=12 bgcolor=$tmp_color_bar1><font
size=1> </font></td>"
echo "<td width=$am_signal2_img bgcolor=$tmp_color_bar2><font size=1> </font></td>"
echo "</tr></table>"
```

```
echo "</td>"
echo "</tr></table>"

echo "</td>"
fi
if [ $MASTER_DISPLAY_RATERX = 1 ]; then echo "<td align=right><font
color=$tmp_color_text>$am_rate_tx Mb/s</font></td>"; fi # switched - you can see it from
CLIENT side
if [ $MASTER_DISPLAY_RATETX = 1 ]; then echo "<td align=right><font
color=$tmp_color_text>$am_rate_rx Mb/s</font></td>"; fi
if [ $MASTER_DISPLAY_RX = 1 ]; then echo "<td align=right><font
color=$tmp_color_text>$am_tx Kb</font></td>"; fi # switched - you can see it from CLIENT
side
if [ $MASTER_DISPLAY_TX = 1 ]; then echo "<td align=right><font
color=$tmp_color_text>$am_rx Kb</font></td>"; fi
echo "</tr>"
tmp_color=`expr $tmp_color + 1 `
if [ 1 -lt $tmp_color ]; then
tmp_color=0
fi
fi
done
echo "</table>"
echo "<br>"

echo "<b>$LNG_MASTER_CLIENTS_COUNT:</b> $count_clients_active / $count_clients_total<br>"
echo "<br><hr size=1><br>"
# Ad-Hoc mode
elif [ $ifmode1 = "Ad-Hoc" ]; then

m_signal1=`cat $PATH_WIRELESS 2>$ERR_DEST | grep $A_IFACE | gawk -F\  '{ if (match($3,
/\./) == 0) { printf "%s", $3 } else { printf "%s", substr($3, 1, (length($3)-1)) } }'`
m_signal2=`expr $RADIO_MAXSIGNAL - $m_signal1 2>$ERR_DEST`
m_signal1_img=`echo $RADIO_MAXSIGNAL $m_signal1 | gawk -F\  '{ printf "%u", (100 / $1 *
$2) }'`
m_signal2_img=`echo $RADIO_MAXSIGNAL $m_signal2 | gawk -F\  '{ printf "%u", (100 / $1 *
$2) }'`
m_snr1=`cat $PATH_WIRELESS 2>$ERR_DEST | grep $A_IFACE | gawk -F\  '{ if (match($4, /\./)
== 0) { printf "%s", $4 } else { printf "%s", substr($4, 1, (length($4)-1)) } }'`
m_snr2=`cat $PATH_WIRELESS 2>$ERR_DEST | grep $A_IFACE | gawk -F\  '{ if (match($5, /\./)
== 0) { printf "%s", $5 } else { printf "%s", substr($5, 1, (length($5)-1)) } }'`
m_snr=`expr $m_snr1 - $m_snr2 2>$ERR_DEST`
echo "<font size=+1><b>$A_IFACE</b></font>"
echo "(Ad-Hoc, $LNG_FREQ $m_channel, $LNG_BITRATE $m_bitrate)<br>"
echo "<table width=180 cellspacing=0 cellpadding=0 border=0><tr>"
echo "<td width=70><b>$LNG_SIGNAL:</b> $m_signal1/$RADIO_MAXSIGNAL</td>"
echo "<td width=110>"
echo "<table width=100 cellspacing=0 cellpadding=0 border=0><tr>"
echo "<td width=$m_signal1_img height=12 bgcolor=$COL_SBAR1><font
size=1> </font></td>"
echo "<td width=$m_signal2_img bgcolor=$COL_SBAR2><font size=1> </font></td>"
echo "</tr></table>"
echo "</td>"
echo "</tr></table>"

echo "<b>$LNG_SNR:</b> $m_snr<br>"
echo "<b>$LNG_DATARX:</b> $m_rx Kb"
echo "<b>$LNG_DATATX:</b> $m_tx Kb"
echo "<br><br>"

echo "<hr size=1><br>"
# Unknown mode
else
```

```
echo "<font size=+1><b>$A_IFACE</b></font>"
echo "($LNG_MODE_UNKNOWN)<br>"
echo "<br><hr size=1><br>"
fi
fi
done
echo "<font size=-1>"
date
echo "<br>"
echo "Generated by Wistat v$WISTAT_VERSION.<br>"
echo "Based on <a
href=\"http://www.mobilnews.cz/honza/en_prog_linux_wewimo.php\">WeWiMo</a>
echo "</font>"
echo "</body>"
echo "</html>"
```