Aditya dixit -2022030

Pranjal Bharti-2021090

Kanav meena-2023266

TCP SACK Parser and Performance Analysis

1. Overview

The objective of this project is to implement a C function to parse the TCP options header, with a specific focus on identifying and printing TCP Selective Acknowledgement (SACK) ranges. This implementation is then used in a practical experiment to analyze the performance difference between file transfers with and without SACK enabled over a simulated network experiencing 10% packet loss. The analysis aims to quantify the benefits of SACK in unreliable network conditions.

2. Implementation of `print_tcp_options`

The core of the implementation lies within the `print_tcp_options` function, which is designed to parse the variable-length TCP options header.

Algorithm Design

The function iterates through the options header, which is a sequence of Type-Length-Value (TLV) encoded fields. The logic is guided by the `kind` (type) of each option.

1. **Initialization**: A counter `i` is initialized to 0 to serve as an index into the `options` character array.
2. **Parsing Loop**: A `while` loop continues as long as the index `i` is within the bounds of the header's total length `len`.
3. **Kind Evaluation**: Inside the loop, the byte at the current index is read as the `kind`.
    - **Kind 0 (End of List)**: Terminates the loop.
    - **Kind 1 (No-Operation)**: A padding byte; the index `i` is incremented by 1.

    - **Other Kinds (`>1`)**: The subsequent byte at `i + 1` is read as the option's total `length`.
4. **SACK Option Handling (Kind 5)**:
    - When `kind` is 5, the number of SACK ranges is calculated using `(length - 2) / 8`.
    - A `for` loop iterates through each range, converting the 4-byte start and end sequence numbers from Network Byte Order to Host Byte Order using `ntohl()`.
    - The formatted ranges are printed to standard output.
5. **Advancing the Index**: For any option of `kind > 1`, the index `i` is advanced by the value of its `length` to move to the next option.

This design ensures a robust traversal of the options header while specifically targeting the SACK data for extraction.

3. Experimental Setup

The experiment was conducted using two machines to simulate a real-world file transfer. A 10% packet loss rule was introduced on the receiver (Machine A) to create a network environment where SACK would be beneficial. A 1GB file was then transferred from the sender (Machine B) to the receiver.

4. Results and Analysis.

Question 1: Transfer Time with SACK Enabled

How much time does it take to transfer a 1 GB file with TCP SACK enabled, and you drop 10% of the packets at the receiver?

- **Time Taken**: **181.93s**

This time represents the baseline performance of TCP in a lossy network when it can intelligently retransmit only the specific segments that were lost.

Question 2: Transfer Time with SACK Disabled

How much time does it take to transfer a 1 GB file with TCP SACK disabled, and you drop 10% of the packets at the receiver?

- **Time Taken**: `241.98s`

Analysis: The transfer time is significantly longer without SACK. Without SACK, TCP must retransmit the entire window of data starting from the first lost packet, leading to redundant data transfer and much slower recovery from packet loss.

Question 3: Total SACK ACKs Received

What is the total number of TCP SACK ACKS received during the transfer of the file, with SACK enabled?

- **Total SACK ACKs**: **1316**

This number indicates how frequently the receiver detected and reported gaps in the received data stream due to the 10% packet loss.

Question 4: SACK Range Counts

Out of all the TCP SACK received, how many times were the numbers of ranges 1, 2, and 3?

- **Number of SACKs with 1 Range**: `1316`
- **Number of SACKs with 2 Ranges**: `0`
- **Number of SACKs with 3 Ranges**: `0`

Analysis: This data shows the degree of packet loss fragmentation. A high number of single-range SACKs indicates isolated packet drops, whereas a significant number of 2 or 3-range SACKs suggests that packet loss was more scattered.

Question 5: Other TCP Option Kinds Encountered

What are the other kinds you have encountered apart from 0, 1, and 5? Briefly describe their purpose.

During the packet capture, the following additional TCP option kinds were observed:

- **Kind 2 (Maximum Segment Size - MSS)**: Negotiates the largest segment size between hosts during the initial handshake to prevent IP fragmentation.
- **Kind 3 (Window Scale)**: Allows the TCP receive window to be scaled beyond its 65,535-byte limit, which is essential for high-speed networks.
- **Kind 8 (Timestamps)**: Used for more accurate Round-Trip Time calculation and to protect against wrapped sequence numbers (PAWS).

5. Conclusion

The experiment successfully demonstrated the critical role of the TCP SACK mechanism in maintaining performance on unreliable networks. The performance data clearly shows that disabling SACK in a lossy environment leads to a dramatic increase in file transfer time, confirming that SACK's ability to enable precise retransmissions is a highly effective optimization for the TCP protocol.