USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

The GOODPUT System: A Machine Learning-Driven Optimization Framework for Dynamic Spectrum Control in Heterogeneous WLANs

Yu Wang and Robert Dick, *University of Michigan*

https://www.usenix.org/conference/nsdi26/presentation/wang-yu

This paper is included in the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation.

May 4–6, 2026 • Renton, WA, USA

ISBN 978-1-939133-54-0

Open access to the Proceedings of the 23rd USENIX Symposium on Networked Systems Design and Implementation is sponsored by

جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

The GOODPUT System: A Machine Learning-Driven Optimization Framework for Dynamic Spectrum Control in Heterogeneous WLANs

Yu Wang
University of Michigan

Robert P. Dick
University of Michigan

Abstract

Rapid proliferation of wireless devices is leading to spectrum scarcity, particularly in the unlicensed 2.4 GHz and 5 GHz bands. This paper describes a method to optimize goodput (useful communication throughput) in spectrum-constrained scenarios via access point (AP) channel assignments, channel bonding decisions, and wireless device (station) to AP mappings. Goodput demand prediction and real-time spectrum sensing are used to formulate goodput-maximizing mixed-integer linear programming (MILP) problem instances, which are optimally and efficiently solved to produce network configuration decisions. Our approach is compatible with existing IEEE 802.11 protocols (with primary emphasis on IEEE 802.11ax), commercial access points, modern wireless stations, and legacy infrastructures. A prototype of the system was evaluated and compared with the best existing solutions; it increases goodput by 17.1% relative to multi-AP load-balancing, by 19.3% compared to a machine-learning-based AP selection, and by 47.9% relative to the static RSSI-based methods widely used in existing systems.

1 Introduction

As of 2025, over 20 billion devices are connected to the internet, most wirelessly [25, 43]. The rapid growth of AI-driven applications, such as generative AI services, real-time cloud inference, intelligent assistants, and AI-enhanced video conferencing, has increased both upstream and downstream wireless traffic. Many of these workloads are offloaded to cloud or edge servers, requiring sustained high-bandwidth, low-latency wireless connectivity [9, 16, 21]. Similarly, in industrial and IoT settings, AI-enabled mobile systems (e.g., robots, drones, and smart cameras) transmit sensor data or intermediate features to remote servers, further increasing spectrum demand.

Although Wi-Fi 6E introduced the 6 GHz band, a large fraction of deployed devices operate exclusively on the 2.4 GHz and 5 GHz bands under IEEE 802.11ax/ac/n. Industry projections indicate that billions of Wi-Fi 6 devices will remain limited to these legacy bands, driving them toward saturation in dense deployments and resulting in increased latency and reduced effective throughput [33].

Current Wi-Fi deployments rely on sub-optimal heuristics for spectrum sharing. Most stations select APs primarily based on received signal strength indicator (RSSI), and neglect network load and traffic demands. While adequate in lightly loaded environments, RSSI-based selection leads to imbalance in dense multiple access point (AP) settings, where many stations associate with the same AP, despite available capacity elsewhere [11]. Existing AP- and station-side optimizations remain suboptimal for three reasons: (1) they are reactive, adjusting only after congestion occurs; (2) they rely on instantaneous conditions rather than traffic prediction, leading to unstable associations; and (3) they optimize AP and station decisions separately, ignoring their coupled impact on global goodput. Moreover, many prior solutions require protocol modifications or are incompatible with legacy devices, limiting deployability.

This paper presents **GOODPUT** (Goodput Optimization via Over-the-air Decisions using Predicted User Traffic), a software-defined system that maximizes global goodput in dense, heterogeneous multi-AP Wi-Fi networks, and is compatible with the IEEE 802.11 standard. GOODPUT proactively coordinates channel configurations and station-to-AP associations based on predicted traffic demands and estimated capacity, approaching goodput-optimal spectrum allocation in real time. The system distinguishes between *proactive stations*, which can predict demand and respond to control decisions, and *legacy stations*, which operate under default behaviors and are accommodated without protocol changes. We implement GOODPUT on low-cost commodity hardware (Raspberry Pi 4B) and evaluate it in representative dense-network scenarios.

2 Contributions

This paper makes the following contributions.

1. A problem formulation supporting joint, optimal, real-time solution of the most goodput-relevant network control problems: AP channel selection, bonding configuration, and station-to-AP association assignments. Current station goodput demands and channel capacity estimates are encoded as multiple knapsack problem instances that can be solved in real time by mixed integer programming optimization software to maximize global goodput.

2. An over-the-air software coordination system capable of managing commercial APs via web interface automation, enabling dynamic spectrum allocation and station steering (controlled reassociation of a client from one AP to another) in public settings.

3. A lightweight and automated device classification mechanism that assigns legacy stations to separate channel groups to reduce contention and optimize the overall goodput with little reduction for legacy devices.

The GOODPUT system was validated on a commercial tri-router testbed controlled entirely by our over-the-air coordination software; we conducted a one-hour deployment study in a dense Wi-Fi 6 (IEEE 802.11ax) environment. Across 100 simultaneously active stations driven by realistic human-traffic traces, the GOODPUT system raised aggregate goodput by 47.9% over default RSSI-based association, 17.1% over a state-of-the-art load-balancing scheme, and 19.3% over a deep neural network-based method, while preserving trace fidelity (>95%) and without changes to the Wi-Fi 6 protocol.

3 Related Work

Wi-Fi networks from Wi-Fi 4 (802.11n) through Wi-Fi 6 (802.11ax) are widely deployed and support diverse devices, but face severe congestion in both 2.4 GHz and 5 GHz bands, especially in multi-AP settings such as offices and campuses, leading to degraded performance [54]. Heterogeneous networks with different channel configurations, bonding decisions, and traffic loads exacerbate interference, since clusters often operate with little coordination [22]. This has motivated work ranging from Wi-Fi saturation sensing [18] to advanced throughput and spectrum utilization techniques, and more recently, machine-learning–based approaches [45].

Research on improving multi-AP Wi-Fi performance falls into two categories: traffic load balancing and AP selection. Both are typically evaluated in terms of throughput, which can obscure actual improvements, since collisions are not distinguished from useful transmissions. We instead focus on goodput: useful throughput.

Load balancing is a well-studied technique [10, 17, 30, 37, 44] to equalize workload across APs through reassociation. It often relies on custom firmware monitoring centralized or distributed AP load and computing decisions. After software-defined networking (SDN) introduced more flexible network control and reduced operational costs, load-balancing schemes were ported to controller frameworks [12, 19]. ML-based variants have also been proposed [23]. However, wireless traffic is highly dynamic, and no previous ML model captures all consequential features in real time. Load balancing depends on client steering, which takes several seconds and disrupts connectivity. Hardware [10] and software [49] solutions reduce, but cannot eliminate, this delay. Fundamentally, load balancing is reactive, reorganizing the network only after overload occurs, so users already experience degradation. Moreover, rapidly changing traffic can cause frequent reassociation, reducing goodput. In contrast, the proposed GOODPUT system predicts station demands in advance, enabling proactive bandwidth allocation and preventing congestion.

AP selection aims to select the best AP for each station. The RSSI-based heuristic is suboptimal [5, 6, 52], since it ignores congestion and available capacity. Researchers have proposed alternatives such as signal-to-noise ratio, packet error rate, and channel busy time. More advanced work uses statistical models or ML models trained on large labeled datasets [24, 29, 36, 50]. Yet, AP selection remains limited: it can only choose among existing APs without improving their quality. In dense deployments, this often means selecting the least-bad AP from a congested set, yielding poor performance. In contrast, GOODPUT enables APs to monitor spectrum conditions in real time and reconfigure their channel settings, allowing candidate APs to provide more usable bandwidth.

GOODPUT differs from previous work by treating APs and stations as a supply–demand system: stations generate diverse traffic loads, to which APs adapt their spectrum resource management. The GOODPUT system monitors and predicts station traffic, allocates resources proactively, and jointly coordinates AP reconfiguration, channel bonding [7], and station associations. This goes beyond load balancing and AP selection to maximize system-wide goodput.

4 Problem Definition and Formulation

Problem: Given (i) a Wi-Fi spectrum usage snapshot that associates every transmitter (GOODPUT system controlled APs and potentially interfering APs) with a basic-service-set identifier, channel center frequency, bonded channel width, and observed RSSI-equivalent power level, together with (ii) a demands vector representing all proactive stations in the network, where each station reports its predicted goodput demand for the upcoming k-minute interval, determine a network configuration X that maximizes the sum of down-link goodput over all stations during the next time interval k. X is a tuple containing (i) the center frequency of each AP, (ii) the channel bonding state for each GOODPUT system-controlled AP, and (iii) the AP assigned to each station. We assume (i) APs' locations are static and (ii) proactive stations can execute a lightweight LSTM algorithm for demand prediction (as we experimentally verified). Legacy clients that cannot run prediction models are handled separately and will be discussed

in Section 5.6.

Two components in the GOODPUT system admit performance loss: (i) ML model demand forecasts, and (ii) channel goodput capacity estimates. The GOODPUT system derives overlap-aware goodput capacity from commodity wireless interface readings rather than a laboratory spectrum analyzer, introducing unavoidable measurement error. Our proposed approach is based on a mixed integer linear programming formulation, for which the solver produces globally optimal solutions. Perfect prediction and capacity measurement would yield globally optimal solutions; therefore, the current results constitute a lower bound on the attainable gains.

The remainder of this section presents (i) LSTM-based demand prediction, (ii) channel-capacity estimation, and (iii) the MILP formulation.

4.1 LSTM for Stations Goodput Prediction

The network traffic is predictable [32], and the GOODPUT optimizer requires a real-time estimate of how much goodput each proactive station will need in the next k minute interval, rather than an exact replay of the traffic waveform. We therefore predict every station's scalar goodput demand, the expected average Mbps during the coming k minutes, using a computationally efficient LSTM.

Formulating the input as mean goodput during a time interval rather than a microsecond-by-microsecond time series keeps MILP instances tractable even when hundreds of clients are present. Moreover, the average rate is the most relevant metric: wider channels should be allocated to stations with higher demand, and short-lived spikes should be accommodated by 802.11 MAC contention and TCP congestion control. LSTMs are chosen for their gated recurrence capture the medium-range temporal structures of application traffic, while remaining efficient enough to run on phones or IoT without firmware or hardware changes.

The LSTM predictor is intentionally lightweight to support real-time deployment on heterogeneous devices, from servers to resource-constrained edge systems. Instead of pursuing marginal gains in forecasting accuracy with large deep neural networks, we prioritize responsiveness and operational stability. The model avoids online retraining to control computational overhead and delivers predictions within 2 s, enabling timely integration into the GOODPUT optimization loop. Although heavier models may achieve slightly better accuracy, their increased latency would delay resource allocation decisions and ultimately degrade system responsiveness. This design reflects a fundamental trade-off: in dense multi-AP Wi-Fi environments, predictable, low-latency inference is more valuable than marginal improvements in forecast precision.

4.2 Channel Goodput Capacity Estimation

In dense Wi-Fi deployments, the 5 GHz band can be partitioned into 20 MHz, 40 MHz, 80 MHz, and 160 MHz [7, 28] blocks whose center frequencies may align, so channels may collide with neighbors over a portion of their bandwidth. Channel bonding strategy [14, 26, 27, 35, 38, 39, 41] has been extensively studied: wider channels are more susceptible to interference caused by overlapping channels, reducing the aggregate goodput well below the nominal PHY capacities advertised for the wider channel [1, 13]. The impact scales with the percentage of overlaps and the received power from each interfering transmitter. Effective spectrum allocation, therefore, demands a rapid, quantitative estimate of each candidate channel's goodput capacity that accounts for fractional overlap and interference power. This enables the optimizer to consider current measurements rather than nominal PHY capacity models.

Given the candidate channel $\langle f_c, W_c, P_c \rangle$, and a list of passively observed neighboring AP hosted channels, $\langle f_i, W_i, P_i \rangle$, and noise floor k (in dBm), we can compute a closed-form estimate of the candidate channel's goodput in $O(N)$ time: (1) f_i: center frequency in MHz, (2) w_i: channel width in MHz (e.g., 20, 40, 80, 160), and (3) p_i: estimated received power in dBm.

We now describe the step-by-step process for estimating a candidate channel's goodput capacity.

1. **Fractional spectral overlap.** For every neighbor i,

$$\alpha_i = \max\left(0,\, 1 - \frac{2|f_c - f_i|}{W_c + W_i}\right), \tag{1}$$

   a linear model that yields 0 for non-overlap and 1 for full overlap.

2. **Interference power (mW).** Convert each neighbor's RSSI to linear power and scale by its overlap factor, $I_{\text{total}} = \sum_{i=1}^{N} \alpha_i \, 10^{P_i/10}$.

3. **Noise power.** $N = 10^{k/10}\,\text{mW} \times \frac{W_c}{20\,\text{MHz}}$.

4. **Instantaneous SINR.** $\text{SINR} = \frac{10^{P_c/10}}{I_{\text{total}} + N}$.

5. **Spectral efficiency.** Apply the Shannon–Hartley theorem, $C_{\text{capacity}} = B \log_2(1 + \text{SINR})$.

6. **Capacity and goodput.** The GOODPUT system uses a piecewise-linear calibration model mapping from capacity to goodput that is adapted online using observed AP-side transmitted bytes and station-side received payload bytes (excluding retransmission, headers, etc.).

The Shannon–Hartley theorem provides an upper-bound estimate of channel capacity, and the relationship between this theoretical capacity and application-level goodput is nonlinear due to protocol overhead, retransmissions, and contention effects. Therefore, the GOODPUT system employs

a piecewise-linear calibration model to capture this nonlinearity. For interval k, the goodput estimate is modeled as $C_{\text{goodput}} = \alpha_k C_{\text{capacity}}$, where α_k is an empirical efficiency coefficient associated with that interval. All coefficients are initialized to 1 and updated dynamically using observed statistics from both the AP and stations. Specifically, over a short observation window, we compute the ratio between AP-reported transmitted bytes and successfully received payload bytes at the station, and use this ratio to adjust the corresponding α_k. This piecewise-linear calibration preserves the nonlinear structure implied by the Shannon capacity while allowing online model adaptation to measured link behavior.

Our estimator is based on several simplifications. First, the linear overlap factor in Equation 1, models frequency-domain intersection but ignores adjacent-channel rejection (ACR), orthogonal frequency-division multiple access (OFDMA) subcarrier leakage, and time-varying duty cycles, so interference may be overstated or understated. Second, the single, static noise floor disregards variations in receiver noise, temperature, and spurious emissions. Third, this estimation neglects fast-fading, MIMO beamforming, and spatial diversity; hence, the measured SINR can deviate by ±5–10 dB from the instantaneous value seen by each spatial stream. Fourth, mapping SINR to spectral efficiency via the Shannon–Hartley theorem omits implementation losses and contention effects such as back-off and collisions. The effects of these imperfections are captured in our experimental evaluations, i.e., improvements are reported in the presence of suboptimal estimation.

4.3 Formulation and Objective Function

One of the core technologies in the IEEE 802.11ax protocol, OFDMA, allows it to slice the channel into resource units (RUs) that serve multiple stations concurrently [4, 34, 53]. Conceptually, each AP behaves like a knapsack whose capacity equals the total goodput based upon channel configuration, while each station's predicted demand is an item whose weight and value are the same. The scheduler's task is picking a configuration (setting the knapsack size) and packing a subset of station demands without exceeding the channel capacity, mirroring the classical multiple-knapsack problem.

We study a dense IEEE 802.11ax WLAN containing a set of access points $\mathcal{A} = \{a_1, \ldots, a_m\}$ and a set of stations $\mathcal{S} = \{s_1, \ldots, s_n\}$. Time is discretized/slotted: let $\mathcal{T} = \{1, \ldots, T\}$ be the planning slots and Δt be the slot duration (s). Each AP can be set, at the beginning of every planning slot, to one of K discrete center-frequency with channel-width options $\mathcal{C} = \{c_1, \ldots, c_K\}$; a configuration c offers a goodput capacity C_c (Mbps). A LSTM predictor (explained in Section 4.1) supplies the goodput demand $d_i(t) \in \mathbb{R}_{\geq 0}$ (Mbps) for every station $s_i \in \mathcal{S}$ and slot $t \in \mathcal{T}$. Reconfiguring an AP from one configuration to another causes a 30 s outage (on ASUS RT-AX88U PRO), and steering a station between APs causes a 5 s outage. These delays are properties of current commodity AP and client hardware rather than limitations of the GOODPUT design; commercial platforms trigger a device reboot when changing channel or bonding settings, causing seconds of disruption, whereas managed network infrastructure can reduce steering latency to about 200 ms [10] and emerging Wi-Fi standards (IEEE 802.11bn) are designed to support near-seamless roaming [31]. Given the demand trace, determine (i) the configuration of each AP in each slot and (ii) the station–AP association schedule to **maximize the total useful data** (goodput × time). We define these decision variables,

$x_{i,j,t} \in \{0,1\}$ 1 iff station i is associated to AP j in slot t,
$y_{j,c,t} \in \{0,1\}$ 1 iff AP j uses configuration c in slot t, and
$z_{i,t} \in \{0,1\}$ 1 iff station i is *up* (not in outage) in slot t.

The objective function is

$$\max \sum_{t=1}^{T} \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j,t}\, z_{i,t}\, d_i(t)\, \Delta t \tag{2}$$

under the following constraints:

$$\sum_{c=1}^{K} y_{j,c,t} = 1, \qquad \forall j,t, \tag{3}$$

$$\sum_{j=1}^{m} x_{i,j,t} = 1, \qquad \forall i,t, \tag{4}$$

$$\sum_{i=1}^{n} x_{i,j,t}\, z_{i,t}\, d_i(t) \leq \sum_{c=1}^{K} y_{j,c,t}\, C_c, \qquad \forall j,t. \tag{5}$$

A mixed-linear integer program [48] is a natural formulation for our WLAN goodput optimization problem because all key design decisions, such as selecting a channel configuration for each AP in each time slot, determining the AP association for each station, and indicating whether a station is temporarily offline, including capacity limits and outage rules, can be expressed as linear equalities or inequalities. This combination of discrete decision variables and linear constraints is precisely where MILP solvers excel: they guarantee global optimality (or provide a bounded optimality gap), and easily accommodate new engineering requirements through additional linear constraints.

Google OR-Tools' MPSolver engine [20] is well-suited to our optimization problem because it supports pure integer linear models, applies automatic presolve, cutting-plane generation, and lazy-clause evaluation in addition to traditional branch-and-bound, and exploits all available CPU cores in parallel. In our implementation, the OR-Tools-based solver can handle the largest benchmark case (3 APs and 100 stations) in approximately 3 minutes, producing the globally optimal solution. The GOODPUT system design and optimization formulation naturally extend to larger deployments with more APs and stations. However, as the number of devices increases, the MILP problem size grows, and the time required to compute the optimal configuration increases as well. In practice,

this can be addressed by adjusting the optimization interval or by partitioning large deployments into smaller coordination domains, thereby maintaining tractable optimization while preserving the benefits of coordinated spectrum management.

5 GOODPUT System Design

This section describes the architecture and implementation of the GOODPUT system. As shown in Figure 1, the system consists of three integrated modules: network sensing, optimization algorithm, and infrastructure control. The network sensing module gathers inputs, including spectrum usage data and future goodput predictions from proactive stations. These inputs feed into the optimization algorithm, where network analysis and an MILP solver jointly determine reconfiguration strategies. Finally, the infrastructure control module enacts these decisions through channel re-assignment, router re-configuration, and station re-association, resulting in optimal network deployment. The following sections provide detailed descriptions of each component, including sensing, prediction, reconfiguration, deployment options, station control, and legacy device accommodation.

5.1 Network State Sensing

Existing routers and APs face clear limitations in channel selection and frequency assignment, especially in dense or dynamic environments. We use the LinSSID tool [42] to scan spectrum occupancy, illustrating a typical multi-AP environment such as an office, in Figure 2. Consumer-grade devices typically perform a one-time scan at boot or reset, choosing the least congested channel based on simple measures such as RSSI or the number of nearby APs. Such static choices ignore dynamic traffic demand, hidden node interference, and time-varying channel quality, leaving channels fixed regardless of changing conditions. Even enterprise and mesh systems that support periodic re-evaluation update infrequently, since channel switching disrupts clients and forces re-association. This cost is unavoidable, as effective coordination requires steering stations to new APs. In contrast, the GOODPUT system explicitly accounts for this overhead and reconfigures only when it produces net benefit.

To characterize the external spectrum occupancy conditions in the 5 GHz band, the GOODPUT system first scans the spectrum using the LinSSID tool. LinSSID interfaces with the underlying Wi-Fi chipset for an active scan over all available 5 GHz channels. This scan collects beacon frames and probe responses transmitted by nearby access points (APs), enabling the GOODPUT system to infer the external channel usage profile of co-located wireless networks not under its control. The resulting data are parsed to extract the following key attributes for each observed Basic Service Set (BSS): $\mathit{channel\_usage} = [(f_1, w_1, p_1),\ (f_2, w_2, p_2),\ \ldots]$, where each tuple (f_i, w_i, p_i) represents a single observed external channel.

This dataset captures the spectrum-occupancy profile of the surrounding wireless environment, including potential channel overlaps, interference regions, and signal-strength gradients. The sensed spectrum data are then fed into the GOODPUT system's optimizer, which evaluates the goodput capacity of candidate channel configurations for each AP. In particular, the algorithm incorporates interference models that penalize overlap with external channels (e.g., shared 20 MHz subchannels), the power level of the interfering signal, and the channel width of both the candidate channel and the external AP's transmission. This interference model enables the GOODPUT system to intelligently avoid heavily occupied channels and mitigate adjacent-channel interference.

The GOODPUT system prototype uses single-point spectrum sensing as a lightweight, practical approximation to capture dominant channel-occupancy trends while keeping sensing overhead and deployment costs low. This is a design simplification rather than a limitation of the GOODPUT optimization framework. The optimization formulation is not restricted to single-point measurements and can directly incorporate richer inputs, such as per-AP spectrum scans or distributed sensing reports.

5.2 Station Goodput Demand Prediction

The GOODPUT controller proactively optimizes spectrum allocation using predicted demand rather than reacting to lagging indicators such as airtime utilization or queue length. If stations provide short-interval forecasts, the system avoids channels projected to be capacity-limited and redistributes stations with impending bursts. We predict the average goodput over k-minute windows, since average demand determines the allocation of OFDMA resource units. Although sub-second bursts occur, prior work [15, 46, 47] shows that IEEE 802.11 MAC fairness ameliorates these effects.

Each station runs a lightweight daemon that captures packets via `tcpdump`, aggregates payload bytes per second (Tx/Rx), and produces a 1 Hz goodput estimation stream. Over 20 days, we collected 150 hours of real usage traces from nine participants (Section 6.2). Packet captures were filtered with `tshark` to remove retransmissions and control frames, then split into training, validation, and hold-out sets. Training completes in 30 minutes on a GPU (GTX-1660 Ti); the final model is deployed using TensorFlow-Lite Micro.

On the hold-out set, the model predicts future goodput demand within ± 2 Mbps of the ground truth 32% of the time, and within ± 4 Mbps 81% of the time. Given that the average per-station demand in our traces is around 10 Mbps, these correspond to relative errors of approximately $\pm 20\%$ and $\pm 40\%$, respectively. This level of accuracy is sufficient for the GOODPUT system's channel bins (20/40/80/160 MHz). Inference is extremely lightweight on the ARM Cortex-A72 at 1.5 GHz, forecasting the next 3 minutes in less than 2 seconds, with no more than 6% processor utilization. Consequently,

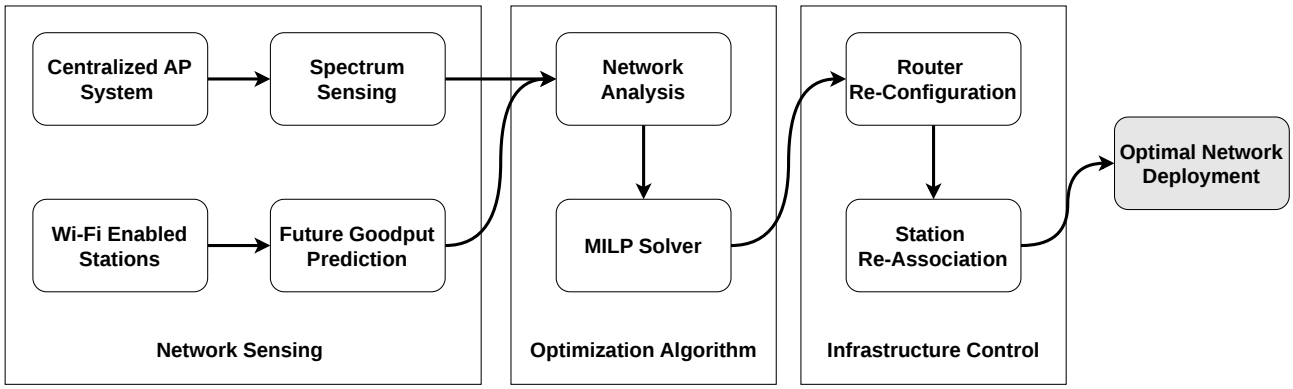

Figure 1: GOODPUT system components and data flow.

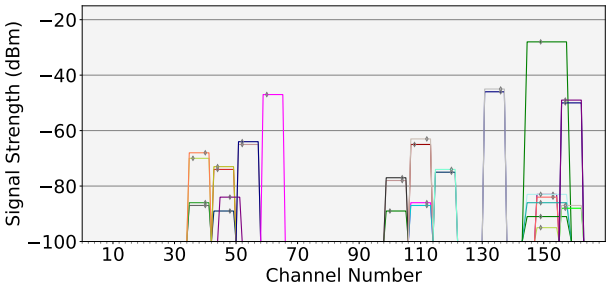

Figure 2: LinSSID scan results in the 5 GHz band. Colored blocks show detected Wi-Fi channels with bandwidth and signal strength. Channel overlaps indicate interference, with interferer signal strength positively related to severity.

the predictor can run continuously on phones, PCs, or IoT devices with little time or energy overhead.

To evaluate the resilience of the station demand predictor under real deployment conditions, we conducted controlled experiments comparing three configurations: (i) an oracle predictor, (ii) the deployed LSTM-based predictor, and (iii) a sliding-window historical average predictor. The oracle predictor is a hypothetical upper bound that uses the ground-truth future demand trace, which no real prediction algorithm can outperform. It serves only to quantify the maximum achievable performance under perfect prediction.

For each configuration, we deployed the system under identical workload and network conditions. Results show that the oracle configuration achieves 12.7% higher aggregate goodput than the LSTM-based solution, establishing an upper-bound performance gap attributable to prediction error. In contrast, the sliding-window historical average method achieves 9.7% lower aggregate goodput than the LSTM-based predictor. These results indicate that the LSTM-based predictor provides more accurate demand estimates than simple temporal averaging, enabling performance substantially closer to the oracle configuration.

5.3 Network Reconfiguration

Once the GOODPUT system has completed both the background spectrum measurement and predicted the future goodput of each station, the resulting information is passed to the solver described in Section 4.3. Given correct input data, the solver produces an optimal network configuration consisting of three elements: (i) the channel setting for each controlled AP, (ii) the channel bonding decision for each AP, and (iii) the station-to-AP association mapping. To realize these decisions in practice, the GOODPUT system executes a two-stage reconfiguration process: (i) **AP Reconfiguration:** the GOODPUT system controller remotely accesses the selected AP via its administrative interface (described in Section 5.4) and applies the new channel and bonding configuration. During this step, the AP is temporarily unavailable to stations. (ii) **Station Reassociation:** prior to AP reconfiguration, the controller informs each station of its new AP association via BSSID, since the temporary outage prevents wireless communication over TCP. Once stations lose their previous Wi-Fi association, they initiate a 35-second pause before attempting to establish a new connection with the assigned AP.

This two-stage reconfiguration procedure ensures that the solver's output can be faithfully applied on commodity hardware while accounting for the temporary disruptions caused by AP resets and client handoffs. Although the process introduces short outages during reassociation, these are explicitly captured in our formulation, allowing the GOODPUT system to consider reconfiguration overhead.

5.4 Software and Hardware Deployment Options

The GOODPUT system can be deployed as a software-only solution without modifying commercial APs. In this mode,

the controller interacts with standard APs through their web-based administrative interfaces, while proactive stations run a lightweight background script (Section 5.5) to provide demand forecasts and receive steering commands.

The controller can run on desktops, laptops, Raspberry Pi boards, or smartphones. With administrator credentials, it authenticates to AP management consoles and centrally coordinates configuration changes across the network. Using Selenium [40], a headless browser automation framework, the controller (i) adjusts channel and bonding settings, (ii) verifies applied configurations, and (iii) retrieves connected-station lists for steering and demand queries. Our prototype uses ASUS RT-AX88U Pro routers.

The controller operates as five concurrent threads: (1) spectrum sensing via the host Wi-Fi interface (Section 5.1); (2) AP configuration control; (3) demand collection via TCP; (4) station steering; and (5) MILP-based optimization (Section 4.3). This modular architecture enables centralized, over-the-air management of all APs within radio range, providing a non-intrusive and scalable deployment model.

Although evaluated on commodity hardware, the GOODPUT system could be integrated into custom AP hardware. Unlike commercial APs with one radio per band (2.4 GHz/5 GHz/6 GHz), a custom design could host multiple radios per band and run the solver locally. Powered by IEEE 802.3bt Type 4 PoE (up to 90 W), such hardware could reduce reassociation latency from 5 s to 200 ms [10], making it suitable for multi-channel PoE-powered deployments.

5.5 Station Control

The station control subsystem serves two roles: (i) providing per-station goodput predictions, and (ii) executing steering commands from the GOODPUT controller. Proactive station support is optional, not required. Stations that run the goodput predictor and steering background script are proactive, all others are legacy stations. Although IEEE 802.11k/v/r define advisory roaming, support is inconsistent across commercial APs and clients, and stations may reject requests. This makes roaming unpredictable and incompatible with GOODPUT's need for deterministic steering, so we deploy a lightweight software agent on each proactive station to do that.

Communication uses TCP sockets in a request–response model initiated by the controller. At the start of each optimization cycle, the controller queries stations for predicted goodput demand. Each station replies with its embedded LSTM predictor's output of the average goodput during the next time interval. This ensures the optimization formulation (Section 4.3) always uses current, station-specific forecasts.

The GOODPUT system commands stations to steer to a target AP identified by its BSSID. Steering is triggered using platform-specific mechanisms: `netsh` on Windows, `wpa_supplicant` on Linux, and the `WifiConfiguration` class in `android.net.wifi` package for Android devices.

To ensure reliability, the agent includes a recovery mechanism for failed steering. About 5% of commands fail due to transient contention or insufficient inter-command spacing. In such cases, the agent resets the association state and retries steering. With this safeguard, we observed no failure during deployment experiments.

5.6 Legacy Device Interference Mitigation

Most prior optimization systems assume cooperative stations (exposing PHY/MAC statistics or accepting driver changes), leaving IEEE 802.11a/b/g/n legacy clients unaddressed. Systems such as SAP [10] rely on responsive stations, but legacy IoT devices, printers, or set-top boxes cannot be upgraded and often remain associated with their initial AP for extended periods, consuming bandwidth and degrading load balancing. Ignoring them risks starving proactive stations. The GOODPUT system explicitly accounts for legacy devices and migrates them to one AP to maximize predictability.

A lightweight over-the-air "Wi-Fi blinking" protocol requiring no client modification. Proactive stations are temporarily disassociated (120 s) while all but one AP are disabled, forcing legacy devices to reconnect to the remaining AP. Over 90% of tested legacy chipsets reassociate within 120 s and rarely roam thereafter unless RSSI drops below -75 dBm. Although blinking causes a brief network-wide interruption, it is performed infrequently (once per week) during low-usage periods, quarantining legacy clients for extended durations.

Legacy traffic is monitored for channel utilization and avoid starving proactive stations. Since legacy device throughput cannot be directly measured,measured directly, we compute it by subtracting the proactive station throughput from the This correction is used only for legacy devices: the actual goodput of proactive devices is visible to the GOODPUT system.

When reconfiguring the AP hosting legacy devices, brief disconnections occur during reboot. In deployment, legacy clients consistently reconnected to their original AP before considering alternatives. Legacy devices experienced only a 15.8% throughput reduction, while GOODPUT achieved substantially larger system-wide (Section 6.6).

6 Evaluation

We evaluate GOODPUT on a real Wi-Fi 6 testbed with three ASUS RT-AX88U Pro APs and 100 emulated stations driven by 150 hours of human-in-the-loop wireless communication activity traffic traces. Unlike simulator-based studies, our deployment with commodity routers and clients captures real channel contention and interference, so results reflect real-world dense-network conditions. We benchmark against three baselines: RSSI association, load balancing, and AP selection. System-wide and per-station performance are reported.

6.1 Hardware and Deployment Setup

Our deployment used three ASUS RT-AX88U Pro access points running firmware version 3.0.0.6.102, each with a 4×4 dual-band radio interface controlled by the GOODPUT system. Traffic was replayed by fifteen hosts: twelve Raspberry Pi 4B boards with BrosTrend AX1800 USB adapters and three 2021 MacBook Pros. Each host used client virtualization [51], allowing a single physical Wi-Fi interface to instantiate multiple virtual clients with independent traffic traces.

Each AP was paired with one MacBook Pro and four Raspberry Pis. The MacBook Pro emulated high-goodput demand devices, and two of the Raspberry Pis emulated low-goodput demand devices. These three devices formed a *trace-replay group*. The remaining two Raspberry Pis served as the *steering machines*, executing Wi-Fi handoffs by intercepting station IDs and trace points, performing the handoff, and notifying replay devices to resume at the correct point. Afterward, they reverted to the original AP. This setup allowed realistic station steering with limited hardware while preserving both the handoff delay and the possibility of failures, with recovery protocols defined in Section 5.5. In total, we realized 100 logical stations. The GOODPUT system controller is deployed on Raspberry Pi 4B. All testbed components were commercial Wi-Fi hardware, showing that GOODPUT requires no hardware changes.

Experiments were conducted in a 1,500 ft^2 office with 35 occupants and 97 wireless devices generating background traffic. The environment included 7 university-managed APs on fixed 80 MHz- wide channels in the 5 GHz band, yielding RSSI values between -45 dBm and -70 dBm. To evaluate GOODPUT, three controlled APs were deployed at table height (30 inches) with overlapping coverage, yielding RSSI values between -35 dBm and -60 dBm. This setup enabled interference, multiple viable AP associations, and controlled reconfiguration and steering in the dense multi-AP network. Our evaluation targets dense indoor WLAN deployments rather than wide-area or campus-scale networks, and is designed to stress contention, interference, and heterogeneous demand under controlled and repeatable conditions.

6.2 Data Collection

Synthetic network traffic traces generated by ns-3 or similar simulators are statistically convenient, but systematically underrepresent the temporal burstiness, protocol overhead, and multi-application concurrency that dominate today's Wi-Fi environments. Prior measurement papers also indicated that synthetic traces introduce error and compromise the result by 30% to 40% [2, 8], and human-driven flows exhibit interarrival times and on/off clustering that Poisson or Pareto generators miss [3]. Goodput-centric optimization, therefore, requires traces that capture application mix, device mobility, and MAC-layer retransmissions. Consequently, we collected

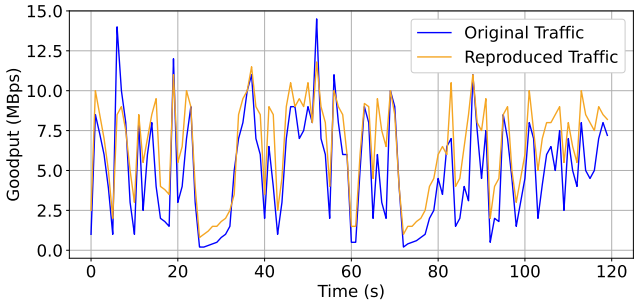

Figure 3: Original vs. reproduced downlink traffic traces. The reproduced trace closely follows the original, preserving burstiness and variability in goodput.

a device-level trace repository that preserves the diversity of real human behavior, and we use it both to train our LSTM demand predictor (Section 5.2) and to drive the 100-station replay in the deployment experiment.

Our dataset focuses on **downlink payload traffic**, which dominates spectrum contention in dense Wi-Fi deployments. Uplink, while relevant for some latency-sensitive tasks, is not part of GOODPUT's current objectives. We recruited nine volunteers (four engineers, three graduate students, and two administrative staff members) and instrumented their daily-use laptops and phones using Wireshark (Windows, Linux, macOS) and PCAPdroid (Android). Data collection spanned homes, cafés, offices, and transit across weekdays and weekends. Volunteers used devices freely, producing realistic browsing, streaming, and gaming. In total, we captured over 150 hours of time-stamped traffic at one-second granularity.

The database was split into 150 one-hour segments. Fifty were used for LSTM training and validation, while the remaining 100 were reserved for deployment experiments. Each station (100 stations in total) was assigned a distinct one-hour segment for trace replay to create a dense network condition.

6.3 Trace Replay Methodology

To convert second-granularity downlink traces into live, over-the-air traffic, we adopt a station-driven pull model that faithfully preserves the byte-level traffic without the storage overhead of full payloads. Each logical station runs a lightweight Python client that, at the start of every wall-clock second t, reads the pre-recorded goodput sample G_t (bytes). It then issues an HTTP request to our remote Linux replay server that hosts a content generator. The server's multi-threaded handler synthesizes a G_t bytes payload on-the-fly and streams it back over TCP; data are not stored on disk.

We measure station received payload $\hat{G}_t$ size. If $\hat{G}_t \geq G_t$, the trace requirement for that second is satisfied, and any headroom is left idle to avoid biasing subsequent slots. If the station fails to receive the full amount ($\hat{G}_t < G_t$), the shortfall is not carried into the next second; the client immediately

issues the size request for G_{t+1} when the next 1-second timer fires. This strict one-second bucket discipline preserves temporal burstiness and prevents back-to-back HTTP pipelining from smoothing the workload.

Post-experiment validation, plotted in Figure 3, shows a correlation coefficient of $\rho = 0.92$ between the original traces and the over-the-air captures, and 95% of 1 s buckets deliver at least the requested bytes (slightly conservative over-delivery ensures the GOODPUT system is evaluated against a harder workload). Thus, the replay pipeline maintains both the timing fidelity and volume pressure necessary for testing.

6.4 Comparison Baselines and Measures

To evaluate the GOODPUT system, we benchmark its performance against three representative spectrum management strategies: (i) the default RSSI-based association used by commercial stations and APs, (ii) a dynamic multi-AP load balancing scheme [10], and (iii) a deep neural-network-aided AP selection technique [24]. These baselines were selected to capture the lowest-effort legacy behavior and the most advanced research. All techniques are evaluated under identical conditions by replaying the same traces on the same testbed.

In the RSSI-Based Association (factory default) baseline, all stations search and associate with the strongest signal, subject to a default roaming threshold based on the station's native system settings. This emulates the factory firmware behavior of most consumer and enterprise Wi-Fi devices. Traffic load, capacity, and station demand are ignored. Steering occurs when signal strength drops below a threshold or upon explicit disconnection. This scheme is known to generate potential load imbalance and poor aggregate goodput in high-density deployments, but remains the most common approach in unmanaged WLANs.

For the **dynamic multi-AP load balancing** baseline [10], we used the concepts and formulation from the SAP system, with code-level modifications to ensure compatibility with our three-AP testbed. Our framework supplies station statistics, including transmission rates, traffic rates, and data payload lengths, for the SAP algorithm to compute association decisions. It is important to note that we did not adopt the unique One-AP-Multiple-Interface (OAMI) architecture, as commodity AP hardware provides only one radio interface. Since our system is designed to work with off-the-shelf products, we do not use SAP's ultra-fast handoff mechanism. However, the load-balancing technique from SAP is evaluated.

The **deep neural network-aided AP selection** baseline uses Adaptive Target-Condition Neural Network (A-TCNN) [24]. The model integrates two components: (i) the target-condition neural network (TCNN), which performs AP selection for a single target station conditioned on the presence of other stations, and (ii) the adaptive mechanism. This design enables A-TCNN to generalize across varying station numbers without retraining, while maintaining decision consistency. For training, we followed the mixture training process described in the original paper, combining samples from multiple user-number cases into a single dataset. To align the model with our Wi–Fi–only environment, we disabled the LiFi link capacity term $C_{i,j}$ and retained only the Wi-Fi channel capacities. All other parameters and functions were kept consistent with the original setup.

The GOODPUT system is designed to optimize network-level goodput rather than application-specific quality-of-service metrics. While interactive applications such as VoIP, cloud gaming, and video conferencing are sensitive to latency and transient disconnections, evaluating application-level QoS requires application-specific instrumentation and workload models, which are beyond the scope of this work. Therefore, our evaluation focuses on (i) the system-wide aggregated goodput across all stations (proactive and legacy stations combined), (ii) the system-wide aggregated goodput for proactive stations, (iii) the average goodput fulfillment rating (AGFR) at one-second granularity, (iv) the average steering cost per station, (v) the sum of channel widths across all APs, and (vi) the system-wide aggregated goodput for legacy stations. Results and implications are shown in Section 6.6.

6.5 Experimental Procedure

In each trial, the GOODPUT system runs the Wi-Fi blinking protocol to isolate legacy devices. After isolation, the coordinator broadcasts a start signal over the control network, prompting all stations to begin traffic replay as described in Section 6.3. Each station issues download requests to the remote traffic server based on its assigned goodput trace.

The optimization solver runs iteratively. In the first iteration, it waits for the demand predictor to accumulate sufficient history; subsequent iterations start immediately upon completion of the previous iteration. Trials showed that a two-minute warm-up period is required for the LSTM predictor to produce stable estimates (see Section 5.2), so the coordinator enforces this window to allow traffic to stabilize.

After warm-up, the coordinator polls stations for predicted demand while simultaneously scanning the 5 GHz band with its built-in Wi-Fi card to capture co-channel and adjacent-channel activity. These two inputs—demand predictions and spectrum sensing—are fed into the optimization solver. Whenever the solver's output differs from the current configuration, the system triggers a reconfiguration cycle. This closed-loop process continues throughout the one-hour runtime.

For all baselines, we used the same testbed and deployed traffic and spectrum management policies as described in previous work.

6.6 Experimental Results

The performance of the GOODPUT system and three baselines is evaluated using six key metrics (Section 6.4). In this

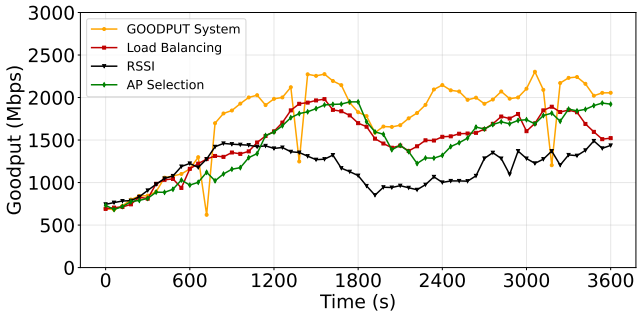

Figure 4: System-wide goodput (including proactive and legacy stations) for the GOODPUT system, load balancing, AP selection, and RSSI-based association.

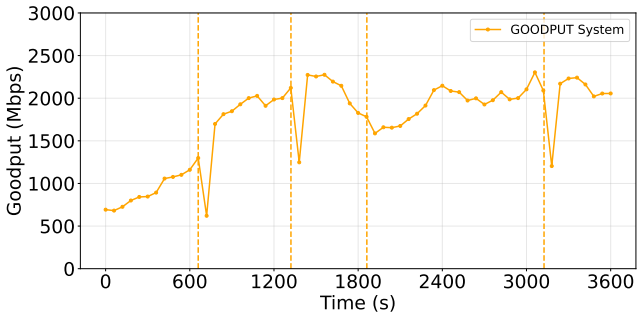

Figure 5: System-wide goodput (including proactive and legacy stations) for the GOODPUT system. Four network reconfiguration events (dashed vertical lines) occur at 660 s, 1,321 s, 1,862 s, and 3,125 s. Only the event at 1,862 s involved station reassociation without AP reconfiguration.

section, we present the results for each metric and discuss the significance of the observed improvements.

System-Wide Aggregated Goodput For All (Proactive and Legacy) Stations

The system-wide aggregated goodput is defined as $\sum_{t=1}^{T}\sum_{i=1}^{N} G_{i,t}$, where N denotes the total number of stations (including both proactive and legacy stations); t indexes the time slots in seconds, corresponding to a one-hour experimental runtime; and $G_{i,t}$ represents the actual goodput achieved by station i during second t, measured as successfully delivered payload data (e.g., in Mbps).

$G_{i,t}$ captures the total volume of successfully transmitted data across all stations and over the entire experiment. Figure 4 shows that the GOODPUT system increases system-wide aggregated goodput by 17.1% on average relative to load balancing, by 19.3% compared to AP selection, and by 47.9% relative to the static RSSI-based method, shown in Figure 6.

Figure 5 shows the goodput trace under the GOODPUT system, highlighting four network-wide reconfiguration events indicated by vertical dashed lines. The first, second, and fourth

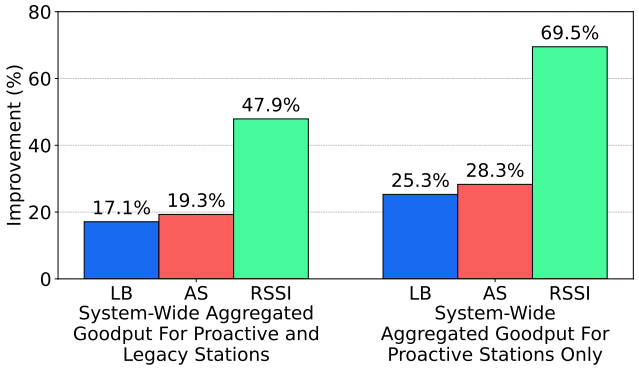

Figure 6: System-wide goodput improvements. LB = Load balancing, AS = AP selection

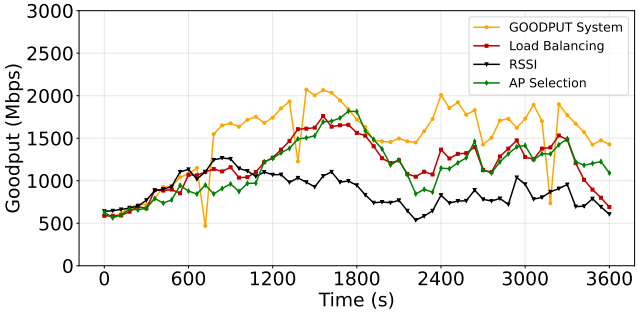

Figure 7: System-wide proactive stations goodput for the GOODPUT system, load balancing, AP selection, and RSSI-based association.

events involve AP channel reassignment combined with distributed station steering, while the third event involves only station steering. Each reconfiguration window (lasting approximately two minutes) causes a temporary reduction in goodput due to transient link outages and reassociation delays. Nevertheless, the system quickly recovers as the new configuration takes effect, consistently achieving higher throughput than in the pre-reconfiguration state.

System-Wide Aggregated Goodput For Proactive Stations

This section considers the impact of the GOODPUT system on proactive stations that we expect will dominate network membership in the future. The GOODPUT system significantly improves goodput for this group compared to all three baselines. From Figure 7, GOODPUT achieves a 25.3% increase relative to load balancing, a 28.3% increase over AP selection, and a 69.5% increase over static RSSI-based association, shown in Figure 6. These improvements highlight the system's ability to dynamically allocate spectrum and reconfigure AP associations in ways that directly benefit actively managed stations.

Table 1: Aggregate Goodput Fulfillment Rating across schemes

Wireless Network Management Systems			
GOODPUT	Load Balancing	AP Selection	RSSI
0.86	0.73	0.72	0.58

Average Goodput Fulfillment Rating (AGFR)

While aggregate goodput measures the total data delivered, it does not indicate whether station demands are satisfied. To capture this, we evaluate the *Average Goodput Fulfillment Index*, defined as the ratio of achieved goodput to requested goodput, averaged over the 3,600 seconds of the experiment:

$$\text{AGFR} = \sum_{t=1}^{T} \frac{\sum_{i=1}^{N} G_{i,t}}{\sum_{i=1}^{N} G_{i,t}^{\text{demand}}}, \tag{6}$$

where $G_{i,t}$ is the actual goodput of station i in second t, and $G_{i,t}^{\text{demand}}$ is its demand.

AGFR is important because the GOODPUT system aims for *system-wide optimization* based on station demands; the system must align delivered goodput with aggregate demand at every instant. A higher AGFR means stations receive a larger fraction of their requested rates, connecting this measure closely with user experience. It distinguishes between scenarios where high throughput masks unmet demand and cases where resources are efficiently matched to user needs. GOODPUT achieved the highest AGFR at 0.86. This was substantially higher than load balancing (0.73), AP selection (0.72), and the RSSI-based method (0.58), shown in Table 1.

Station Steering Cost

Evaluating the average steering cost per station captures the overhead introduced by GOODPUT's dynamic reconfiguration. Steering improves goodput and demand fulfillment, but each handoff causes a brief disruption. This measure quantifies how effectively GOODPUT balances reconfiguration frequency against user experience and system stability. The station steering cost is defined as follows:

$$\text{Steering Cost} = \frac{\text{Total Steering Downtime (s)}}{N \times T}, \tag{7}$$

where N is the total number of stations and T is the total experiment duration (in seconds). Total Steering Downtime is the cumulative number of seconds lost across all steering events in the system. This measure represents the fraction of experiment time, on average, that each station spends disconnected due to reconfiguration. In our one-hour experiment, the steering cost of the GOODPUT system is 0.25% with 179 steering events. In comparison, load balancing incurred a cost of 0.46% with 330 events, and AP selection incurred a cost of 0.14% with 103 events, shown in Table 2. A lower steering cost reflects fewer station reassociations, but a lower value is not necessarily better. An optimal solver accounts for steering cost explicitly but causes reassociations when this brings a net goodput benefit. The observed 19.3% goodput improvement over the AP selection baseline is due to the fact that, in the absence of sufficient steering, stations remain suboptimally connected to APs. However, application tolerance to short outages varies (e.g., buffered streaming vs. latency-sensitive workloads), and a detailed analysis of application-level impact is beyond the scope of this work.

Table 2: Steering Cost and Events in a One-Hour Experiment

Steering Cost Results		
Scheme	Steering Cost (%)	Events Number
GOODPUT	0.25	179
Load Balancing	0.46	330
AP Selection	0.14	103

Steering cost: fraction of time stations spend disconnected due to reconfiguration. Lower cost reflects fewer reassociations, but higher values can be justified if they yield better overall goodput.

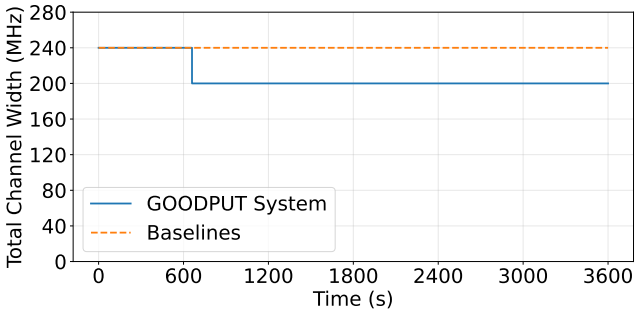

Figure 8: Total channel width usage over time. Baselines consistently use 240 MHz, while GOODPUT adapts to 200 MHz after 482 s, reducing the potential for interference.

Interference Implications of Limiting Spectrum Usage

All experiments were conducted under a fixed spectrum budget of 240 MHz across three managed APs. Baseline schemes used static 80 MHz allocations per AP, fully occupying the spectrum at all times (Figure 8). In contrast, the GOODPUT system adapts channel selection based on predicted demand and observed interference. While the GOODPUT system does not explicitly minimize channel width, every allocation decision of the MIP solver is made to optimize goodput. Due to interference by external Wi-Fi networks [13], sometimes the optimal solution uses less than the total available bandwidth. Although the optimization formulation does not consider the effects of interference on external networks, using less than the full bandwidth has this beneficial side effect.

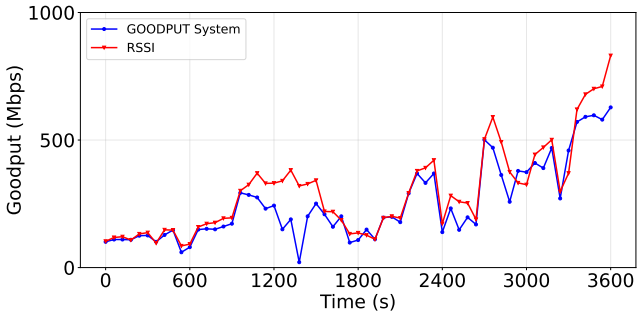

Figure 9: System-wide goodput (for legacy stations only) comparison between the GOODPUT system and RSSI-based association.

The expression for the normalized spectrum use follows:

$$\text{Avg. Spectrum Use} = \frac{\sum_{t=1}^{T} W(t)}{T}, \tag{8}$$

where $W(t)$ is the total channel width allocated at time slot t and T is the experiment duration (in seconds).

The baselines yield an average use of 240 MHz. In contrast, the GOODPUT used 206.7 MHz, about 13.9% less spectrum than the static baselines, while still achieving higher system-wide goodput.

Legacy Device Goodput

Applying Wi-Fi blinking, GOODPUT consolidates all legacy devices to a single AP to prevent uncontrolled interference with proactive stations. In our deployment, legacy devices account for 14.4% of total system goodput, and one AP provides sufficient capacity. Because legacy stations cannot be steered, consolidation localizes their impact. However, many legacy devices are not IEEE 802.11ax-compliant and cannot leverage OFDMA, increasing in-channel contention when multiple legacy clients transmit simultaneously.

Legacy throughput is typically not directly observable, but our managed emulation allowed measurement. As shown in Figure 9, legacy goodput under the GOODPUT system decreases by 15.8% compared to the RSSI baseline, while total system-wide goodput increases by 47.9%. The legacy goodput reduction could be mitigated by reserving additional bandwidth (satisfying overall performance) or by incorporating AP-side prediction models for legacy traffic. These results show that GOODPUT substantially improves proactive performance without imposing unacceptable degradation on legacy devices.

6.7 Component Ablation Analysis

The GOODPUT system achieves its gains by jointly coordinating several capabilities. To quantify each's contribution, we conducted an ablation study in which we disabled capabilities and compared performance against baselines.

Table 3: Goodput Change Relative to Full GOODPUT System

Capabilities Ablated					
FT	CB	PM	FT+CB	FT+PM	CB+PM
-5.2%	-12.7%	-30.2%	-16.7%	-33.9%	-37.2%

FT = frequency tuning, CB = channel bonding, and PM = prediction + MKP.

The system has three main capabilities: (i) active channel frequency tuning, which adaptively selects AP center frequencies to mitigate interference; (ii) dynamic channel bonding, which adjusts channel widths in real time to increase per-channel goodput; and (iii) goodput prediction with MKP-based optimization, which combines LSTM demand forecasting with a multi-knapsack formulation to assign stations. For each ablation variant, one or more components were replaced with static defaults. We evaluated six variants: three with a single component disabled and three with pairs disabled. Each was tested under identical conditions using one-hour replayed-trace experiments. We report system-wide aggregate goodput changes to reveal the net benefit of the capabilities and how limited versions of the GOODPUT system compare with the full version of the GOODPUT system. The detailed results are shown in Table 3. Each column lists the disabled capability. Resulting goodput degradations (in percentages) relative to the full system.

The ablation results clarify that the prediction + MKP module makes the largest contribution to performance. Disabling it alone causes a -30.2% change in goodput, far larger than turning off channel bonding (-12.7%) or frequency tuning (-5.2%). Prediction + MKP remains important when multiple capabilities are ablated.

7 Conclusion

This paper describes the GOODPUT system, a proactive spectrum management framework that unifies demand prediction, real-time network sensing, and dynamic channel bonding for dense WLANs. At its core, it has a real-time optimization engine that makes management decisions to maximize system-wide goodput. The GOODPUT system was prototyped using commodity hardware, and the prototype was used in evaluation. The GOODPUT system was found to improve aggregate goodput by 17.1%, relative to the most effective existing approach (which is based on load balancing), by 19.3% compared to AP selection, and by 47.9% relative to the static RSSI-based method commonly used in real-world deployments, while coexisting gracefully with legacy devices and external interference.

References

[1] Murad A. Abusubaih. An intelligence-based framework for managing WLANs: The potential of non-contiguous channel bonding. *IEEE Access*, 12:56240–56248, April 2024.

[2] Prakash Agrawal and Mythili Vutukuru. Trace based application layer modeling in ns-3. In *Proc. National Conf. on Communication (NCC)*, pages 1–6, March 2016.

[3] Doreid Ammar, Thomas Begin, and Isabelle Guerin-Lassous. A new tool for generating realistic internet traffic in ns-3. In *Proc. Int. ICST Conf. on Simulation Tools and Techniques*, pages 81–83, 2011.

[4] Stefano Avallone, Pasquale Imputato, Getachew Redieteab, Chittabrata Ghosh, and Sumit Roy. Will OFDMA improve the performance of 802.11 Wi-Fi networks? *IEEE Wireless Communications*, 28(3):100–107, June 2021.

[5] Rozin Badeel, Shamala K. Subramaniam, Abdullah Muhammed, and Zurina Mohd Hanapi. A multicriteria decision-making framework for access point selection in hybrid LiFi/Wi-Fi networks using integrated AHP–VIKOR technique. *Sensors*, 23(3):1312, January 2023.

[6] Suzan Bayhan, Estefanía Coronado, Roberto Riggio, and Anatolij Zubow. User-AP association management in software-defined WLANs. *IEEE Trans. on Network and Service Management*, 17(3):1838–1852, September 2020.

[7] Syed Hashim Raza Bukhari, Mubashir Husain Rehmani, and Sajid Siraj. A survey of channel bonding for wireless networks and guidelines of channel bonding for futuristic cognitive radio sensor networks. *IEEE Communications Surveys & Tutorials*, 18(2):924–948, November 2015.

[8] Lelio Campanile, Marco Gribaudo, Mauro Iacono, Fiammetta Marulli, and Michele Mastroianni. Computer network simulation with ns-3: A systematic literature review. *Electronics*, 9(2):272, February 2020.

[9] Jingrong Chen, Yongji Wu, Shihan Lin, Yechen Xu, Xinhao Kong, Thomas Anderson, Matthew Lentz, Xiaowei Yang, and Danyang Zhuo. Remote procedure call as a managed system service. In *Proc. USENIX Symp. on Networked Systems Design and Implementation (NSDI)*, pages 141–159, April 2023.

[10] Xi Chen, Yue Zhao, Brian Peck, and Daji Qiao. SAP: Smart access point with seamless load balancing multiple interfaces. In *Proc. INFOCOM*, pages 1458–1466, March 2012.

[11] Estefanía Coronado, Roberto Riggio, José Villalón, and Antonio Garrido. Wi-balance: Channel-aware user association in software-defined Wi-Fi networks. In *Proc. IEEE/IFIP Network Operations and Management Symp. (NOMS)*, pages 1–9, April 2018.

[12] Estefanía Coronado, Roberto Riggio, José Villalón, and Antonio Garrido. Wi-balance: Channel-aware user association in software-defined Wi-Fi networks. In *Proc. Network Operations and Management Symp.*, pages 1–9, April 2018.

[13] Yousri Daldoul, Djamal-Eddine Meddour, and Adlen Ksentini. IEEE 802.11ac: Effect of channel bonding on spectrum utilization in dense environments. In *Proc. IEEE Int. Conf. on Communications*, pages 1–6, May 2017.

[14] Lara Deek, Eduard Garcia-Villegas, Elizabeth Belding, Sung-Ju Lee, and Kevin Almeroth. Intelligent channel bonding in 802.11n WLANs. *IEEE Trans. on Mobile Computing*, 13(6):1242–1255, June 2013.

[15] Paal E. Engelstad and Olav N. Østerbø. Non-saturation and saturation analysis of IEEE 802.11e EDCA with starvation prediction. In *Proc. Int. Symp. Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 224–233, October 2005.

[16] Chao Fang, Xiangheng Meng, Zhaoming Hu, Fangmin Xu, Deze Zeng, Mianxiong Dong, and Wei Ni. AI-driven energy-efficient content task offloading in cloud-edge-end cooperation networks. *IEEE Open J. of the Computer Society*, 3:162–171, September 2022.

[17] Eduard Garcia, Josep L. Ferrer, Elena Lopez-Aguilera, Rafael Vidal, and Josep Paradells. Client-driven load balancing through association control in IEEE 802.11 WLANs. *European Trans. on Telecommunications*, 20(5):494–507, October 2008.

[18] Merkebu Girmay, Adnan Shahid, Vasilis Maglogiannis, Dries Naudts, and Ingrid Moerman. Machine learning enabled Wi-Fi saturation sensing for fair coexistence in unlicensed spectrum. *IEEE Access*, 9:42959–42974, March 2021.

[19] Blas Gómez, Estefanía Coronado, José M. Villalón, Roberto Riggio, and Antonio Garrido. WiMCA: multi-indicator client association in software-defined Wi-Fi networks. *Wireless Networks*, 27(5):3109–3125, May 2021.

[20] Google Optimization Team. OR-Tools: Google optimization tools. https://developers.google.com/optimization, February 2025. Version accessed February 2025.

[21] Yixue Hao, Yingying Jiang, Tao Chen, Donggang Cao, and Min Chen. itaskoffloading: Intelligent task offloading for a cloud-edge collaborative system. *IEEE Network*, 33(5):82–88, October 2019.

[22] Felix Hernandez-Campos and Maria Papadopouli. A comparative measurement study the workload of wireless access points in campus networks. In *Proc. Int. Symp. Personal, Indoor and Mobile Radio Communications*, pages 1776–1780, September 2005.

[23] Pedro Enrique Iturria-Rivera and Melike Erol-Kantarci. Competitive multi-agent load balancing with adaptive policies in wireless networks. In *Proc. Consumer Communications & Networking Conf. (CCNC)*, pages 796–801, January 2022.

[24] Han Ji, Xiping Wu, Qiang Wang, Stephen J. Redmond, and Iman Tavakkolnia. Adaptive target-condition neural network: DNN-aided load balancing for hybrid LiFi and Wi-Fi networks. *IEEE Trans. on Wireless Communications*, 23(7):7307–7318, July 2024.

[25] Falguni Jindal, Rishabh Jamar, and Prathamesh Churi. Future and challenges of internet of things. *Int. J. of Comput. Sci. Inf. Technol.*, 10(2):13–25, April 2018.

[26] Caihong Kai, Yuting Liang, Tianyu Huang, and Xu Chen. To bond or not to bond: An optimal channel allocation algorithm for flexible dynamic channel bonding in WLANs. In *Proc. IEEE Vehicular Technology Conf. (VTC-Fall)*, pages 1–6, September 2017.

[27] Raja Karmakar, Samiran Chattopadhyay, and Sandip Chakraborty. SmartBond: A deep probabilistic machinery for smart channel bonding in IEEE 802.11ac. In *Proc. IEEE Conf. on Computer Communications (INFOCOM)*, pages 2599–2608, July 2020.

[28] Sami Khairy, Mengqi Han, Lin X. Cai, Yu Cheng, and Zhu Han. A renewal theory based analytical model for multi-channel random access in IEEE 802.11ac/ax. *IEEE Trans. on Mobile Computing*, 18(5):1000–1013, May 2019.

[29] Muhammad Asif Khan, Ridha Hamila, Adel Gastli, Serkan Kiranyaz, and Nasser Ahmed Al-Emadi. ML-based handover prediction and AP selection in cognitive Wi-Fi networks. *J. of Network and Systems Management*, 30(4):72, August 2022.

[30] Yuan Le, Liran Ma, Hongjun Yu, Xiuzhen Cheng, Yong Cui, Mznah A. Al-Rodhaan, and Abdullah Al-Dhelaan. Load balancing access point association schemes for IEEE 802.11 wireless networks. In *Proc. Int. Conf. Wireless Algorithms, Systems, and Applications*, pages 271–279, 2011.

[31] Ben Mecklenburg, Ahmed Hasan Ansari, BjÖrn Richerzhagen, Michael Bahr, and Georg Carle. Seamless roaming based on distributed multi-link operation over IEEE 802.11bn. In *Proc. IEEE Int. Conf. on Factory Communication Systems (WFCS)*, pages 1–8, June 2025.

[32] Rajasekar Mohan, K. Venkat Ramnan, and J. Manikandan. Predicting the throughput of next generation IEEE 802.11 WLANs in dense deployments. *Procedia Computer Science*, 203:24–31, August 2022.

[33] Erfan Mozaffariahrar, Fabrice Theoleyre, and Michael Menth. A survey of Wi-Fi 6: Technologies, advances, and challenges. *Future Internet*, 14(10):293, October 2022.

[34] Erfan Mozaffariahrar, Fabrice Theoleyre, and Michael Menth. A survey of Wi-Fi 6: Technologies, advances, and challenges. *Future Internet*, 14(10):293, October 2022.

[35] Amr Nabil, Mohammad J. Abdel-Rahman, Allen B. MacKenzie, and Fahid Hassan. A stochastic optimization framework for channel bonding in wireless LANs under demand uncertainty. *IEEE Trans. on Wireless Communications*, 19(11):7528–7542, August 2020.

[36] Hung Nguyen, Duc Long Pham, Mau Hien Doan, Thi Thanh Sang Nguyen, Duc Anh Vu Dinh, and Adrianna Kozierkiewicz. Reinforcement learning for optimizing Wi-Fi access channel selection. In *Proc. Advances in Computational Collective Intelligence Int. Conf. (ICCCI)*, pages 334–347, September 2021.

[37] Eng Hwee Ong, Jamil Y. Khan, and Kaushik Mahata. On dynamic load distribution algorithms for multi-AP WLAN under diverse conditions. In *Proc. Wireless Communication and Networking Conf.*, pages 1–6, April 2010.

[38] Vivek Parashar, Ramgopal Kashyap, Ali Rizwan, Dimitrios A Karras, Gilder Cieza Altamirano, Ekta Dixit, and Fardin Ahmadi. Aggregation-based dynamic channel bonding to maximise the performance of wireless local area networks (WLAN). *Wireless Communications and Mobile Computing*, 2022(1):4464447, June 2022.

[39] Hang Qi, Hao Huang, Zhiqun Hu, Xiangming Wen, and Zhaoming Lu. On-demand channel bonding in heterogeneous WLANs: A multi-agent deep reinforcement learning approach. *Sensors*, 20(10):2789, May 2020.

[40] Selenium contributors. Selenium: browser automation framework. https://www.selenium.dev, 2023. Version 4.0.

[41] Seung seob Lee, TaeYoung Kim, SuKyoung Lee, Kyungsoo Kim, Yoon Hyuk Kim, and Nada Golmie. Dynamic channel bonding algorithm for densely deployed 802.11ac networks. *IEEE Trans. on Communications*, 67(12):8517–8531, September 2019.

[42] Warren Severin. LinSSID: Graphical Wi-Fi scanner for Linux. https://sourceforge.net/projects/linssid/, 2018. Version 3.6, GPLv3.

[43] ML Sharma, Sachin Kumar, and Nipun Mehta. Internet of things application, challenges and future scope. *Int. Research J. of Engineering and Technology (IRJET)*, 5(2):1376–1382, February 2018.

[44] Wooi King Soo, Teck-Chaw Ling, Aung Htein Maw, and Su Thawda Win. Survey on load-balancing methods in 802.11 infrastructure mode wireless networks for improving quality of service. *ACM Computing Surveys*, 51(2):1–21, February 2018.

[45] Szymon Szott, Katarzyna Kosek-Szott, Piotr Gawłowicz, Jorge Torres Gómez, Boris Bellalta, Anatolij Zubow, and Falko Dressler. Wi-Fi meets ML: A survey on improving IEEE 802.11 performance with machine learning. *IEEE Communications Surveys & Tutorials*, 24(3):1843–1893, June 2022.

[46] Nada Chendeb Taher, Yacine Ghamri-Doudane, Bachar El Hassan, and Nazim Agoulmine. An accurate analytical model for 802.11e EDCA under different traffic conditions with contention-free bursting. *J. of Computer Networks and Communications*, 2011(1):136585, June 2011.

[47] Godfrey Tan and John V. Guttag. Time-based fairness improves performance in multi-rate WLANs. In *Proc. USENIX Annual Tech. Conf.*, pages 269–282, June 2004.

[48] Juan Pablo Vielma. Mixed integer linear programming formulation techniques. *SIAM Review*, 57(1):3–57, 2015.

[49] Ting-Hui Wang, Li-Hsiang Shen, and Kai-Ten Feng. Distributed multi-agent deep q-learning for fast roaming in IEEE 802.11ax Wi-Fi systems. In *Proc. Consumer Communications and Networking Conf. (CCNC)*, pages 433–438, March 2024.

[50] Wenjia Wu, Yujing Liu, Jiazhi Yao, Xiaolin Fang, Feng Shan, Ming Yang, Zhen Ling, and Junzhou Luo. Learning-aided client association control for high-density WLANs. *Computer Networks*, 212:109043, 2022.

[51] Lei Xia, Sanjay Kumar, Xue Yang, Praveen Gopalakrishnan, York Liu, Sebastian Schoenberg, and Xingang Guo. Virtual Wi-Fi: bring virtualization from wired to wireless. *ACM SIGPLAN Notices*, 46(7):181–192, March 2011.

[52] Fengyuan Xu, Xiaojun Zhu, Chiu C. Tan, Qun Li, Guanhua Yan, and Jie Wu. Smartassoc: Decentralized access point selection algorithm to improve throughput. *IEEE Trans. on Parallel and Distributed Systems*, 24(12):2482–2491, December 2013.

[53] Hujun Yin and Siavash Alamouti. OFDMA: A broadband wireless access technology. In *Proc. Sarnoff Symp.*, pages 1–4, March 2006.

[54] Zhenzhe Zhong, Parag Kulkarni, Fengming Cao, Zhong Fan, and Simon Armour. Issues and challenges in dense Wi-Fi networks. In *Proc. Int. Wireless Communications and Mobile Computing Conf. (IWCMC)*, pages 947–951, August 2015.