

# COMP4336/9336 Mobile Data Networking: 2024 Term 3 Individual Term Project/Assignment

Written by Jiayang Jiang z5319476

## 1. Data Collection:

For this project, WiFi traffic data was collected using **Wireshark** at two crowded locations on the University of New South Wales (UNSW) campus: the **Main Library** and the **Law Library**. The data collection process was carried out during both **daytime and nighttime** sessions to ensure **time diversity**, capturing the variations in WiFi network activity at different periods of the day. The libraries were chosen due to their high density of users and **WiFi Access Points (APs)**, making them ideal locations to observe significant WiFi retransmissions and collisions, which are crucial for this study.

To gather a comprehensive dataset, I focused on capturing WiFi traffic from the **2.4 GHz frequency band**, ensuring that data from **all available channels (1-13)** within the 2.4 GHz band was collected. Data collection was conducted across **multiple days** and at **various times**, spanning different weeks, to capture any temporal fluctuations in WiFi usage patterns and ensure the diversity needed for effective model training and evaluation.

The tools used for data collection included a **Windows laptop** for initial testing, but due to hardware limitations in capturing traffic from other APs, a **MacBook** (Borrowed from someone else) was ultimately used for more comprehensive data capture. This allowed for the collection of data from all accessible APs in the targeted locations, providing a richer dataset for subsequent analysis.

The pictures of collection locations:

-Main library and Law library:



Data summary:

I collected about 2 million packets of data from channels 1-13.

```
<class 'pandas.core.frame.DataFrame'>
Index: 225894 entries, 11971 to 91711
Data columns (total 18 columns):
```

	Time	Length	Channel	Frame Number	Duration	Data rate	Sequence number	channel utilization
count	225894.000000	225894.000000	225894.000000	225894.000000	225894.000000	225892.00000	147354.000000	225894.000000
mean	97.821744	308.752543	5.214446	30453.746872	316.091871	16.97141	1991.226326	11.323408
std	92.530756	341.916482	3.805147	24265.409544	456.628754	26.85577	1205.409179	8.917809
min	0.000000	40.000000	1.000000	1.000000	0.000000	1.00000	0.000000	0.068765
25%	34.571418	64.000000	1.000000	10216.000000	68.000000	11.00000	947.000000	4.734471
50%	71.766804	308.000000	6.000000	25830.500000	252.000000	12.00000	1978.500000	8.838930
75%	129.499668	409.000000	6.000000	46044.750000	280.000000	12.00000	3030.000000	16.335141
max	609.899100	3336.000000	13.000000	114147.000000	12224.000000	243.80000	4095.000000	37.603982

## 2. Performance Analysis:

### Data processing:

#### 1. Data Collection:

- WiFi data was captured using Wireshark, focusing on 802.11 frames to gather information on wireless communication.
- Filters such as wlan were applied to ensure only WiFi-related frames were captured. Specifically, the filter wlan.fc.retry == 1 was used to identify all retransmitted packets.

No.	Time	Source	Destination	Length	Signal strength (RSSI)	Retry	Type/Subtype	SSID
353	3.8217716	b2:54:49:b4:3e:2b	ff:ff:ff:ff:ff:ff	134	-75 dBm	Frame is being retransmitted	Probe Request	<MISSING>
10.	11.840561	da:4f:4a:c4:99:7b	ff:ff:ff:ff:ff:ff	134	-62 dBm	Frame is being retransmitted	Probe Request	<MISSING>
10.	12.73871	fa:e8:f1:21:ea:81	ff:ff:ff:ff:ff:ff	134	-62 dBm	Frame is being retransmitted	Probe Request	<MISSING>
21.	24.009302	fe:d8:8b:e8:bc:06	ff:ff:ff:ff:ff:ff	134	-57 dBm	Frame is being retransmitted	Probe Request	<MISSING>
21.	24.057913	fe:d8:8b:e8:bc:06	ff:ff:ff:ff:ff:ff	140	-58 dBm	Frame is being retransmitted	Probe Request	"高铁"
24.	27.142418	72:85:15:b8:d1:e5	ff:ff:ff:ff:ff:ff	134	-68 dBm	Frame is being retransmitted	Probe Request	<MISSING>
26.	29.373262	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	182	-66 dBm	Frame is being retransmitted	Association Request	"unide"
27.	30.632930	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	141	-65 dBm	Frame is being retransmitted	Probe Request	"unide"
27.	30.635572	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	141	-65 dBm	Frame is being retransmitted	Probe Request	"unide"
27.	30.650764	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	141	-68 dBm	Frame is being retransmitted	Probe Request	"unide"
27.	30.651617	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	72	-63 dBm	Frame is being retransmitted	Authentication	
27.	30.657200	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	72	-67 dBm	Frame is being retransmitted	Authentication	
27.	30.659106	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	72	-62 dBm	Frame is being retransmitted	Authentication	
27.	30.659875	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	72	-66 dBm	Frame is being retransmitted	Authentication	
27.	30.659804	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	72	-67 dBm	Frame is being retransmitted	Authentication	
27.	30.659899	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	72	-66 dBm	Frame is being retransmitted	Authentication	
27.	30.660194	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	72	-72 dBm	Frame is being retransmitted	Authentication	
27.	30.660206	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	72	-69 dBm	Frame is being retransmitted	Authentication	
27.	30.660210	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	72	-67 dBm	Frame is being retransmitted	Authentication	
27.	30.660651	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	72	-70 dBm	Frame is being retransmitted	Authentication	
27.	30.661289	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	72	-66 dBm	Frame is being retransmitted	Authentication	
27.	30.661301	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	72	-66 dBm	Frame is being retransmitted	Authentication	
28.	30.769506	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	72	-61 dBm	Frame is being retransmitted	Authentication	
28.	30.822000	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	182	-71 dBm	Frame is being retransmitted	Association Request	"unide"
28.	30.891715	HuaweiDevice_79:8d...	HewlettPacka_Be:b5...	85	-52 dBm	Frame is being retransmitted	QoS Data	

Frame 353: 134 bytes on wire (1072 bits), 134 bytes captured (1072 bits) on interface wlan0mon, id 0  
Radiotap Header v0, Length 38  
802.11 radio information  
IEEE 802.11 Probe Request, Flags: ....R...C  
IEEE 802.11 Wireless Management

0000 00 00 26 00 2f 43 00 00 20 08 00 a0 20 08 00 00 ...&/B...  
0010 1d ff a0 01 00 00 00 10 02 6c 09 a0 00 bb 00 ...@-d...  
0020 00 00 b5 00 b5 01 40 08 64 00 ff ff ff ff ff ff ...TE...  
0030 b2 54 49 b4 3e 2b ff ff ff ff ff 00 c0 00 00 ...  
0040 01 00 02 84 0b 0c 12 96 18 24 03 01 01 32 04 30 ...\$-2.0  
0050 48 60 6c 2d 1a ef 01 17 ff ff 00 00 00 00 00 00 ...H1...  
0060 00 00 00 01 00 00 00 00 00 00 00 00 00 00 7f ...@-P...  
0070 08 01 00 00 00 00 00 00 40 dd 07 00 50 f2 08 00 ...@...  
0080 19 00 0b 31 f6 16 ...i...

Packets: 51188 - Displayed: 441 (0.9%)

- The captured data was processed and exported into CSV files for further analysis.

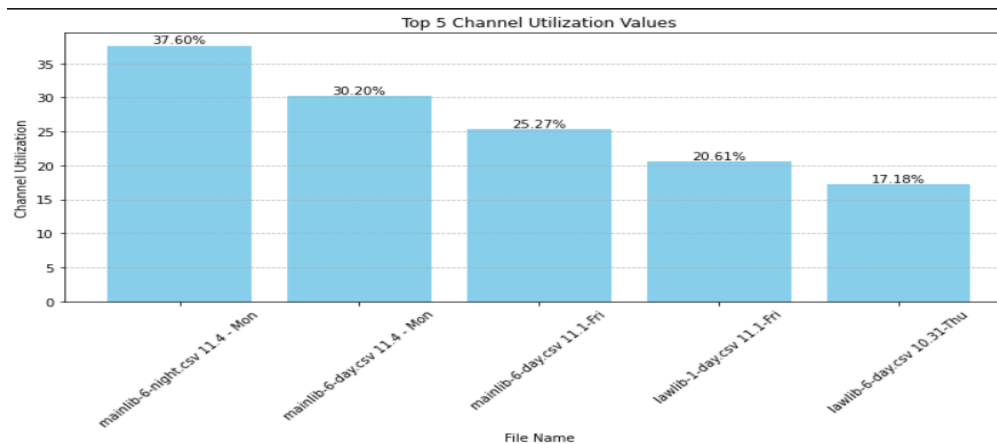
#### 2. Data Merging and Feature Expansion:

- Additional feature columns were added, including weather, channel utilization, location, and datetime, to enrich the analysis.
- Preprocessing steps included converting the duration column into integer type to enable statistical analysis.
- All trace files were merged to create a comprehensive dataset for analysis.

#### 3. Channel Utilization Calculation:

- Channel utilization was calculated to measure the frequency of frame transmission attempts on a particular channel relative to the total monitoring time. This served as a metric for network load, providing insights into how busy the network was during specific times and locations.

$$\text{Channel Utilization} = \frac{\text{Time Occupied by Packets}}{\text{Total Monitoring Time}}$$



## Retransmission Analysis

### 1. Processing the Retry Flag:

- Using Python's pandas library, the Retry column was processed to ensure retransmitted frames were correctly marked.
- If the Retry column contained the keyword "not being retransmitted," the frame was marked as 0, indicating that it was not retransmitted; otherwise, it was marked as 1, indicating a retransmission.

The "Retry" distribution:

```
# check the "Retry" distribution
print(df['Retry'].value_counts())
```

✓ 0.0s

```
Retry
0    2108049
1     150888
Name: count, dtype: int64
```

### 2. Calculating Retransmission Counts:

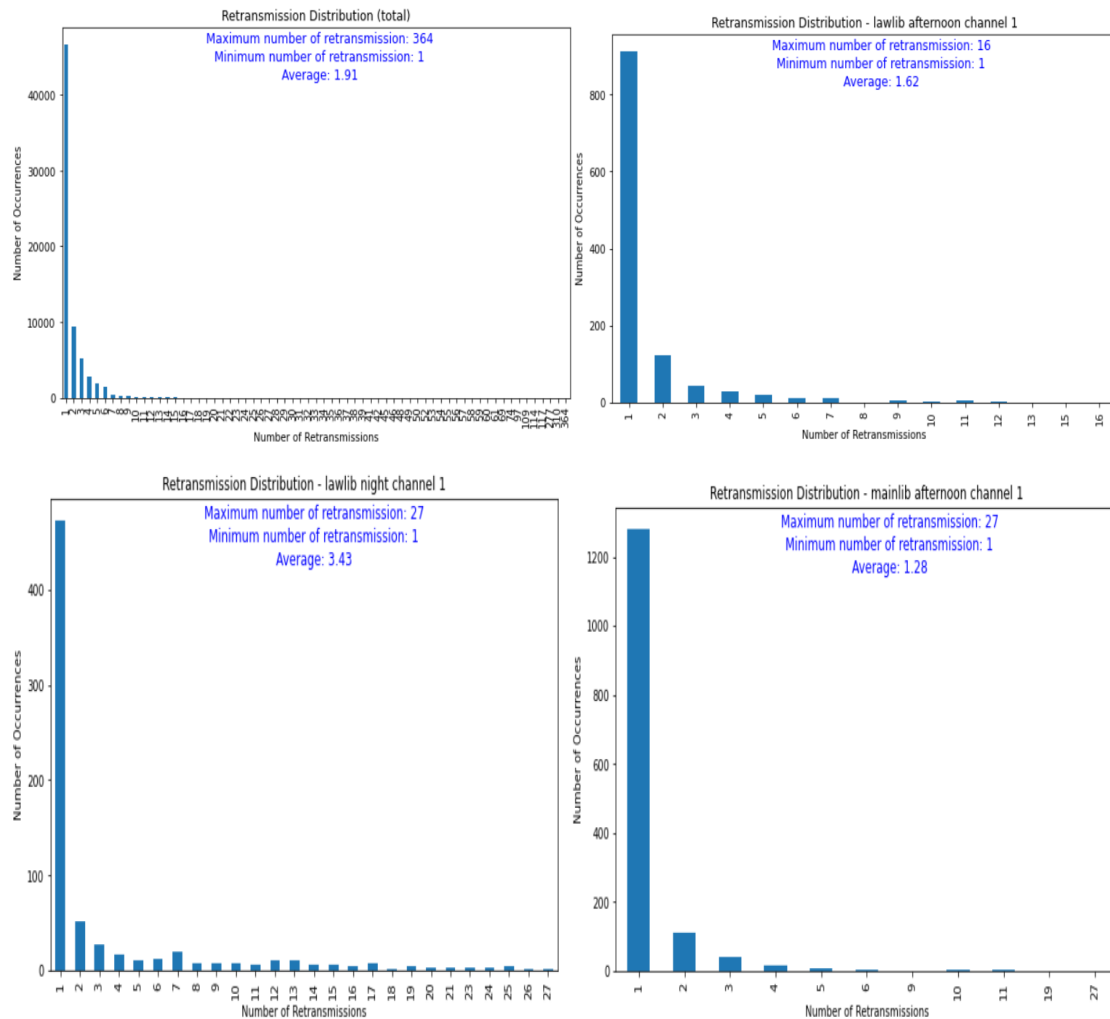
- Each packet was identified based on the combination of Source (source address), Destination (destination address), and Sequence number to uniquely distinguish packet flows.
- The retransmission count for each unique flow was recorded by grouping by these identifiers and calculating the number of retransmission occurrences.

```
# Record the number of retransmissions of each packet by Source, Destination, and Sequence number
retransmission_counts = df[df['is_retransmission']].groupby(['Source', 'Destination', 'Sequence number']).size().reset_index(name='Retransmission_Count')
```

### 3. Statistical Summary and Visualization:

- The maximum, minimum, and average retransmission counts were computed to summarize retransmission behavior across different packet flows.
- A bar chart was used to visualize the retransmission distribution, where:
  - The x-axis represents the number of retransmissions.
  - The y-axis represents the frequency of occurrences for each retransmission count.
- Key statistics such as max, min, and average retransmissions were annotated dynamically on the chart to provide a comprehensive view of the retransmission patterns.

I compared the retransmissions distribution of 4 different places:

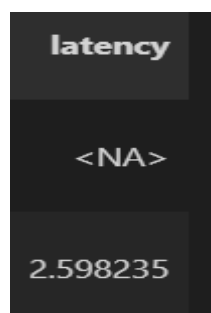


#### 4. Network Performance Analysis:

The congested network was selected based on the channel utilization being the highest among all observed networks. Channel utilization is a key indicator of how much of the available bandwidth is being used, and high utilization often correlates with congestion and network inefficiencies. By selecting the network with the highest channel utilization, we can effectively study the impacts of congestion on network performance.

- **Latency Calculation**

Latency measures the time taken for a data packet to reach its destination and receive an acknowledgment (ACK) frame. To calculate latency, I located a data packet and the corresponding ACK packet. The latency was then computed as the difference between the timestamp of the ACK packet and the timestamp of the original data packet. This approach provides an accurate assessment of end-to-end delay.

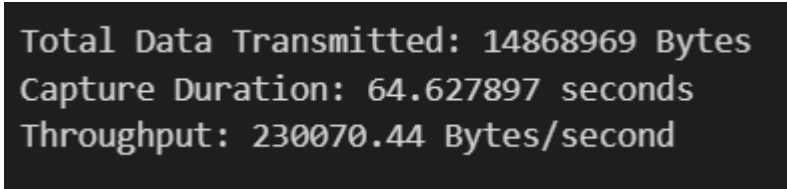


- **Throughput Calculation**

Throughput represents the total amount of data successfully transmitted over a period. It provides insight into the data transfer capacity of the network.

**Methodology:**

- Filtered data packets by device MAC address.
- Summed up the total data transmitted by a given device.
- Calculated the duration of the capture from the timestamp of the first and last packet.



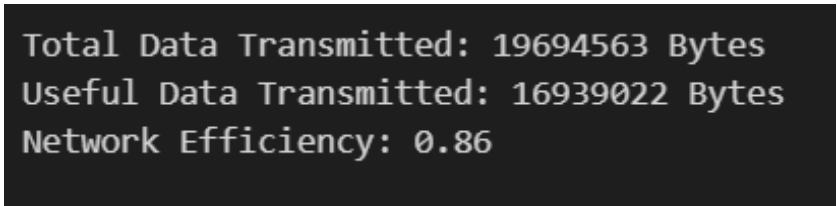
```
Total Data Transmitted: 14868969 Bytes
Capture Duration: 64.627897 seconds
Throughput: 230070.44 Bytes/second
```

- **Efficiency Calculation**

Efficiency is an important metric that quantifies how much useful data was transmitted relative to the total number of transmissions, taking into account retransmissions.

**Methodology:**

- Identified all packets that were not retransmissions as useful data.
- Calculated the total amount of data, including retransmissions.



```
Total Data Transmitted: 19694563 Bytes
Useful Data Transmitted: 16939022 Bytes
Network Efficiency: 0.86
```

### 3. Results and Findings

- **Latency Analysis**

The latency for the selected network showed a consistent pattern under normal conditions. However, during periods of congestion, the latency significantly increased due to delayed acknowledgments. A scatter plot of retransmission frequency versus latency was generated to illustrate the impact of retransmissions on latency. The plot revealed that as retransmission frequency increased, latency tended to rise, indicating reduced network efficiency.

- **Throughput Analysis**

Throughput was calculated over a set period for the selected device. During times of heavy retransmissions, throughput dropped significantly, highlighting the inefficiencies brought by packet collisions and retransmissions. The relationship between retransmission frequency and throughput was visualized using a line graph, showing an inverse correlation.

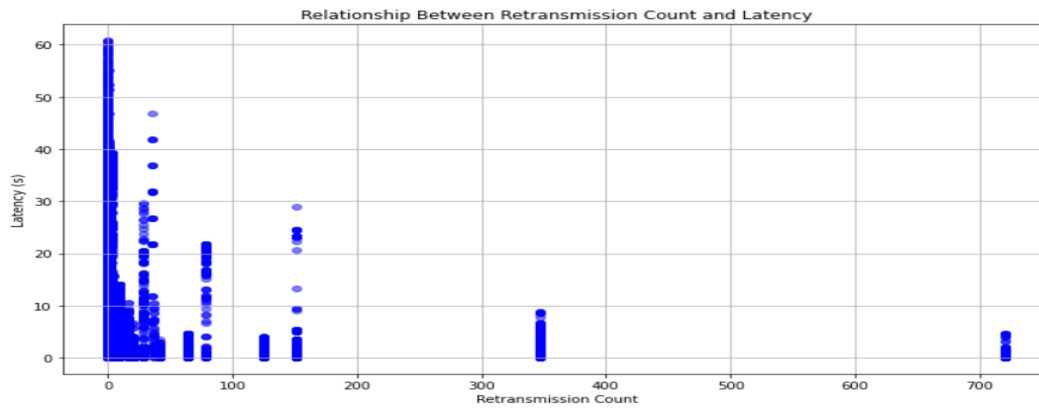
- **Efficiency Analysis**

Efficiency was evaluated by comparing the amount of useful data transmitted against the total data (including retransmissions). During network congestion, efficiency dropped significantly due to increased retransmission attempts. The results indicated that higher retransmission frequencies led to lower efficiency, as more data was retransmitted rather than successfully delivered.

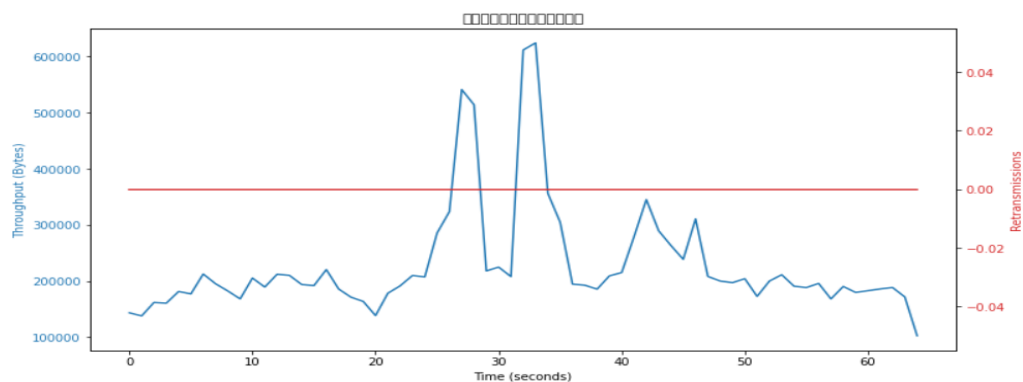
### 4. Visualization

To better understand the relationship between retransmission frequency and network performance metrics, the following charts were created:

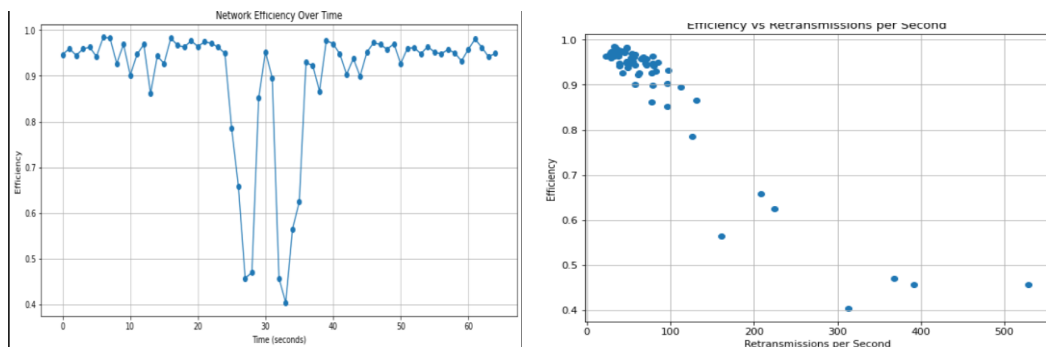
- **Scatter Plot:** Depicting the relationship between retransmission count and latency.



- Showing the relationship between retransmission frequency and throughput. The graph indicates that higher retransmission rates are associated with a decline in throughput.



- Representing network efficiency versus retransmission frequency, where the efficiency drops as retransmissions increase.



### 3. Machine Learning Evaluation:

I explored the application of machine learning to predict the likelihood of packet retransmissions in a WiFi network environment. By leveraging features like signal strength, packet size, and channel utilization, I aimed to understand which factors contribute most significantly to retransmission events. For this purpose, I used three machine learning models: k-Nearest Neighbors (kNN), Random Forest, and Naive Bayes. The following sections elaborate on feature extraction, model training and evaluation, and the impact of different features on model performance.

#### Feature Extraction:

The dataset contains several features collected using Wireshark, representing key parameters of the WiFi environment. The features used in this study include:

- Signal Strength (RSSI): Indicates the strength of the received signal.
- Packet Length: Represents the size of each data packet.
- Channel Utilization: Reflects the congestion level of the channel.
- Retry (Target Variable): Indicates whether a packet was retransmitted.
- Data Rate: The rate at which data was transmitted.
- Location and Weather: Environmental conditions that could influence the network performance.
- Channel, Frame Number, Sequence Number, etc.: Other features capturing details about the packet and environment.

These features were used as inputs to the machine learning models to predict whether a packet would be retransmitted.

### Model Training and Evaluation:

To predict packet retransmissions, I trained three machine learning models: kNN, Random Forest, and Naive Bayes. Each of these models was evaluated using accuracy, precision, recall, and F1-score metrics on a portion of the dataset reserved for validation.

### k-Nearest Neighbors (kNN)

kNN is a simple, instance-based learning algorithm that classifies a sample based on the majority vote of its k-nearest neighbors. For this task, kNN was effective in scenarios with distinct clusters of features, such as signal strength and channel utilization. However, due to the high dimensionality of the dataset, kNN struggled with overlapping classes, resulting in relatively lower accuracy. The model's performance was influenced significantly by the choice of k value and distance metric.

### Random Forest

Random Forest, an ensemble method using multiple decision trees, provided robust predictions for packet retransmissions. By capturing feature interactions, the Random Forest model was able to identify complex relationships between variables like signal strength, packet length, and channel utilization. The model's feature importance analysis showed that signal strength, channel utilization, and data rate were the most critical factors influencing retransmission likelihood. This model achieved great accuracy and recall compared to Naive Bayes.

### Naive Bayes

Naive Bayes, a probabilistic classifier, assumes conditional independence between features. While it performed reasonably well for this problem, it was limited by the dependencies between features like signal strength and channel utilization, which are not strictly independent. Despite this, Naive Bayes offered valuable insights into the distribution of features and their respective impact on retransmissions. It performed particularly well in identifying non-retransmission cases, yielding high precision for the majority class.

### Comparison of Feature Sets

During model training, I experimented with different combinations of features to understand their influence on model performance. Notably, including signal strength, channel utilization, and data rate consistently improved model accuracy. Removing **location, weather, source, destination** had minimal effect, suggesting that these features were less relevant to retransmission prediction.

- Signal Strength (RSSI): This feature was one of the most influential. Weak signal strength was often associated with higher retransmission rates, which is intuitive given the increased likelihood of packet loss.
- Channel Utilization: High channel utilization was correlated with higher retransmissions due to

congestion. This feature was particularly important for the Random Forest model, contributing significantly to its ability to classify packets accurately.

- **Packet Length:** Larger packet sizes increased the likelihood of retransmissions, as they were more prone to errors, particularly in poor signal conditions.

#### 4. Protocol Enhancement Design

The IEEE 802.11 standard defines the protocols for implementing wireless local area networks (WLANs). It operates in the 2.4 GHz and 5 GHz frequency bands, utilizing mechanisms such as Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) to manage medium access.

The standard also includes features like Request to Send/Clear to Send (RTS/CTS), which helps to mitigate the hidden node problem by ensuring that neighboring devices are aware of the ongoing transmission. These mechanisms, while effective, can introduce latency and increase retransmission rates under high network load or interference conditions.

##### Proposed Enhancements

##### 1. Adaptive Retransmission Mechanisms [1]

- **Design Rationale:** The adaptive retransmission mechanism aims to dynamically adjust retransmission strategies based on real-time network conditions. By monitoring metrics such as signal strength, congestion level, and packet error rate, the system can intelligently decide how to handle retransmissions. For example, if the network is congested, it could increase the backoff time to reduce the probability of collisions or adjust the maximum retransmission attempts based on the predicted likelihood of success.
- **Technical Implementation:**
  - Machine learning models can be integrated into the MAC (Medium Access Control) layer of protocol. These models could be trained using supervised learning techniques to predict optimal backoff times and retransmission limits.
  - The adaptive algorithm could use Reinforcement Learning (RL) to dynamically adjust backoff parameters and retransmission strategies, based on the network state and historical performance.
- **Feasibility Analysis:**
  - **Complexity:** The implementation of adaptive mechanisms requires additional computational power, as devices need to monitor and analyze real-time data. However, this could be managed by optimizing the machine learning models to run efficiently on embedded hardware.
  - **Scalability:** Adaptive mechanisms are highly scalable since each device independently optimizes its behavior, making it suitable for both small and large networks.
  - **Impact on Standards:** Changes would mainly impact the MAC layer of the WiFi stack, and could be made backward compatible by ensuring legacy devices use a fixed retransmission strategy while new devices use the adaptive model.
- **Challenges:**
  - The primary challenge lies in the computational overhead and the requirement for real-time decision-making capabilities.
  - Ensuring fairness between devices with different capabilities (e.g., older vs. newer devices) could also be challenging.

##### 2. Dynamic Channel Allocation [2]



- **Design Rationale:** Dynamic channel allocation aims to reduce interference by allowing Access Points (APs) to continuously monitor and switch to the optimal channel based on real-time congestion and interference levels. Machine learning models can be employed to predict which channel is most likely to provide stable communication, thus minimizing retransmission rates.
- **Technical Implementation:**
  - The AP continuously collects data on each available channel, including noise levels, channel occupancy, and historical performance.
  - A machine learning-based decision model is used to select the optimal channel. The model could be a classification or regression model that predicts the best channel based on collected features.
  - The AP could also employ reinforcement learning to learn which channels perform better over time under different network conditions.
- **Feasibility Analysis:**
  - **Complexity:** Implementing dynamic channel switching adds complexity, particularly in ensuring that channel switches do not disrupt ongoing sessions. Coordination among neighboring APs is also required to avoid multiple APs choosing the same channel simultaneously.
  - **Scalability:** Dynamic channel allocation is effective in both dense urban networks and home networks. The machine learning model should be lightweight enough to run on standard AP hardware.
  - **Impact on Standards:** This enhancement would require updates to the channel selection and coordination protocols to ensure compatibility and efficient operation in mixed-device environments.
- **Challenges:**
  - Frequent channel switching could lead to instability if not carefully managed.
  - Synchronization among neighboring APs to avoid interference is a significant challenge that requires additional protocol support.

### 3. Error Control Enhancements [3]

- **Design Rationale:** Introducing advanced error detection and correction mechanisms, such as Forward Error Correction (FEC), aims to reduce the need for retransmissions by enabling the receiver to correct certain errors without requesting a retransmission. This method is particularly useful in scenarios where retransmissions are costly, such as high-latency networks.
- **Technical Implementation:**
  - FEC can be implemented by adding redundant bits to each data packet, allowing the receiver to detect and correct errors within a certain limit.
  - Machine learning models could be employed to predict the optimal level of redundancy based on network conditions, ensuring that overhead is minimized while maintaining reliability.
  - An adaptive FEC mechanism could adjust the redundancy rate dynamically, depending on the observed error rates and network congestion.
- **Feasibility Analysis:**
  - **Complexity:** Adding FEC increases the computational load for both encoding at

the sender and decoding at the receiver. However, this complexity is manageable with modern hardware.

- **Scalability:** FEC is suitable for both small-scale and large-scale networks, especially in environments with high levels of interference. The trade-off lies in balancing the amount of redundancy to minimize both packet loss and bandwidth usage.
- **Impact on Standards:** Incorporating FEC would require updates to the physical layer specifications in the WiFi protocol, making this enhancement more challenging to standardize across different hardware platforms.
- **Challenges:**
  - Finding the optimal balance between redundancy and overhead is challenging, as excessive redundancy can reduce network efficiency.
  - Compatibility with legacy devices that do not support advanced FEC could be an issue, requiring careful implementation to ensure backward compatibility.

### Critical Thinking

- Rationale Behind Proposals:
  - The adaptive retransmission mechanism addresses network variability by allowing devices to tailor their behavior to current conditions, reducing unnecessary retransmissions and improving efficiency.
  - Dynamic channel allocation helps avoid congestion and interference by proactively selecting the optimal communication channel, minimizing collisions and packet losses.
  - Error control enhancements, through FEC, reduce the number of retransmissions needed by enabling the correction of errors without incurring additional round-trip delays.
- Trade-offs Considered:
  - Complexity vs. Performance: All proposed enhancements increase system complexity, either through additional processing or increased protocol requirements. However, these trade-offs are balanced by significant improvements in throughput and reliability.
  - Scalability: Each of these solutions has been evaluated for its ability to scale across different network environments. Machine learning-based approaches allow flexibility, making them suitable for networks of varying sizes and usage densities.
- Implementation Challenges:
  - Hardware Limitations: Older devices may struggle with the increased computational demands introduced by these enhancements. To mitigate this, lightweight models and efficient algorithms must be used.
  - **Interoperability:** Ensuring that devices with and without these enhancements can operate on the same network without performance degradation is a key challenge that must be addressed during implementation.

[1] Bianchi, G. (2000). "Performance Analysis of the IEEE 802.11 Distributed Coordination Function." *IEEE Journal on Selected Areas in Communications*, 18(3), 535-547.

[2] Mishra, A., Banerjee, S., & Arbaugh, W. A. (2006). "Weighted coloring based channel assignment for WLANs." *IEEE/ACM Transactions on Networking*, 14(6), 1259-1272.

[3] Lin, S. (2004). Error Control Coding. *Pearson Prentice Hall google schola*, 2, 17-22.