CAKE QoS Configuration – Calculation and Rationale

✅ System Details

| Parameter | Value |
| --- | --- |
| Modem Model | F@ST 3864V3HP |
| ISP | Optus (Australia) |
| Connection Type | VDSL2 over PTM |
| Encapsulation | PPPoE |
| Upload Sync Rate | 22,600 Kbps |
| Download Sync Rate | 60,104 Kbps |

Bandwidth Calculation

wlp3s0)

CAKE shapes the interface it’s applied to — in this case, the Wi-Fi interface on the client machine (outgoing/upload).

Actual Sync Rate:

22,600 Kbps = 22.6 Mbps

Target shaping (95%):

22.6 Mbps × 0.95 = 21.47 Mbps

bandwidth 21Mbit

Overhead Calculation

✉️ PTM + PPPoE Overhead

| Layer | Bytes |
| --- | --- |
| PTM Framing | 22 |
| PPPoE Header | 8 |
| Ethernet II Header | 14 |
| Total | 44 |

overhead 44

This ensures CAKE accounts for the full per-packet overhead introduced by the DSL framing and encapsulation.

⚙️ Full tc Command

tc qdisc replace dev wlp3s0 root cake \

bandwidth 21Mbit \

diffserv4 \

triple-isolate \

nat \

nowash \

ack-filter \

split-gso \

rtt 25ms \

overhead 44

Explanation:

| Option | Purpose |
| --- | --- |
| bandwidth 21Mbit | Sets shaping rate slightly below actual line rate to prevent bufferbloat |
| diffserv4 | Enables 4-class QoS (Bulk, Best Effort, Video, Voice) |
| triple-isolate | Fairness across flows, internal hosts, and IPs |
| nat | Required if NAT is active on the host (e.g., home router behind modem) |
| nowash | Retains existing DSCP marks (optional, else CAKE resets them) |
| ack-filter | Drops redundant TCP ACKs to save bandwidth |
| split-gso | Handles large packets fairly |
| rtt 25ms | RTT estimate (25ms is a good starting point for AU ISPs) |
| overhead 44 | Accounts for VDSL2/PTM+PPPoE packet framing |

Linux cannot directly shape incoming traffic, but IFB (Intermediate Functional Block) or ingress qdisc + mirroring can be used.

However, it's often better to shape upload only and use SQM on the modem/router (if supported). The F@ST3864V3HP has limited QoS support unless flashed with third-party firmware (not usually possible on Optus firmware).

Here’s a complete version of your CAKE QoS documentation in Markdown format. You can copy this into a README, webpage, Git repo, or use it in your RPM documentation.

#

22.6 Mbps × 0.95 = 21.47 Mbps → use: bandwidth 21Mbit

---

##

overhead 44

---

##

⚙️ Option Breakdown

| Option | Description |
| --- | --- |
| bandwidth 21Mbit | Sets the shaping rate below your actual line rate to prevent bufferbloat |
| diffserv4 | Enables 4-class DiffServ: Bulk, Best Effort, Video, Voice |
| triple-isolate | Ensures fairness across flows, internal hosts, and IPs |
| nat | Enables NAT awareness (important for most home routers) |
| nowash | Preserves DSCP marks; disables default CAKE reset behavior |
| ack-filter | Drops redundant TCP ACKs to save bandwidth |
| split-gso | Prevents large packets from causing unfair scheduling |
| rtt 25ms | Round-trip time estimate (Australia typical RTT to ISP is ~20–30ms) |
| overhead 44 | Accounts for actual per-packet DSL framing + PPPoE encapsulation |

⏬ Ingress (Download) Shaping

Linux can't shape download traffic directly. Options:

Use ifb with ingress mirroring.

Prefer shaping uploads only (practical).

Apply shaping on modem/router (if supported, usually not on locked Optus firmware).

tc -s qdisc show dev wlp3s0)

qdisc cake 8001: root refcnt 2 bandwidth 21Mbit diffserv4 triple-isolate nat nowash ack-filter split-gso rtt 25ms raw overhead 44

Sent 31134535 bytes 108864 pkt (dropped 2, overlimits 7151 requeues 0)

backlog 0b 0p requeues 0

memory used: 23400b of 5000000b

capacity estimate: 21Mbit

Create a systemd service: /etc/systemd/system/linuxtweaks-cake.service

[Unit]

Description=Apply CAKE qdisc to wlp3s0 at boot

After=network-online.target

Wants=network-online.target

[Service]

Type=oneshot

ExecStart=/usr/sbin/tc qdisc replace dev wlp3s0 root cake bandwidth 21Mbit diffserv4 triple-isolate nat nowash ack-filter split-gso rtt 25ms overhead 44

RemainAfterExit=yes

[Install]

WantedBy=multi-user.target

Enable it:

sudo systemctl daemon-reexec

sudo systemctl enable --now linuxtweaks-cake.service

For suspend-resume support, also create linuxtweaks-cake-resume.service.

[CAKE: Common Applications Kept Enhanced](https://www.bufferbloat.net/projects/codel/wiki/Cake/)

[OpenWRT SQM Guide](https://openwrt.org/docs/guide-user/network/traffic-shaping/sqm)

[tc-cake man page](https://man7.org/linux/man-pages/man8/tc-cake.8.html)

✅ This config helps reduce bufferbloat, improve VoIP/gaming quality, and ensure fair bandwidth use on LAN clients.

iptables marks and CAKE's DiffServ (DiffServ4 or DiffServ8) classes.

Prioritize printer traffic (like Samba to 192.168.0.1 or IPP) so that CAKE ensures faster, smoother delivery even under load.

iptables

Assuming your printer is at 192.168.0.1, and you're using Samba or IPP:

iptables marks and CAKE's DiffServ (DiffServ4 or DiffServ8) classes.

Prioritize printer traffic (like Samba to 192.168.0.1 or IPP) so that CAKE ensures faster, smoother delivery even under load.

iptables

Assuming your printer is at 192.168.0.1, and you're using Samba or IPP:

tolga@fedora:~$ sudo lpadmin -p hp601-fast -E -v ipp://192.168.0.7/ipp/print -m everywhere   
tolga@fedora:~$ lpstat -p   
lp -d hp601-fast /etc/hosts   
printer hp601-fast is idle.  enabled since Wed 14 May 2025 23:40:22   
printer HP\_LaserJet\_600\_M601 is idle.  enabled since Wed 14 May 2025 23:29:10   
request id is hp601-fast-15 (1 file(s))   
tolga@fedora:~$ sudo nano /etc/hosts   
tolga@fedora:~$ ipp://hp-m601/ipp/print   
bash: ipp://hp-m601/ipp/print: No such file or directory   
tolga@fedora:~$ ipp://hp-m601   
bash: ipp://hp-m601: No such file or directory   
tolga@fedora:~$ ping hp-m601

#!/bin/bash

# CONFIGURATION

PRINTER\_NAME="hp601-fast"

PRINTER\_IP="192.168.0.7"

PRINTER\_URI="ipp://$PRINTER\_IP/ipp/print"

TEST\_FILE="/etc/hosts"

echo "🔧 Checking for CUPS..."

if ! command -v lpadmin &>/dev/null; then

echo "❌ CUPS (lpadmin) not found. Please install 'cups' first."

exit 1

fi

echo "🧹 Removing any old HP LaserJet printer (if exists)..."

OLD\_PRINTER=$(lpstat -v | grep -i "HP\_LaserJet\_600" | awk '{print $3}')

if [ -n "$OLD\_PRINTER" ]; then

sudo lpadmin -x "$OLD\_PRINTER"

echo "✅ Removed old printer: $OLD\_PRINTER"

else

echo "ℹ️ No old auto-discovered printer found."

fi

echo "➕ Adding printer '$PRINTER\_NAME' at $PRINTER\_URI"

sudo lpadmin -p "$PRINTER\_NAME" -E -v "$PRINTER\_URI" -m everywhere

if [ $? -eq 0 ]; then

echo "✅ Printer added successfully."

else

echo "❌ Failed to add printer."

exit 1

fi

echo "⭐ Setting '$PRINTER\_NAME' as default printer..."

sudo lpoptions -d "$PRINTER\_NAME"

echo "🧪 Sending test page: $TEST\_FILE"

lp -d "$PRINTER\_NAME" "$TEST\_FILE"

echo "✅ All done! Your HP LaserJet 600 M601 is ready to print."

sudo lpadmin -p "FAST HP LaserJet 600 M601" -D "hp601-fast"

sudo lpadmin -p "hp601-fast" -D "FAST\_HP\_LaserJet\_600\_M601"

sudo lpadmin -p FAST\_HP\_LaserJet\_600\_M601 -E -v ipp://192.168.0.7/ipp/print -m everywhere

#!/usr/bin/env bash

# LinuxTweaks CAKE Auto Configuration Script (JSON-based)

# Author: Tolga

# More accurate: Parses speedtest-cli JSON output

# Requirements: speedtest-cli, jq, tc, iproute2

echo "[LinuxTweaks CAKE] Starting CAKE auto-config..."

# Run speedtest-cli in JSON mode and grab upload in bits per second

json\_output=$(speedtest-cli --json)

if [[ -z "$json\_output" ]]; then

echo "[ERROR] Speedtest failed or returned empty result."

exit 1

fi

# Extract upload speed in bits/s

upload\_bps=$(echo "$json\_output" | jq '.upload')

if [[ -z "$upload\_bps" || "$upload\_bps" == "null" ]]; then

echo "[ERROR] Failed to extract upload speed."

exit 1

fi

# Calculate 90% of upload in kbit

upload\_kbit=$(awk "BEGIN {printf \"%.0f\", $upload\_bps \* 0.90 / 1000}")

upload\_mbps=$(awk "BEGIN {printf \"%.2f\", $upload\_bps / 1000000}")

echo "[INFO] Upload (raw): $upload\_bps bit/s"

echo "[INFO] Upload: $upload\_mbps Mbps → Applying 90%: ${upload\_kbit} kbit"

# Detect active outbound interface

active\_if=$(ip route get 1.1.1.1 2>/dev/null | awk '{for(i=1;i<=NF;i++) if ($i=="dev") print $(i+1)}')

if [[ -z "$active\_if" ]]; then

echo "[ERROR] Could not detect default network interface."

exit 1

fi

echo "[INFO] Detected active interface: $active\_if"

# Clean any existing qdisc

tc qdisc del dev "$active\_if" root 2>/dev/null

# Apply CAKE qdisc to upload (egress)

echo "[INFO] Applying CAKE..."

tc qdisc add dev "$active\_if" root cake bandwidth "$upload\_mbpsMbit" besteffort

# Check success

if [[ $? -eq 0 ]]; then

echo "[SUCCESS] CAKE applied to $active\_if with bandwidth: ${upload\_kbit} kbit (90% of $upload\_mbps Mbps)"

else

echo "[ERROR] Failed to apply CAKE."

fi

#!/usr/bin/env bash

# LinuxTweaks CAKE Auto Configuration Script using Mbit and RTT

# Dependencies: speedtest-cli, jq, tc

echo "[LinuxTweaks CAKE] Starting CAKE auto-config..."

# Run speedtest and get JSON

json\_output=$(speedtest-cli --json)

if [[ -z "$json\_output" ]]; then

echo "[ERROR] Speedtest failed or returned empty result."

exit 1

fi

# Extract download, upload (in bits/s), and ping (in ms)

upload\_bps=$(echo "$json\_output" | jq '.upload')

download\_bps=$(echo "$json\_output" | jq '.download')

ping\_latency=$(echo "$json\_output" | jq '.ping')

# Safety checks

if [[ -z "$upload\_bps" || "$upload\_bps" == "null" || -z "$download\_bps" || "$download\_bps" == "null" ]]; then

echo "[ERROR] Failed to extract speed values."

exit 1

fi

# Convert to Mbit and apply 90% safety margin

optimal\_upload=$(awk "BEGIN {printf \"%.2f\", $upload\_bps \* 0.90 / 1000000}")

optimal\_download=$(awk "BEGIN {printf \"%.2f\", $download\_bps \* 0.90 / 1000000}")

ping\_latency=$(awk "BEGIN {printf \"%.0f\", $ping\_latency}")

echo "[INFO] Optimal Upload: ${optimal\_upload} Mbit"

echo "[INFO] Optimal Download: ${optimal\_download} Mbit"

echo "[INFO] RTT (Ping): ${ping\_latency} ms"

# Detect active network interface

interface=$(ip route get 1.1.1.1 2>/dev/null | awk '{for(i=1;i<=NF;i++) if ($i=="dev") print $(i+1)}')

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

echo "[ERROR] Could not detect active network interface."

exit 1

fi

echo "[INFO] Active Interface: $interface"

# Remove existing CAKE qdisc if any

#tc qdisc del dev "$interface" root 2>/dev/null || true

get\_overhead() {

local iface="$1"

local iface\_type

# Check if interface exists

if [[ ! -d /sys/class/net/"$iface" ]]; then

echo "Error: Interface $iface does not exist."

exit 1

fi

# Get interface type (numeric code)

iface\_type=$(cat /sys/class/net/"$iface"/type 2>/dev/null)

# Detect WWAN/LTE interfaces by common names or driver presence

if [[ "$iface" =~ ^wwan[0-9]\*$ || "$iface" =~ ^wwp[0-9]\*$ ]]; then

echo 94

return

fi

# Check if interface is a wireless device (check for wireless directory)

if [[ -d /sys/class/net/"$iface"/wireless ]]; then

echo 94

return

fi

# Ethernet check + PPPoE detection

case "$iface\_type" in

1)

# Check PPPoE (ppp interfaces present)

if ip link show | grep -q ppp; then

echo 44

else

echo 18

fi

;;

512)

# loopback, no overhead needed

echo 0

;;

\*)

# Unknown interface type: default to wifi-like overhead

echo 94

;;

esac

}

overhead=$(get\_overhead "$interface")

echo "Applying TC Cake qdisc to interface: $interface"

echo "Bandwidth: ${optimal\_download}Mbit"

echo "RTT: ${ping\_latency}ms"

echo "Calculated overhead: ${overhead} bytes"

# Apply CAKE to the interface with full config

echo "[INFO] Applying CAKE..."

#sudo tc qdisc replace dev "$interface" root cake \

# bandwidth "${optimal\_download}Mbit" \

# diffserv4 besteffort triple-isolate nat nowash ack-filter split-gso \

# rtt "${ping\_latency}ms" \

# overhead "$overhead"

if [[ $? -eq 0 ]]; then

echo "[SUCCESS] CAKE applied with:"

echo " ▪ Interface: $interface"

echo " ▪ Bandwidth: ${optimal\_download} Mbit"

echo " ▪ RTT: ${ping\_latency} ms"

else

echo "[ERROR] Failed to apply CAKE."

fi

Setting CAKE's upload bandwidth to 90% of your real upstream is not necessarily too aggressive, but it depends on your connection quality, ISP behavior, and buffer sizes.

Here’s a breakdown to help you tune it smartly:

What CAKE needs

CAKE (and fq\_codel before it) works best when you avoid saturating your upstream link. This allows CAKE to manage and prioritize packets before they hit the bottleneck (e.g., your ISP's modem queue).

✅ General Recommendations

| Scenario | Upload Limit Recommendation |
| --- | --- |
| Very stable connection (fiber, no congestion) | 90–95% is fine |
| Cable/DSL with shared bandwidth or inconsistent latency | 85–90% is safer |
| High bufferbloat/modem queueing issues | 80–85% is often needed |

You see high latency under load (e.g., ping 1.1.1.1 spikes during upload).

You still experience bufferbloat despite CAKE.

Speedtest upload hits the exact limit and causes noticeable delay in real-time apps (Zoom, gaming, etc.).

Use the tool [Waveform Bufferbloat Test](https://www.waveform.com/tools/bufferbloat) or run:

ping 1.1.1.1 -i 0.2 -s 100 -D

...while uploading a file (e.g., to Google Drive). If your ping spikes > 40ms over idle, CAKE isn’t doing its job well — drop the bandwidth to 85%.

Start with 90% (like you're doing).

Test under load.

If latency spikes or jitter occurs, drop it in 5% increments until stable.

Example:

If your real upstream is 10 Mbit:

Try bandwidth 9Mbit (90%) first.

If jitter or latency > 30–40ms under load, try 8.5Mbit or 8Mbit.

Let me know your real upload speed and whether you're on DSL, NBN, cable, etc., and I can give a tailored percentage.