

# Project IPv6-First

A Case Study in Achieving an 80%+ Native  
IPv6 SOHO Network

Terry Sweetser

#apricot2026  
**APRICOT 2026**  
APNIC 61

APNIC IPv6 Deployment

# The Objective

**Is a near-native IPv6-first environment achievable today?**

- **Goal:** Maximize the native IPv6 traffic ratio on a sophisticated SOHO network.
- **Method:** Data-driven analysis using NetFlow (Akvorado).
- **Challenge:** Identify and remediate residual IPv4 traffic sources without using transition technologies like CLAT (yet).

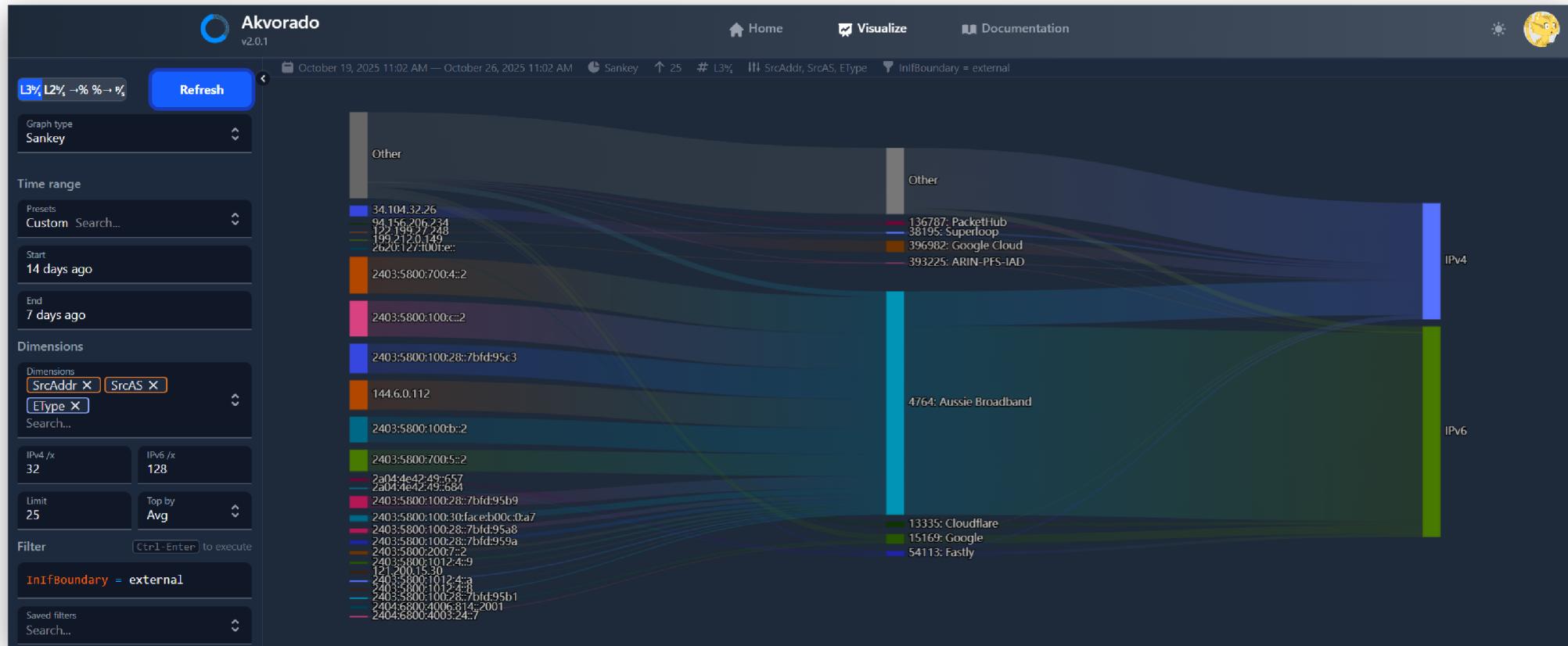
# The Baseline Environment

Starting from a robust dual-stack foundation:

- **ISP:** Aussie Broadband (Native IPv6 /48).
- **Gateway:** MikroTik (RouterOS).
- **DNS:** Public DNS Resolver 1.1.1.1
- **Philosophy:** Local services running on planned IPv6 addresses  
(e.g., `::cafe`, `::beef`).

```
pi@raspberrypi:~ $ avahi-resolve -n mediabox.local
mediabox.local 240[REDACTED]26::cafe
pi@raspberrypi:~ $ avahi-resolve -n lab.local
lab.local      240[REDACTED]26::beef
```

# Data: The Starting Point



(Ref: Akvorado Sankey Chart)

# The Problem

Despite this "perfect" setup, historical analysis showed the network was only achieving **67.7% IPv6 traffic**.

# Methodology: The "Find and Fix" Loop

1. **Establish Baseline:** Analyze destination ports and protocols via Akvorado.
2. **Isolate Laggards:** Identify high-bandwidth applications defaulting to IPv4.
3. **Targeted Intervention:** Reconfigure applications to force IPv6 compliance.
4. **Validate:** Measure the "After" state.

# The First Culprit: BitTorrent

## Initial Status:

BitTorrent was a significant drag on the network, operating at only  
**44% IPv6.**

## The "Working" Dual-Stack Trap:

- The client was bound to "Any Interface."
- It accepted IPv4 peer connections from trackers immediately.
- **Result:** It defaulted to a suboptimal IPv4 NAT path rather than waiting for superior IPv6 peers.

# Results: The Shift

## Application Level:

- BitTorrent traffic flipped from 44% to **ALL IPv6**.

## Network Level:

- The structural change lifted the entire network's stable operational average from ~67% to **79.2%**.

“ How much would an ISP save on NAT444 at a ratio of 80% ipv6 traffic? ”

Peer Statistics										
Country	IP	Port	Connections	Flags	Client	Progress	Download Speed	Up Speed	Downloaded	
Australia	2400:a842:7a31:0:53d7:1556:5d80:cb18	30010	μTP	D X E P	qBittorrent/5.1.4	100%	415.1 KiB/s		41.2 MiB	
Australia	2405:6e00:494:56dd:1b:8736:e0af:b117	30664	μTP	D ? E P	qBittorrent/5.0.5	100%	273.0 KiB/s		17.3 MiB	
Venezuela	2800:484:9b36:4000:a37c:183c:c3df:814a	51413	μTP	D ? X E P	Transmission 2.84+	100%	190.1 KiB/s		5.2 MiB	
USA	2603:6013:c6f0:85b0:ad20:325:2e08:8c72	24778	μTP	D E P	qBittorrent/5.1.1	100%	168.2 KiB/s		4.5 MiB	
Czech Republic	2a00:102a:5004:46d2:1:0:c88c:c244	45399	BT	D I X E	libtorrent/1.2.19.0	100%	142.8 KiB/s		2.3 MiB	
Turkey	2407:d000:f:4266:e4b2:e0ff:fea8:9906	34060	μTP	D ? X E P	libtorrent/1.2.2.0	100%	133.9 KiB/s		4.5 MiB	
USA	2607:fb91:2630:cb2b:9072:2340:47c8:5c48	51413	μTP	D ? I X E P	Transmission 4.0.5	100%	129.5 KiB/s		1.7 MiB	
Brazil	2804:10f8:457e:e400:bf2a:10d3:27c9:61bf	9482	μTP	D E P	qBittorrent/5.1.4	100%	96.8 KiB/s		5.1 MiB	
France	2a01:e0a:9bc:6d00:143f:7bff:fe7e:16de	35947	μTP	D ? X E P	libtorrent/1.2.2.0	100%	92.1 KiB/s		1.5 MiB	
Philippines	2001:4454:7f0:8d00:49f:1195:f43:f55c	6882	μTP	D ? X E P	libtorrent/1.2.2.0	100%	91.6 KiB/s		4.3 MiB	
Philippines	2a01:36d:1000:8ec8:5571:d50a:96b8:ce35	31881	μTP	D ? S X E P	qBittorrent/4.5.1	100%	88.1 KiB/s		3.0 MiB	
Philippines	2001:4451:84c6:b00:d4e2:ab5:74c:9081	22118	μTP	D ? S X E P	qBittorrent/5.0.0	100%	84.8 KiB/s		11.2 MiB	
UK	2a00:23c7:7f9c:7201::ffb	6881	μTP	D ? X E P	libtorrent/1.2.2.0	100%	77.7 KiB/s		1.7 MiB	

# Local DNS & Happy Eyeballs

The "IPv6-First" success relies on a local DNS stack (Pi-hole + Unbound) that optimizes the **Happy Eyeballs v2 (RFC 8305)** race.

## 1. The "Resolution Delay" (50ms):

- **Standard:** RFC 6724 sees to it that all `AAAA` records are ranked first.
- **Local Edge:** Unbound resolves in **microseconds**. The `AAAA` record never arrives "late," so the browser never defaults to IPv4 during lookup.

## 2. The "Connection Head Start" (~250ms):

- Because the IPv6 address is available instantly, the OS starts the IPv6 connection immediately.
- It waits **~250ms** before attempting IPv4.
- **Result:** IPv6 runs unopposed for a quarter of a second.

**Key Takeaway:** Local DNS resolution removes any "latency penalty" from the race, ensuring IPv6 is not just **preferred**, but mathematically **destined** to win.

# Update: Breaking the 90% Barrier

Recent SQL analysis (Dec 7 – Dec 22, 2025) shows continued optimization has pushed the network even further.

Metric	Long Term (Oct-Dec)	Short Term (Dec 7-22)
Total Flows	3.3M	28.1M
IPv6 Bytes %	81.19%	90.74%

“ The network is now operating as an effectively IPv6-Native environment. ”

# Visualizing the Data

```
acbd9d624098 :) select * from report_long_term;

SELECT *
FROM report_long_term

Query id: 52973f2c-2873-4cc9-86cb-ea8fe17ac6ef



|    | protocol | total_flows | inbound_gb        | flow_percent | bytes_percent | first_seen          | last_seen           |
|----|----------|-------------|-------------------|--------------|---------------|---------------------|---------------------|
| 1. | IPv4     | 1137319     | 690.0473929345608 | 34.26        | 18.81         | 2025-10-11 02:00:00 | 2025-12-22 10:00:00 |
| 2. | IPv6     | 2182609     | 2977.681441391818 | 65.74        | 81.19         | 2025-10-11 02:00:00 | 2025-12-22 10:00:00 |
| 3. | TOTAL    | 3319928     | 3667.728834326379 | 100          | 100           | 2025-10-11 02:00:00 | 2025-12-22 10:00:00 |



3 rows in set. Elapsed: 0.050 sec. Processed 13.44 million rows, 201.61 MB (266.87 million rows/s., 4.00 GB/s.)
Peak memory usage: 245.47 KiB.

acbd9d624098 :) select * from report_short_term

SELECT *
FROM report_short_term

Query id: d83a9e0f-de69-4e6b-8d8e-488a4dfd04ae



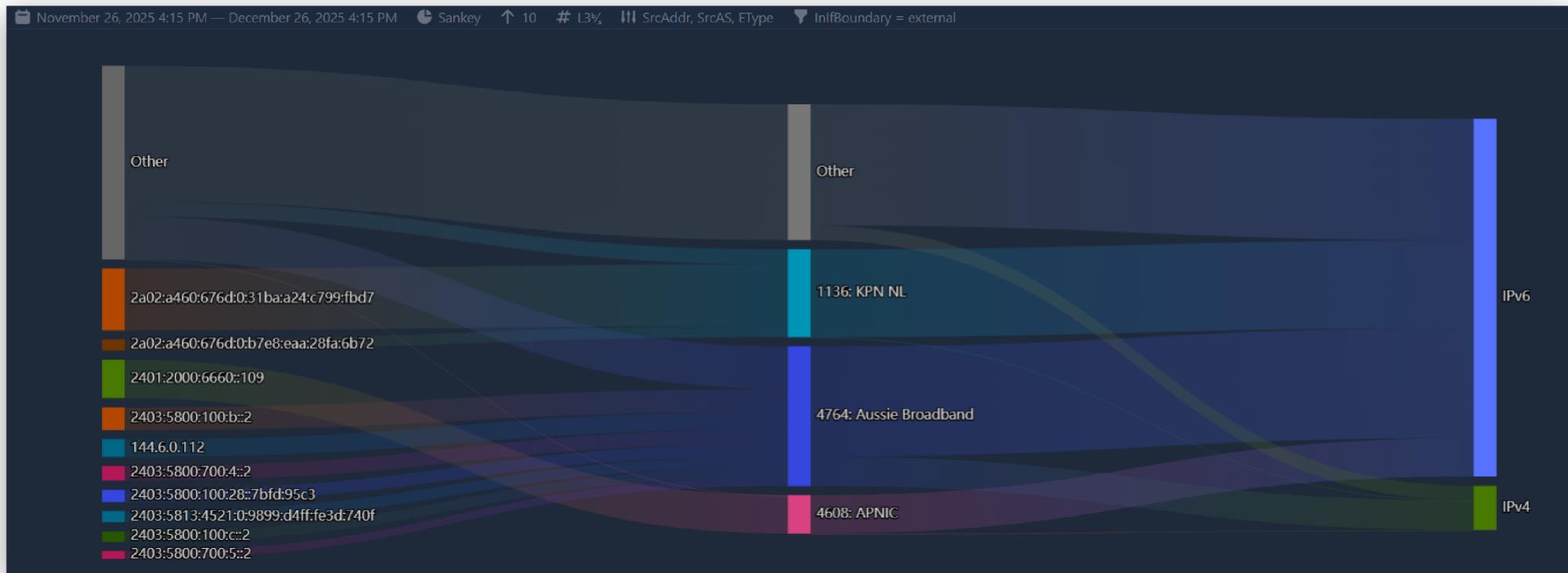
|    | protocol | total_flows | inbound_gb        | flow_percent | bytes_percent | first_seen          | last_seen           |
|----|----------|-------------|-------------------|--------------|---------------|---------------------|---------------------|
| 1. | IPv4     | 1404713     | 90.24691123887897 | 4.99         | 9.26          | 2025-12-07 08:35:16 | 2025-12-22 10:11:49 |
| 2. | IPv6     | 26760060    | 884.683262550272  | 95.01        | 90.74         | 2025-12-07 08:35:15 | 2025-12-22 10:11:51 |
| 3. | TOTAL    | 28164773    | 974.930173789151  | 100          | 100           | 2025-12-07 08:35:15 | 2025-12-22 10:11:51 |



3 rows in set. Elapsed: 0.351 sec. Processed 114.39 million rows, 1.71 GB (326.07 million rows/s., 4.87 GB/s.)
Peak memory usage: 8.09 MiB.
```

(Ref: Clickhouse SQL Analysis confirming >90% IPv6 byte ratio)

# Visualizing the Data



(Ref: Akvorado Sankay Charts)

# Macro vs. Micro: Validating the Data

My SOHO network mirrors global adoption curves, suggesting that ~80% is the current "natural limit" for standard dual-stack implementation.

Context	IPv6 Ratio	Status
Germany (National)	66.67%	Strong legacy infrastructure drag
My Network (Baseline)	67.70%	Default configs favoring IPv4
India (National)	78.90%	Mobile-first / Greenfields architecture
France (National)	79.23%	Aggressive ISP migration
My Network (Optimized)	79.20%	Active deprecation of IPv4 paths
My Network (Dec '25)	90.74%	The "IPv6-Native" Frontier

## Key Insight:

*We can manually achieve today what the global internet would naturally reach in ~5 years.*

# Defining the "IPv4 Floor"

The remaining ~10% of IPv4 traffic is irreducible without CLAT/NAT64. It consists of:

1. **Legacy IoT:** Ring cameras (IPv4-only video upload).
2. **Legacy Web Services:**
  - Amazon.com (Main retail site).
  - GitHub.com (Main site, though assets are v6).
3. **ISP Infrastructure:** Aussie Broadband caching/accelerators.

“ Or we can wait for these services to deploy ipv6 ... ”

# DNS-Aware Dynamic EAM "The Idea"

Standard 464XLAT double-translates traffic (IPv4 → IPv6 → IPv4).  
For the "Ring Camera" problem (IPv4 device → IPv6 API), we can do better.

## The Prototype: Unbound + Jool (SIIT)

I am proposing a dynamic extension to [RFC 7757 \(EAM\)](#):

1. **Detect:** Daemon monitors local DNS (Unbound) for A + AAAA pairs.
2. **Map:** Dynamically populates the SIIT translator table.
3. **Result:** Legacy IPv4 devices connect **directly** to IPv6 APIs.

“ **Impact:** This removes the "Vendor Negligence" traffic (e.g., Ring) from the IPv4 floor without waiting for firmware updates. ”

# A Call to Action

We need 464XLAT on more platforms, including PLAT and CLAT on Mikrotik.



## We Need Your Voice

Support Feature Request [#263487](#) to bring native 464XLAT to RouterOS.

[forum.mikrotik.com/t/feature-request-464xlat/263487](http://forum.mikrotik.com/t/feature-request-464xlat/263487)

# January 2026 Data



“ So, yes, we can live  
without ipv4 ... ”



<https://bit.ly/3L4clnw>

