Awesome! Here’s the full ready-to-go setup for your Fedora iwlwifi Wi-Fi with CAKE QoS, kernel tweaks, power save off, and systemd services for auto-apply on boot and resume.

1. Kernel module options for iwlwifi

Create this file:

sudo tee /etc/modprobe.d/iwlwifi.conf > /dev/null <<EOF

options iwlwifi power\_save=0

options iwlwifi 11n\_disable=8

EOF

Apply immediately (rebuild initramfs and reboot):

sudo dracut --force

sudo reboot

2. Disable Wi-Fi power saving in NetworkManager for your connection (replace OPTUS\_DADS\_5GHz with your actual connection name if different):

nmcli connection modify OPTUS\_DADS\_5GHz 802-11-wireless.powersave 2

nmcli connection up OPTUS\_DADS\_5GHz

3. CAKE QoS Script — create /usr/local/bin/cake-wifi.sh

sudo tee /usr/local/bin/cake-wifi.sh > /dev/null <<'EOF'

#!/bin/bash

IF=wlp3s0

DOWN=50000 # Set your real download bandwidth (in kbit)

UP=10000 # Set your real upload bandwidth (in kbit)

# Delete existing qdisc if any

tc qdisc del dev "$IF" root 2>/dev/null

# Apply CAKE qdisc with besteffort and ethernet overhead for Wi-Fi

tc qdisc add dev "$IF" root cake bandwidth "${DOWN}kbit" besteffort overhead ethernet

EOF

Make it executable:

sudo chmod +x /usr/local/bin/cake-wifi.sh

4. Systemd service to apply CAKE at boot

Create /etc/systemd/system/cake-wifi.service:

sudo tee /etc/systemd/system/cake-wifi.service > /dev/null <<EOF

[Unit]

Description=Apply CAKE QoS on Wi-Fi interface

After=network-online.target

Wants=network-online.target

[Service]

Type=oneshot

ExecStart=/usr/local/bin/cake-wifi.sh

[Install]

WantedBy=multi-user.target

EOF

Enable & start it now:

sudo systemctl daemon-reload

sudo systemctl enable --now cake-wifi.service

5. Systemd service to reapply CAKE after resume from suspend

Create /etc/systemd/system/cake-wifi-resume.service:

sudo tee /etc/systemd/system/cake-wifi-resume.service > /dev/null <<EOF

[Unit]

Description=Reapply CAKE QoS after suspend/resume

After=suspend.target

[Service]

Type=oneshot

ExecStart=/usr/local/bin/cake-wifi.sh

[Install]

WantedBy=suspend.target

EOF

Enable it:

sudo systemctl enable cake-wifi-resume.service

6. DNS tweak: Use Cloudflare + Google DNS

nmcli connection modify OPTUS\_DADS\_5GHz ipv4.dns "1.1.1.1,8.8.8.8"

nmcli connection modify OPTUS\_DADS\_5GHz ipv4.dns-priority 100

nmcli connection up OPTUS\_DADS\_5GHz

7. sysctl low-latency network tweaks

Create /etc/sysctl.d/99-network-tweaks.conf:

sudo tee /etc/sysctl.d/99-network-tweaks.conf > /dev/null <<EOF

# TCP Low Latency Tuning

net.core.netdev\_max\_backlog=5000

net.core.rmem\_max=16777216

net.core.wmem\_max=16777216

net.ipv4.tcp\_low\_latency=1

net.ipv4.tcp\_mtu\_probing = 1 # Enable Path MTU Discovery

net.ipv4.tcp\_rmem=4096 87380 16777216

net.ipv4.tcp\_slow\_start\_after\_idle = 0 # Disable TCP slow start after idle

net.ipv4.tcp\_window\_scaling = 1 # Enable TCP window scaling

net.ipv4.tcp\_wmem=4096 65536 16777216

vm.page-cluster = 0 # Disable page clustering for filesystems

vm.swappiness = 10 # Swappiness parameter (tendency to swap out unused pages)

vm.vfs\_cache\_pressure = 50 # Controls the tendency of the kernel to reclaim the memory used for caching of directory and inode objects

EOF

Apply immediately:

sudo sysctl --system

DONE!

This will give you:

Wi-Fi kernel driver optimizations disabling power save

NetworkManager disables Wi-Fi power saving for your connection

CAKE QoS shaping on your Wi-Fi for bufferbloat + low latency

CAKE auto-applies after boot and suspend/resume

Fast DNS configured

Kernel TCP buffers tuned for low latency

If you want, I can help you test or improve this further — just ask!

#!/bin/bash

# Detect main network interface (default route)

interface=$(ip route get 1 | awk '{print $5; exit}')

if [[ -z "$interface" ]]; then

echo "Could not detect network interface"

exit 1

fi

# Ping 8.8.8.8 to measure RTT (ms)

rtt=$(ping -c 3 8.8.8.8 | tail -1 | awk -F '/' '{print $5}')

if [[ -z "$rtt" ]]; then

rtt=20

fi

# Run a quick speedtest-cli (requires speedtest-cli installed)

# Only get upload speed (Mbit)

upload=$(speedtest-cli --simple | grep Upload | awk '{print $2}')

if [[ -z "$upload" ]]; then

echo "Speedtest failed, defaulting upload bandwidth to 20 Mbit"

upload=20

fi

# Convert upload to integer (round)

upload\_int=$(printf "%.0f" "$upload")

# Wi-Fi overhead (typical for 802.11ac/ax)

overhead=94

echo "Applying CAKE qdisc on interface $interface"

echo "RTT: ${rtt} ms"

echo "Upload bandwidth: ${upload\_int} Mbit"

echo "Using overhead: $overhead bytes"

sudo tc qdisc replace dev "$interface" root cake bandwidth ${upload\_int}Mbit besteffort diffserv4 triple-isolate nat nowash ack-filter split-gso rtt ${rtt}ms overhead $overhead

echo "CAKE qdisc applied successfully."

Got it — you want a single meta systemd service that can start, stop, and manage all your LinuxTweaks-related services (like the CAKE qdisc services, ntp-check, autostart, etc.) together, so you don’t have to juggle them individually.

Here’s a straightforward way to do this:

1. Create a systemd target unit: linuxtweaks.target

This will act as a group for all your related services.

Create /etc/systemd/system/linuxtweaks.target with:

[Unit]

Description=LinuxTweaks meta target for all tweak services

Requires=linuxtweaks-cake.service linuxtweaks-cake-resume.service linuxTweaks-autostart.service ntp-check.service

After=network.target

2. Make sure all individual services are WantedBy=linuxtweaks.target

For each service, create or edit a drop-in override so it’s linked to the target. For example:

sudo systemctl edit linuxtweaks-cake.service

Add:

[Install]

WantedBy=linuxtweaks.target

Do this for:

linuxtweaks-cake.service

linuxtweaks-cake-resume.service

linuxTweaks-autostart.service

ntp-check.service

3. Enable the target and services together

Enable the target:

sudo systemctl enable linuxtweaks.target

Then start or stop all services with:

sudo systemctl start linuxtweaks.target

sudo systemctl stop linuxtweaks.target

Bonus: You can even make the target start at boot:

sudo systemctl enable linuxtweaks.target

Summary:

linuxtweaks.target groups your tweak services.

Starting/stopping/enabling the target controls all tweak services at once.

This keeps things neat and manageable.

 interface=$(ip route get 1 | awk '{print $5; exit}')   
 ethtool -i "$interface"   
 iw dev "$interface" link

 sudo tee /etc/modprobe.d/iwlwifi.conf > /dev/null <<EOF   
 options iwlwifi power\_save=0   
 options iwlwifi 11n\_disable=8   
 EOF   
 sudo dracut --force   
 sudo reboot

 check2   
 nmcli connection modify OPTUS\_DADS\_5GHz 802-11-wireless.powersave 2   
 nmcli connection up OPTUS\_DADS\_5GHz   
 nmcli connection modify OPTUS\_DADS\_5GHz ipv4.dns "1.1.1.1,8.8.8.8"   
 nmcli connection modify OPTUS\_DADS\_5GHz ipv4.dns-priority 100   
 nmcli connection up OPTUS\_DADS\_5GHz

 sudo tee /etc/sysctl.d/99-network-tweaks.conf > /dev/null <<EOF   
 net.ipv4.tcp\_low\_latency=1   
 net.core.netdev\_max\_backlog=5000   
 net.core.rmem\_max=16777216   
 net.core.wmem\_max=16777216   
 net.ipv4.tcp\_rmem=4096 87380 16777216   
 net.ipv4.tcp\_wmem=4096 65536 16777216   
 EOF

 sudo sysctl --system   
 ip link show wlp3s0 | grep mtu   
 speedtest-cli --simple