P5 Tutorial: Multi-STA Fairness and Goodput Analysis (ns-3 Wi-Fi)

This tutorial explains how to study *fairness* and *goodput* in a multi-station (multi-STA) Wi-Fi scenario in ns-3, and how to interpret the resulting plots and CSV outputs. The goal is not just to "produce charts", but to understand *why* the curves and bars look the way they do and what they imply about the MAC layer, rate adaptation, and transport protocols (UDP vs. TCP).

Scenario in one paragraph. The simulation contains an AP and multiple STAs on a Wi-Fi network (10.1.0.0/24). The AP connects to a server over a CSMA wired link (10.2.0.0/24). Each STA sends traffic to the server using a unique destination port. The experiment is repeated for different numbers of STAs (N) and for two transports: UDP (OnOff) and TCP (BulkSend). For each run, we record per-STA received bytes (from packet sinks), compute per-STA goodput, aggregate goodput, and Jain's fairness index. The key question is: *as contention grows with N, how do throughput and fairness change, and how does this differ between UDP and TCP?*

1. Metrics and why we use them

Goodput vs. throughput. In network experiments, *throughput* often means the total bits delivered to the receiver per unit time. *Goodput* is more specific: it counts only useful application-layer data delivered, excluding extra protocol overhead and (depending on measurement point) retransmissions. In this project, the application packet size is 1200 B. The sink receives application payload bytes, so the computed goodput is effectively the rate of useful bytes delivered to the application.

If STA i receives B_i bytes during the *useful time* window $T = (t_{\text{stop}} - t_{\text{start}})$, then

$$G_i = \frac{8B_i}{T} \quad \text{[bps]}, \qquad G_i^{(\text{Mbps})} = \frac{G_i}{10^6}. \tag{1}$$

The *sum goodput* is the aggregate:

$$G_{\text{sum}} = \sum_{i=1}^{N} G_i. \tag{2}$$

Fairness (Jain's index). A very common fairness metric in networking is Jain's fairness index:

$$J(\mathbf{G}) = \frac{\left(\sum_{i=1}^{N} G_i\right)^2}{N \sum_{i=1}^{N} G_i^2}, \qquad 0 < J \leq 1. \tag{3}$$

If all STAs receive exactly the same goodput, $G_1 = \cdots = G_N$, then $J = 1$ (perfect fairness). As goodput becomes more uneven, J decreases. This metric is especially useful because it summarizes distribution *shape* in a single number.

How to read J. Values close to 1 (e.g. 0.95–1.0) indicate near-equal sharing; values around 0.8 mean noticeable imbalance (some STAs obtain much more than others). Importantly, fairness does not imply high throughput: you can be very fair but slow (everyone gets little), or unfair but fast (some users dominate).

2. What outputs mean (CSV files) and how the plots are built

Two CSV outputs matter most here:

- `p5_summary.csv`: one row per experiment (transport, N, run). It contains aggregate goodput (`sumGoodputbps`) and fairness (`jain`), plus configuration fields like packet size and rate manager.
- `persta_{udp|tcp}_nN_runR.csv`: one file per experiment, containing STA ID, received bytes, and per-STA goodput.

Even though these look simple, they encode most of the experiment story. The line plots (Jain vs. N, Sum Goodput vs. N) come from `p5_summary.csv`. The bar charts (Per-STA Goodput) come from the per-STA files.

A quick numeric sanity-check

For UDP, $N = 2$, the summary line shows:

$$G_{\text{sum}} \approx 19.9906 \text{ Mbps}, \qquad J = 1.0.$$

From `persta_udp_n2_run1.csv`, each STA receives about 21.24 MB over $T = 17$ seconds (from $t_{\text{start}} = 3$ to $t_{\text{stop}} = 20$). For STA0, $B_0 \approx 21{,}236{,}400$ bytes, so

$$G_0 \approx \frac{8 \times 21{,}236{,}400}{17} \approx 9.9936 \text{ Mbps}.$$

STA1 is essentially the same, so the sum is about 19.99 Mbps and the fairness is effectively 1. This is exactly what we want: the numbers are internally consistent.

3. The key results (your dataset) in one table

Table 1 summarizes the main outcomes used in the plots.

Transport	N (STAs)	G_{sum} (Mbps)	Jain J
UDP	2	19.991	1.000000
UDP	5	43.091	0.997598
UDP	10	39.710	0.822121
TCP	2	31.778	0.996076
TCP	5	31.307	0.959864
TCP	10	30.403	0.949647

Table 1: Aggregate goodput and fairness from `p5_summary.csv` (converted to Mbps).

Two immediate patterns stand out: (1) UDP aggregate goodput *increases strongly* from $N = 2$ to $N = 5$, then *drops* at $N = 10$. (2) TCP aggregate goodput is *relatively stable* across N, but fairness slightly decreases as N grows.

The rest of the tutorial explains *why these patterns make sense.*

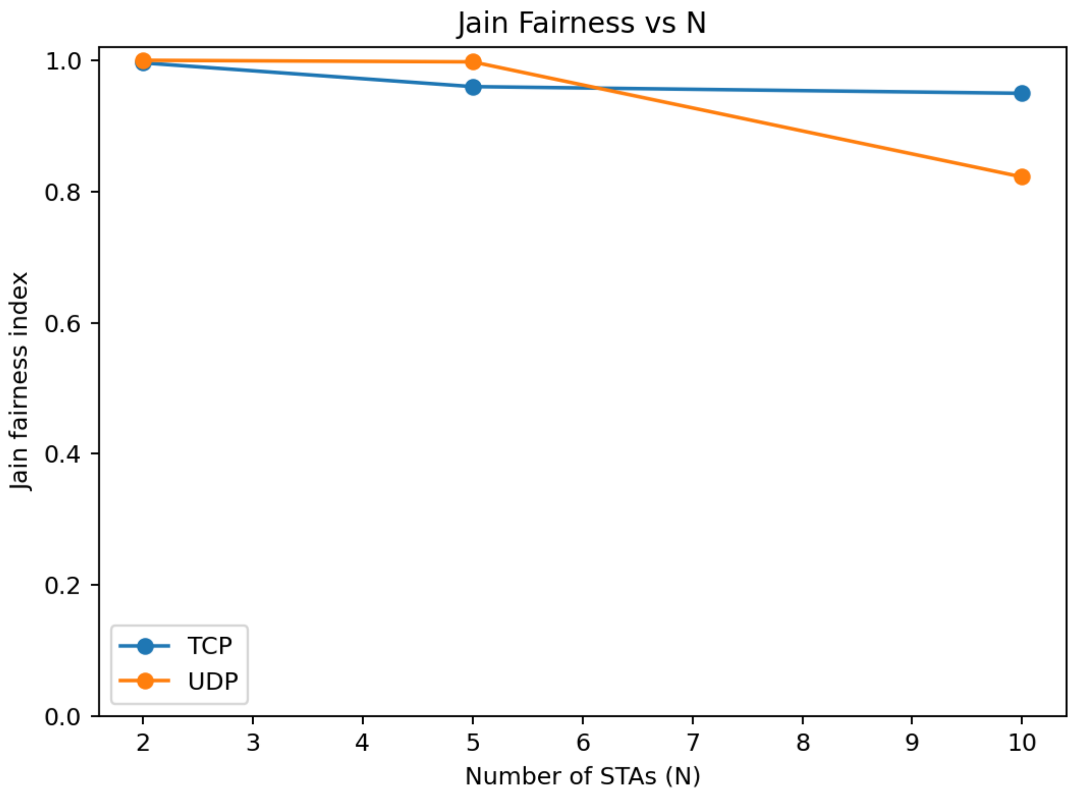


Figure 1: Jain fairness index vs. number of STAs for UDP and TCP.

4. Interpreting Jain Fairness vs. N

Figure 1 shows Jain's fairness index as N increases for UDP and TCP.

UDP: near-perfect fairness at small/medium N, then a sharp drop at $N = 10$. With UDP OnOff traffic, each STA tries to send at a fixed offered rate (e.g. 10 Mbps per STA in your run script). When N is small, the medium has enough capacity and airtime sharing remains close to equal, so $J \approx 1$. At $N = 5$, fairness is still almost perfect ($J \approx 0.998$), indicating that the MAC is distributing airtime in an even way for most stations.

The drop at $N = 10$ ($J \approx 0.822$) indicates that not all STAs are receiving similar goodput anymore. This is typical when contention becomes heavy and the system enters a regime where *small differences amplify*. In Wi-Fi, small differences can come from: rate adaptation (some STAs transmit at a lower PHY rate and consume more airtime per packet), collisions and backoff randomness, capture effects (one signal may be decoded successfully while others collide), and queueing dynamics at the AP/STA. With more contenders, the probability that some nodes experience consistently worse conditions increases, and those nodes can fall behind for long periods, reducing fairness.

TCP: consistently high fairness with mild degradation. TCP's fairness values are high across the board ($J \approx 0.95$–0.996). That is expected because TCP congestion control provides a feedback loop: if a flow receives less service (due to collisions or lower PHY rate), it sees more delay/loss and reduces its sending window. This tends to *pull* flows toward a more balanced operating point. As N grows, some divergence still appears (e.g. due to different RTTs, contention windows, or station rates), so fairness drops slightly, but it does not collapse the way UDP does at $N = 10$.

Important interpretation rule. When J drops, it does *not* necessarily mean the network got "worse" overall; it means *sharing became less equal.* You must read Figure 1 together with sum goodput (next section).

5. Interpreting Sum Goodput vs. N

Figure 2 shows aggregate delivered goodput.

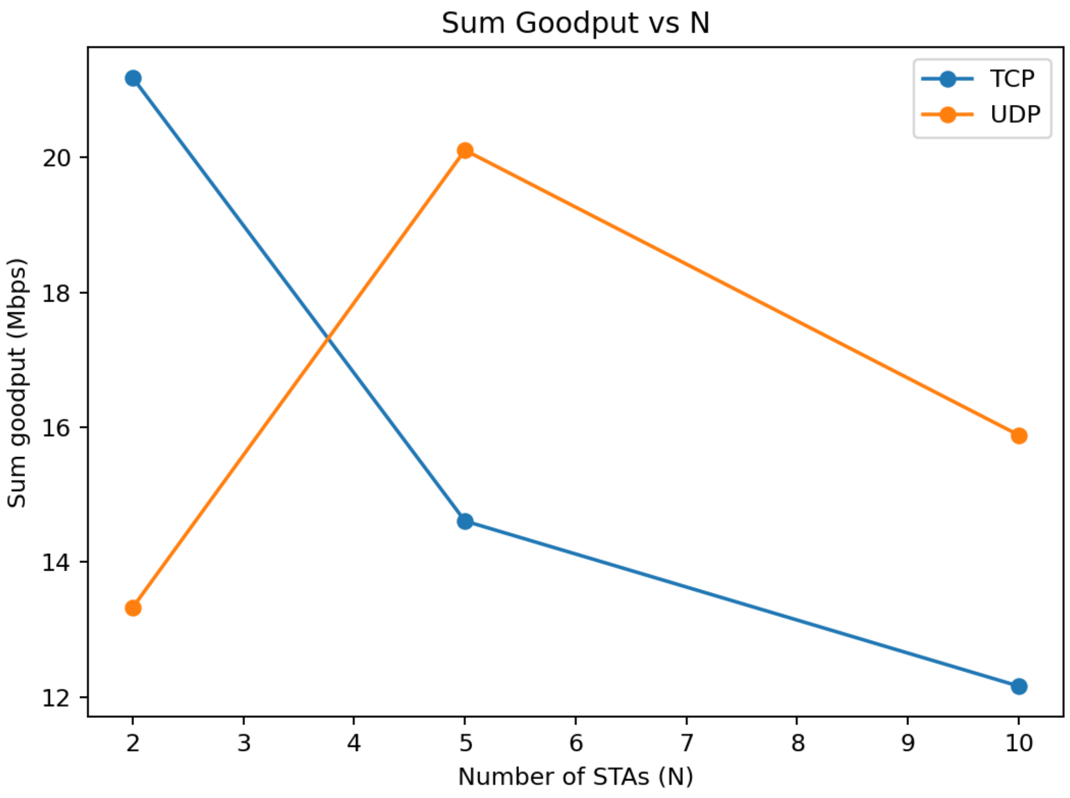


Figure 2: Aggregate (sum) goodput vs. number of STAs for UDP and TCP.

Why does UDP sum goodput jump from $N = 2$ to $N = 5$? At $N = 2$, the aggregate UDP goodput is about 20 Mbps. With five STAs, it rises to about 43 Mbps. This does not mean "adding users always increases capacity". It means that at low N, the offered load may not fully utilize the wireless medium (or some protocol overhead dominates), while at $N = 5$ the aggregate offered load pushes the network closer to its operating capacity, filling idle times and improving utilization.

Another contributing factor is that CSMA (wired) is not the bottleneck (it is 1 Gbps with small delay). The bottleneck is the shared Wi-Fi medium. When more STAs generate traffic, the medium becomes busier, increasing utilization—up to a point.

Why does UDP sum goodput decrease at $N = 10$? At $N = 10$, UDP sum goodput falls to about 39.7 Mbps. With many contenders, collision probability and backoff overhead increase, and a larger fraction of airtime is consumed by contention, retransmissions, and PHY/MAC overhead. Also, if some STAs end up transmitting at lower rates (due to rate control decisions), they consume more airtime for the same payload, reducing total goodput. This is the classic *contention-limited* regime: more senders cause diminishing returns and eventually reduce throughput.

This drop coincides with the fairness collapse in Figure 1. That combination is especially informative: not only does the total delivered data drop, but it also becomes unevenly distributed.

Why is TCP sum goodput relatively flat across N? TCP aggregate goodput is about 31.8 Mbps at $N = 2$ and stays around $30 - -31$ Mbps as N increases. TCP reacts to congestion by reducing sending rates, preventing the network from entering a highly unstable, collision-heavy regime. In other words, TCP traffic "self-regulates" and often stabilizes the operating point of the medium, trading off peak aggregate goodput for stability. That is why TCP can show a lower peak than UDP at $N = 5$ (UDP reaches 43 Mbps) while maintaining higher fairness and avoiding a dramatic drop at $N = 10$.

A useful mental model is:

> UDP behaves like "always push"; TCP behaves like "push, observe congestion, then adapt".

6. Reading the per-STA bar charts (distribution matters)

Per-STA bar charts are where fairness becomes visually obvious: equal bars mean fair sharing; uneven bars mean imbalance. Figures 3 and 4 show the per-STA distributions for UDP and TCP at $N = \{2, 5, 10\}$.

How to connect bar charts to Jain's index. Jain's index is sensitive to the *squared* rates in the denominator. If one STA is much larger than others, G_i^2 increases strongly, pulling J down. That is why the $N = 10$ UDP bars (some high, some low) translate into $J \approx 0.822$ even though several STAs still have decent throughput.

Why do per-STA rates spread out at larger N? In an idealized symmetric setup, all STAs are at the same distance and should have similar channel conditions. Yet Wi-Fi is not purely deterministic: backoff and collision processes are random. With more contenders, the stochastic variations persist longer (some flows have "bad luck" for longer). Additionally, the rate control algorithm may occasionally pick different MCS levels due to transient success/failure histories, and those decisions affect airtime consumption. The combined effect is a wider distribution as N increases.

7. A deeper explanation: MAC airtime, overhead, and transport dynamics

At the Wi-Fi MAC layer, each successful frame transmission requires: contention/backoff time, PHY preamble and headers, MAC headers, inter-frame spaces, ACKs, and possibly retransmissions. When N increases, the channel sees more attempts per unit time, increasing collision probability. Collisions waste airtime and force exponential backoff, so the fraction of airtime used for *useful payload* decreases.

UDP does not reduce its offered load when collisions happen. If the senders collectively offer more than the network can deliver, queues build up, collision probability increases, and performance can degrade. This is why UDP can achieve very high aggregate throughput at moderate N but can lose both fairness and throughput at large N.

TCP behaves differently because its end-to-end control loop reduces the number of packets injected when congestion is detected. That often keeps the network from saturating too aggressively, producing smoother performance across N and more stable fairness. However, TCP also adds overhead (ACKs, congestion window growth dynamics) and may underutilize the channel in certain regimes, so its aggregate goodput may not reach the UDP peak.

8. Practical guidance: how to validate and extend the analysis

A good analysis workflow is: *(i)* verify internal consistency (per-STA sums match summary), *(ii)* interpret fairness and aggregate throughput jointly, *(iii)* look for mechanisms in logs/FlowMonitor that explain anomalies.

If you need to go deeper, the following are effective:

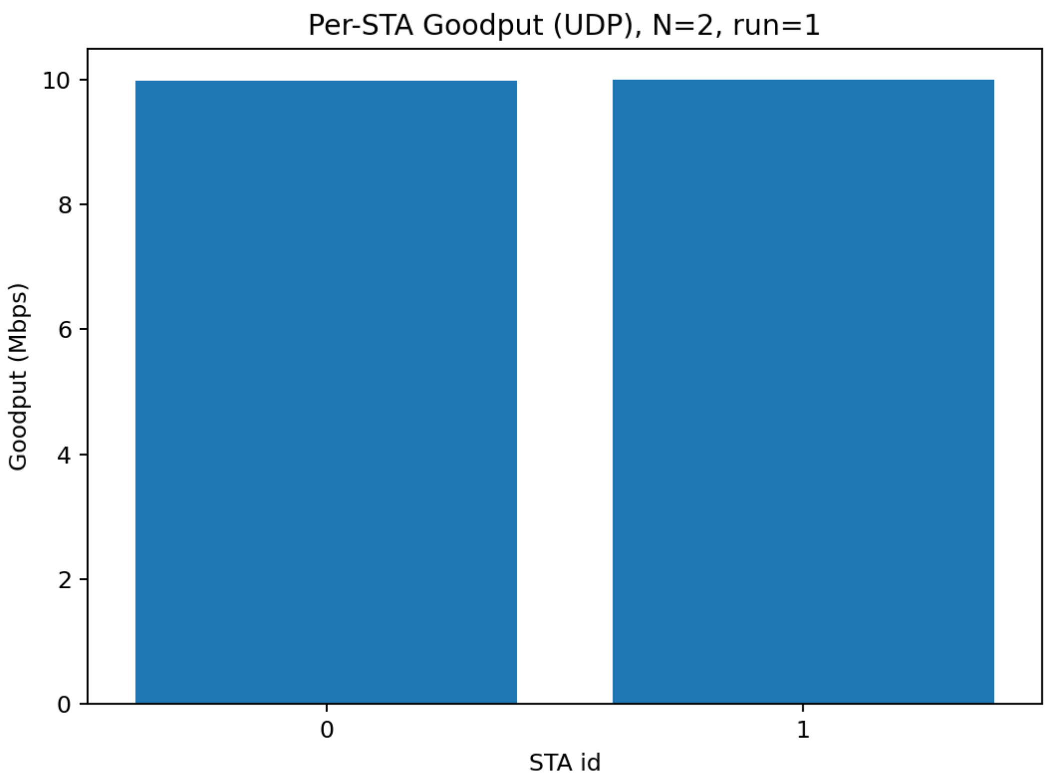
- Use FlowMonitor XML to extract per-flow delay/loss and verify whether unfairness is driven by loss bursts or persistent rate differences.
- Compare different rate managers (e.g. Minstrel vs. constant rate) to isolate the role of rate adaptation.
- Repeat with multiple random seeds/runs and plot mean $\pm$ variability; single runs can be influenced by random contention outcomes.

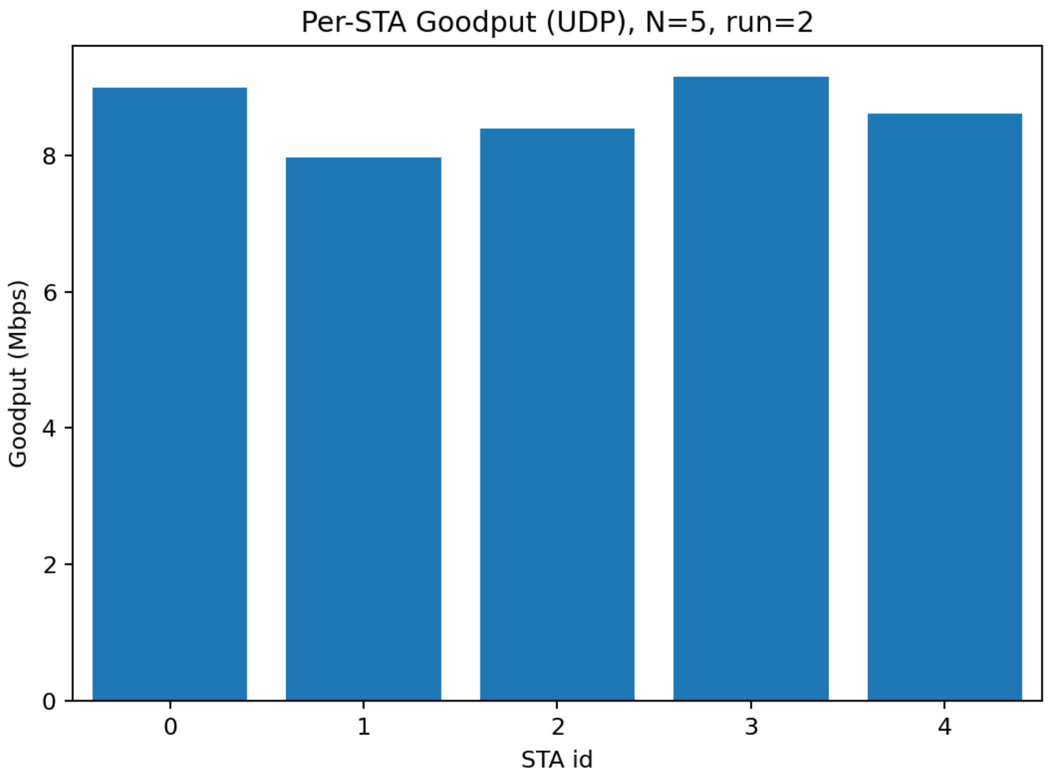
Common pitfall (and what you fixed). Trace callback signatures in ns-3 can vary by version. If you connect a callback with a mismatched signature, ns-3 will terminate with an "Incompatible types" fatal error. The robust approach is either to match the exact traced callback type for your ns-3 version or to connect only the traces that you actually need. Once you fixed the Drop trace signature mismatch, your runs completed successfully and all CSV outputs were produced.

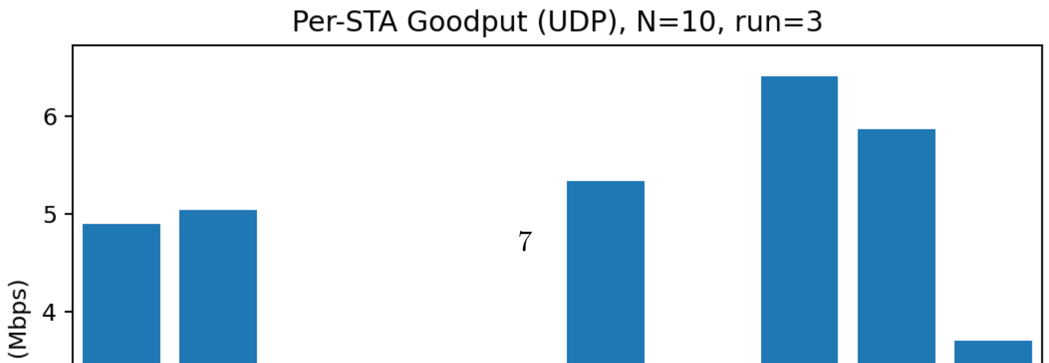
9. Conclusion (what the dataset teaches)

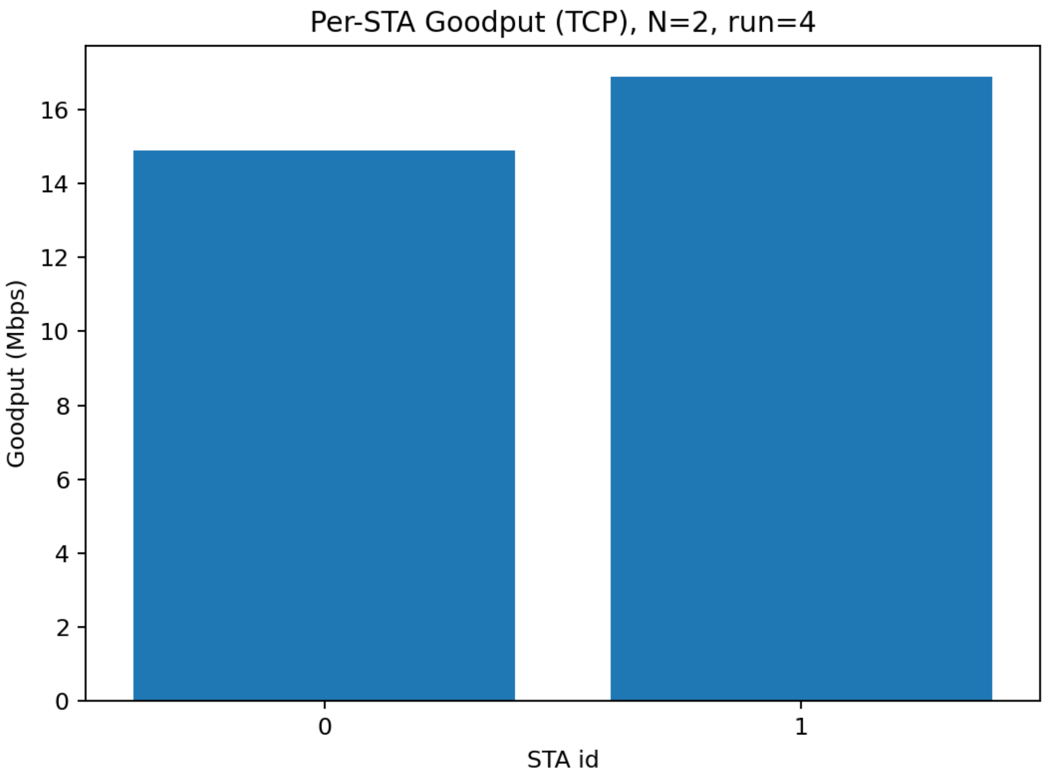
Your results demonstrate a classic lesson in wireless networking: as the number of contenders increases, *performance is not only about capacity, but about contention dynamics and fairness.* UDP can deliver high aggregate goodput at moderate N but can become unfair and less efficient under heavy contention. TCP tends to keep fairness high and aggregate throughput stable by adapting to congestion, at the cost of not reaching the highest UDP peak.

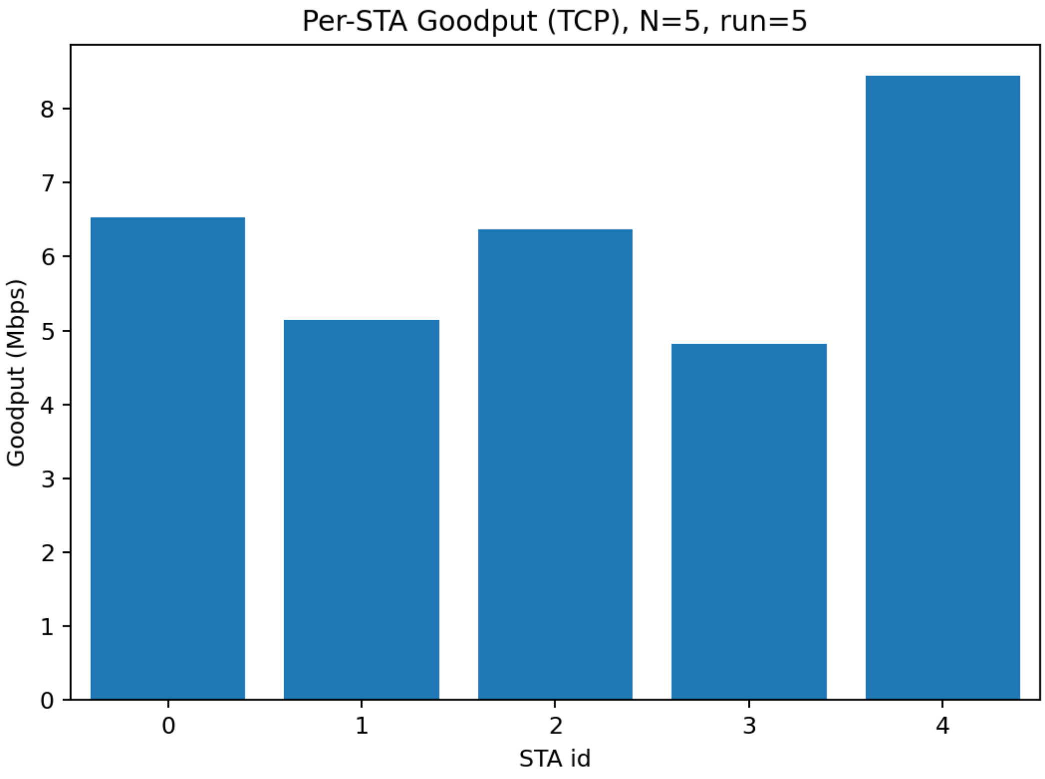
When reporting these experiments, always present fairness and per-STA distributions alongside aggregate throughput; otherwise, you risk missing the most important part of the story: *who benefits and who suffers as load increases.*

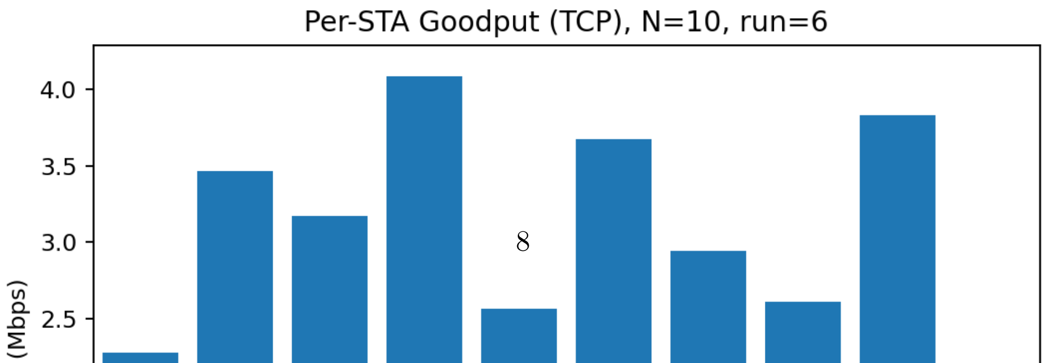

Per-STA Goodput (UDP), N=2, run=1
Goodput (Mbps)
0
2
4
6
8
10
0
1
STA id


Per-STA Goodput (UDP), N=5, run=2
Goodput (Mbps)
0
2
4
6
8
0
1
2
3
4
STA id


Per-STA Goodput (UDP), N=10, run=3
(Mbps)
4
5
6


Per-STA Goodput (TCP), N=2, run=4
Goodput (Mbps)
0
2
4
6
8
10
12
14
16
0
1
STA id


Per-STA Goodput (TCP), N=5, run=5
Goodput (Mbps)
0
1
2
3
4
5
6
7
8
0
1
2
3
4
STA id


Per-STA Goodput (TCP), N=10, run=6
(Mbps)
2.5
3.0
3.5
4.0