Flow Control Protocols - Comprehensive Sentence-wise Study Notes

Introduction and Context

Flow Control Protocols are fundamental mechanisms used specifically in the Data Link Layer of Computer Networks architecture. The Data Link Layer is the second layer in the OSI model and is responsible for reliable data transfer between adjacent network nodes. There are exactly three main Flow Control Protocols that we use in computer networking: Stop & Wait Protocol, Go Back N Protocol, and Selective Repeat Protocol. These three protocols are designed to manage and control the flow of data frames between a sender and receiver in a network communication system.

The primary purpose of these protocols is to ensure that the sender does not overwhelm the receiver with data frames faster than the receiver can process them. Flow control is essential because different devices may have different processing speeds and buffer capacities. These protocols help in managing the flow of data between sender and receiver to prevent data loss and ensure reliable communication.

Numerical questions based on these protocols have been asked many times in various competitive exams including GATE, NET, and other computer science entrance examinations. Theoretical questions about window sizes, efficiency calculations, and protocol characteristics also have been asked many times on these topics in competitive examinations. Before going to any competitive exam, you should read these comprehensive points once so that you can get a lot of help during the examination and score better marks.

Stop & Wait Protocol - Detailed Analysis

Basic Working Mechanism and Principle

Stop & Wait protocol operates on a very simple and straightforward working principle that forms the foundation of flow control. The protocol works in the following manner: it transmits exactly one frame at a time from sender to receiver, then it waits for the acknowledgement of that specific frame from the receiver. The sender will not proceed to send the next frame until the acknowledgement for the current frame doesn't return from the receiver. This fundamental rule means that only 1 frame is transmitted at any given time in the entire communication process.

The complete cycle works as follows: after sending one frame, the sender enters a waiting state for the acknowledgement of that frame, and only when the acknowledgement is successfully received, then the sender proceeds to send the second frame. This process continues sequentially for all frames that need to

be transmitted. The sender maintains a strict one-frame-at-a-time policy throughout the entire data transmission process.

Efficiency Problems and Performance Issues

Due to this mandatory waiting mechanism, the sender ends up waiting for a lot of time during the data transmission process. This waiting time occurs because the sender must wait for the round trip time (time for frame to reach receiver + time for acknowledgement to return) before sending the next frame. Because of this extensive waiting time, the utilization of available bandwidth becomes very poor and inefficient.

The bandwidth we are utilizing in Stop & Wait protocol becomes very poor compared to the total available bandwidth capacity. If we talk about the overall efficiency of this protocol, the efficiency is very poor in comparison to other flow control protocols. This poor efficiency is the major drawback of the Stop & Wait protocol. If you encounter a question anywhere in competitive exams asking which of these three protocols has the least efficiency, by default the correct answer will always be Stop & Wait protocol. This is because only 1 packet can be transmitted at a time, leaving the communication channel idle for most of the time during the round trip delay period.

Window Size Concept and Explanation

The concept of window size is crucial in understanding flow control protocols. In Stop & Wait protocol, the sender window size is exactly 1. Similarly, the receiver window size is also exactly 1. Since we are saying that we can send only 1 packet at a time due to the protocol's fundamental design, obviously this window size makes logical sense.

When only 1 packet can be sent at a time by the protocol design, then it would be of no practical sense making the window size more than 1. The window size represents what is our transmission capacity - at any given time how many packets can the sender's window accommodate and send simultaneously? When I am sending only 1 packet or 1 frame at a time due to protocol constraints, then the logical window size will be exactly 1. The size of receiver window is also 1 because the receiver can only process and acknowledge one frame at a time in this protocol.

Important Examination Questions

This specific question about window sizes has been asked many times in theory-based competitive examinations: what is the window size of sender and what is the window size of receiver in Stop & Wait protocol? So you must remember that in both cases, the answer is 1 for sender and 1 for receiver. Why is the answer 1 for both sender and receiver? The reason is because we are transmitting only 1 frame at a time according to the fundamental principle of Stop & Wait protocol.

Efficiency Formula and Mathematical Analysis

Efficiency calculation is a very important formula that you must remember for numerical problems. The efficiency of Stop & Wait protocol is mathematically expressed as 1/(1+2x) where the variable x equals propagation delay divided by transmission time. This formula is derived from fundamental principles of data transmission.

What does efficiency actually represent in the context of data transmission? Efficiency is defined as the transmission time divided by the sum of transmission time plus round trip time. This mathematical expression means our useful time divided by total time spent in the communication process. So the question becomes: how much time are we making useful out of the total time available? How much transmission time is actually used in sending 1 frame, that duration represents our useful time in the entire communication process. How much is the total time consumed? The total time is transmission time plus round trip time.

Understanding Round Trip Time Components

Round trip time is defined as what exactly? Round trip time equals 2 multiplied by Propagation delay. This is because the data frame must travel from sender to receiver (one propagation delay) and then the acknowledgement must travel back from receiver to sender (another propagation delay). So whether you write 2 multiplied by Propagation delay in your formula or simply write RTT (Round Trip Time), both representations mean exactly the same thing mathematically.

When you take TT (Transmission Time) as a common factor from both numerator and denominator, then this efficiency formula becomes the standard form that you have to remember: 1/(1+2x), where x is defined as TP/TT (Propagation delay divided by Transmission time). This ratio x is also known as the bandwidth-delay product parameter in networking terminology.

Fundamental Reason for Poor Efficiency

What is the fundamental reason behind the poor efficiency of Stop & Wait protocol? The core issue is that we are only sending 1 frame during the entire total time period of communication cycle. During the round trip time, the communication channel remains completely idle and unused. Because of this significant idle time, the efficiency becomes very poor in the case of Stop & Wait protocol. The channel utilization is minimal because most of the time the channel is waiting rather than actively transmitting data.

Retransmission Mechanism and Scenarios

When do we actually use the retransmission mechanism in Stop & Wait protocol? Let's consider a practical scenario: suppose we sent a packet or a frame from sender to receiver and unfortunately it gets lost somewhere on the transmission path. If that specific frame gets lost on the way during transmission, or if the acknowledgement gets lost while returning from receiver to sender, or in case the packet gets

corrupted due to transmission errors, whatever the specific case may be, if the acknowledgement does not come back to the sender within the expected time, in that problematic case the sender will automatically retransmit the same packet again.

What is the fundamental reason behind this retransmission mechanism? The acknowledgement does not come back to the sender within the timeout period. The sender has no way of knowing whether the frame reached successfully or failed without receiving the acknowledgement.

Real World Analogy for Understanding

Like when we use PayTM, Google Pay, or any digital payment application for financial transactions, until the acknowledgement (success message) does not come back to us, we will always feel that our transaction is not complete and may have failed. We remain uncertain about the transaction status until we receive confirmation. Similarly, in Stop & Wait protocol, until the acknowledgement does not come back to the sender, the sender will wait for some predetermined time period. That specific time period we call the Time-out timer or timeout duration.

So the sender will patiently wait until the Time-out timer expires, and after that timeout period elapses, it will retransmit the same frame. This timeout mechanism ensures that lost frames are eventually retransmitted and communication continues reliably.

Retransmission Count and Analysis

How many packets will the sender retransmit when a timeout occurs? Exactly 1 packet will be retransmitted. Why only 1 packet and not more? Because according to Stop & Wait protocol design, only 1 packet was sent initially from the sender. So 1 packet was sent first, and if it didn't reach the receiver successfully then the sender will again retransmit the same identical packet. So there is only 1 packet that needs retransmission, which is relatively very less compared to other protocols in case of Stop & Wait.

This low retransmission count is actually an advantage of Stop & Wait protocol, even though its efficiency is poor. When errors occur, only minimal retransmission is required, which reduces network overhead in error scenarios.

Implementation Complexity Analysis

If we talk about smaller implementation details in these protocols, if we say this Stop & Wait algorithm is to be implemented in actual programming code, then it is obvious that it is very easy to implement programmatically. There is no complex searching algorithm or sorting algorithm used in this protocol implementation. It is a very straightforward and easy method to code and implement in any programming language.

The simplicity comes from the fact that the sender only needs to maintain state for one frame at a time, and the receiver only needs to process one frame at a time. There are no complex data structures or algorithms required for implementation.

Sequence Numbers Required for Operation

The last important point in Stop & Wait protocol is about sequence numbers - they can ask you in examinations what is the total number of sequence numbers needed for proper operation. Let's say if I have sender window size equal to 1, and receiver window size equal to 1, then how many different sequence numbers do I need to uniquely identify frames? So you must always remember the standard formula for calculating required sequence numbers. The formula is: sender window size plus receiver window size equals total sequence numbers required.

So when sender window size is 1, and receiver window size is 1, then the calculation becomes 1 + 1 = 2. Therefore, available sequence numbers means how many different sequence numbers do I need for proper protocol operation? I need exactly 2 sequence numbers here (typically 0 and 1). These two sequence numbers are sufficient to distinguish between the current frame and the previous frame, preventing confusion between old and new transmissions.

Go Back N Protocol - Comprehensive Analysis

Basic Introduction and Protocol Classification

Now we proceed to discuss the Go Back N protocol in detail. The Go Back N protocol comes under the category of sliding window protocols, which are more advanced than simple Stop & Wait. What does the letter N represent here in the protocol name? N specifically represents the window size, which is the number of frames that can be transmitted without waiting for acknowledgement. Here what fundamental change we made compared to Stop & Wait protocol? We say don't send just 1 frame or 1 packet at a time like Stop & Wait, instead we are sending an entire window full of frames simultaneously.

In that window we are sending multiple packets or multiple frames at the same time without waiting for individual acknowledgements. This represents a significant improvement over the Stop & Wait approach. The sender can transmit up to N frames before requiring any acknowledgement from the receiver.

Efficiency Improvement Over Stop & Wait

If we talk about the overall efficiency of this Go Back N protocol, then obviously it is much better than Stop & Wait protocol. This improvement occurs because we are trying to send more frames at the same time instead of waiting after each frame. So here we send multiple frames simultaneously instead of just sending one frame and then waiting. This parallel transmission approach significantly improves channel utilization and overall throughput of the communication system.

The improvement in efficiency comes from the fact that while acknowledgements are traveling back, the sender can continue transmitting new frames, thereby keeping the channel busy and productive.

Sender Window Size Formula and Derivation

The sender window size that I have written here follows a specific mathematical formula: 2 raised to power k minus 1, expressed as (2^k - 1). What does the variable k represent here in this mathematical expression? K is specifically the number of bits required to represent the sequence numbers in the protocol implementation.

This relationship between k and window size is crucial for understanding the protocol's limitations and capabilities. The formula ensures that there are enough sequence numbers to distinguish between different frames in the sliding window mechanism.

Understanding Bits Representation in Networking

Many times what happens is you are asked in examination papers that "window size is represented by 3 bits", or "window size is represented by 4 bits". What does this statement actually mean in practical terms? If the problem states that window size is represented by 3 bits, it absolutely does not mean that the actual window size is simply 3. This is a common misconception among students.

What it actually means is that we are using a total number of 3 bits to represent all possible sequence numbers, so what is the actual maximum window size possible? The answer is 2 raised to power 3 which equals 8 different sequence numbers. This calculation is fundamental to understanding the protocol's capacity.

Detailed Calculation Example

How do we mathematically arrive at the number 8? When the number of bits required to represent a sequence number is 3, then we can ask: how many different numbers can be represented using 3 bits? The range goes from 000 to 111 in binary representation, which means from 0 to 7 in decimal representation. So from 0 to 7, how many different numbers are there in total? There are exactly 8 different numbers (0, 1, 2, 3, 4, 5, 6, 7).

So if you are given the number of bits in any problem, then you must remember to calculate 2 raised to power of that number of bits. This gives you the total number of available sequence numbers in the system.

Window Size Calculation Process

Now you can determine what is the total available window size? 8 represents the total sequence numbers available for use. How many available sequence numbers we have here for the entire system? We have 8 different sequence numbers. But what is the actual usable sender window size according to Go Back N

protocol? The sender window size is 2 raised to power k minus 1. This means whatever your total available sequence numbers are, you must reduce 1 from that total, and the result becomes your actual usable sender window size.

The reason for subtracting 1 is to avoid confusion between old and new frames when sequence numbers wrap around. This ensures reliable operation of the sliding window mechanism.

Alternative Mathematical Representation

So whether you write this formula as 2 raised to power k minus 1, or you first calculate the value of 2 raised to power k and call that result N (where N represents the total window size), then you can write the sender window size as N-1 also. You can write it as N-1 or write it as 2 raised to power k minus 1, that choice is your personal preference and both are mathematically equivalent.

But I am specifically writing 2 raised to power k so that you don't get confused during examinations. Many times instead of giving the window size directly as simple numbers like 8, 16, or 32, examination questions give the number of bits used for representation. In case of number of bits being given, you have to do nothing complicated, just calculate 2 raised to power k, and the answer you get is N (total sequence numbers). What is N exactly? N represents the total available sequence numbers. What will be the actual usable window size in Go Back N? The formula is 2 raised to power k minus 1, or alternatively N-1. You can use either of the two representations.

Receiver Window Size Analysis

So what is the receiver window size in Go Back N protocol? The receiver window size is exactly 1. This is a very interesting and important point that creates a significant contrast with the sender. At any given time, if I take the value of k as 3 for example, so what is the sender window size in this case? The sender window size becomes 7. Why does it become 7? Because 2 raised to power 3 equals 8, and 8 minus 1 equals 7. This means the sender is capable of sending 7 frames at the same time without waiting for acknowledgement.

Let the sender send 7 frames simultaneously, but the receiver will receive and process only 1 frame at a time due to its window size constraint. This asymmetry between sender and receiver window sizes is a key characteristic of Go Back N protocol.

Out of Order Packets Acceptance

So many times a critical question is asked in competitive examinations: can out of order packets be accepted in Go back N protocol? The definitive answer is absolutely No. Packets will always be accepted strictly in order only by the receiver. The fundamental reason for this strict ordering requirement is that the receiver window size is limited to exactly 1.

This means that even if multiple frames arrive at the receiver, it can only accept the next expected frame in sequence. Any frame that arrives out of order will be immediately discarded, regardless of whether it is a valid frame or not. This is a significant limitation of the Go Back N protocol.

Efficiency Formula and Calculation

Next if we talk about the efficiency calculation for Go Back N, the basic efficiency formula structure is similar but modified. The normal efficiency formula was 1/(1+2x) which was used in our Stop & Wait case. But in Go Back N we are not sending just 1 packet at a time like Stop & Wait. In this Go Back N protocol, at any given time we send 2 raised to power k minus 1 packets simultaneously, which represents the sender window size.

So whatever sender window size you calculate, let's say in our previous case it was 7, so the efficiency becomes 7 multiplied by the basic efficiency formula. The formula becomes: (sender window size) × 1/(1+2x). Put the calculated window size value in this formula, and again what does the variable x represent here? x represents propagation delay (or propagation time) divided by transmission time.

Numerical Problem Solving

So if you are given these specific values in a numerical problem during examination, you simply substitute them in this efficiency formula and your final answer will easily come out. This makes numerical problems on Go Back N quite straightforward once you understand the formula structure.

Cumulative Acknowledgement System

Here in Go Back N we use a sophisticated system called Cumulative acknowledgement system. Cumulative acknowledgement means let's say I send a sequence of packets 1, 2, 3, and 4 from sender to receiver. Then the acknowledgement that comes back is always for the next expected frame number. This point you must remember very clearly that whenever, let's say we send frame numbers 1, 2, and 3 from sender, so when these frames 1, 2, and 3 are successfully received by the receiver, then what acknowledgement number will the receiver send back? It will send acknowledgement for the next expected frame number.

Next expected frame concept means I have successfully received frames 1, 2, and 3, now I want frame number 4 as the next one in sequence. We always send the acknowledgement number for the next frame we expect to receive. Here we use cumulative acknowledgement for efficiency. Its major advantage is that the total traffic in the network will be significantly reduced compared to individual acknowledgements.

Real World Example of Cumulative Acknowledgement

Like for example, consider a practical scenario: if I am sending 100 packets from Chandigarh to Delhi every day, and at the same location I need to send 100 packets, then what may happen in different

approaches? One approach is I send 1 packet and I get an acknowledgement from Delhi, then I send the second packet and get acknowledgement, then third packet and get acknowledgement, and so on. So what can be a much better and more efficient situation?

I send all 100 packets at the same time in a batch, or send 100 packets one at a time and then receive just 1 cumulative acknowledgement that confirms all packets are successfully received. What will happen because of this cumulative approach is that the total traffic in the network will reduce dramatically. Instead of 200 messages (100 data + 100 individual ACKs), we only need 101 messages (100 data + 1 cumulative ACK).

Retransmission Problem Analysis

Next, let's analyze the retransmission behavior. Retransmission will obviously be very high in Go Back N protocol compared to Stop & Wait. What is the fundamental reason behind this high retransmission? If in case of Go Back N protocol, let's say I sent a sequence of 1, 2, 3, 4 packets, meaning I sent 4 frames simultaneously from sender. The receiver says I will accept only 1 frame at a time due to my window size constraint. Whether frames 1, 2, 3, or 4 arrives first at the receiver, I will accept frame 1 first because that's the next expected frame.

Detailed Retransmission Scenario

Let's say frame 1 gets lost somewhere on the transmission path. The receiver says I want frame 1 as the next expected frame. Frame 2 is waiting at the receiver buffer. Frame 2 says accept me, but receiver says no, I need frame 1 first. Frame 3 says accept me, but receiver says no, I still need frame 1 first. Frame 4 says accept me, but receiver says no, I still need frame 1 first. Why does receiver reject all other frames? Because it strictly wants frame 1 first due to the in-order delivery requirement.

Entire Window Retransmission Consequence

So what will happen in this problematic case? Frames 2, 3, and 4 which had successfully reached the receiver will be rejected and discarded by the receiver. So again the entire window containing all frames will need to be sent again from the sender. This means all the packets I sent before must be retransmitted, even those that were successfully delivered. So how many maximum packets might need retransmission? 7 packets in our example. How did we determine this number? Using the formula 2 raised to power k minus 1.

So maximum 2 raised to power k minus 1 packets need to be retransmitted, which equals the complete size of the sender window. The entire window may need to be sent again in the worst case scenario. So retransmission overhead is highest in case of Go Back N protocol among all flow control protocols. This is the major disadvantage of Go Back N.

Selective Repeat Protocol - Complete Analysis

Basic Introduction and Protocol Philosophy

Then if we talk about Selective Repeat protocol, then in Selective Repeat also we send multiple frames simultaneously like Go Back N. Selective Repeat is actually a mixed version that combines the best features of Stop & Wait and Go Back N protocols. It says send multiple frames at a time like Go Back N for efficiency, but handle errors more intelligently like an improved version of Stop & Wait.

The philosophy behind Selective Repeat is to overcome the major disadvantage of Go Back N (excessive retransmission) while maintaining the efficiency benefits of sending multiple frames simultaneously. It represents the most sophisticated approach among the three flow control protocols.

Window Sizes and Their Significance

But there is a crucial difference: sender window size and receiver window size in Selective Repeat protocol are exactly the same. The window size of both sender and receiver is equal, which is a major departure from Go Back N. What is the mathematical formula for window size? Sender window size is 2 raised to power (k-1) and receiver window size is also 2 raised to power (k-1).

Here you must be very careful about the mathematical formula, because in Go Back N the minus 1 was outside the power (2^k - 1), but here in Selective Repeat the minus 1 is inside the power in the exponent (2^(k-1)). So here again, what does the variable k represent? K is the number of bits used to represent sequence numbers in the protocol implementation.

Detailed Calculation Example

So if I take the value of k as 3 for calculation purposes, then how many total sequence numbers will be available? The available sequence numbers will be 8 (because 2^3 = 8). When total sequence numbers become 8, then here what will be the sender window size according to our formula? If you put the value 3 in the formula 2^(k-1), then what result will it give? 2^(3-1) = 2^2 = 4. And receiver window size will also become exactly 4 using the same calculation.

This equal window size for both sender and receiver is the key feature that enables Selective Repeat's advanced capabilities. Both sender and receiver can handle up to 4 frames simultaneously in this example.

Major Advantage - Out of Order Acceptance

From this equal window size what major advantage do we get? From this the most significant advantage we get is that it can accept out of order packets also. What is the fundamental reason behind this capability? I sent 4 frames from sender, and receiver can also accommodate and process 4 frames

simultaneously due to its window size. So if in case the first frame doesn't come, but second frame reaches first or third frame reaches first, the receiver will accept it and store it temporarily.

Why can receiver accept out of order frames? Because receiver has sufficient empty buffer space. It has empty space to accept all 4 packets simultaneously and can store them until the missing frames arrive. Let's say packet number 2 comes first before packet 1, receiver will say okay, I will put frame 2 here in buffer position 2. When frame 1 comes later, I will put it in the correct buffer position 1. But this was the fundamental problem in Go back N protocol.

Comparison with Go Back N Limitation

In Go Back N protocol, receiver has only 1 buffer space for storing frames. So no matter how many packets arrive at the receiver, the specific one it needs first (next expected frame), that particular frame will have to be received first before any other frame can be accepted. So here in Selective Repeat, out of order packets can be accepted and stored intelligently.

Low Retransmission Advantage

And that's why our retransmission overhead will be very low in Selective Repeat protocol. Or you can say it will be equal to just 1 packet typically. What is the fundamental reason behind this low retransmission? I will only need to retransmit that specific packet which gets lost during transmission. Only that particular packet needs to be retransmitted, not the entire window.

This is a massive improvement over Go Back N, where the entire window needed retransmission if any single frame was lost. In Selective Repeat, if frame 2 is lost but frames 1, 3, and 4 are received successfully, only frame 2 needs to be retransmitted.

Retransmission Comparison

So retransmission behavior of Selective Repeat and Stop & Wait is similar in terms of quantity - both protocols retransmit only 1 packet when errors occur. However, Selective Repeat achieves this low retransmission while maintaining the efficiency of multiple frame transmission.

Efficiency Formula and Performance

Next, let's discuss efficiency calculation. What is efficiency here in Selective Repeat? The efficiency formula is 2 raised to power (k-1), which represents the sender window size, multiplied by the standard efficiency formula. So the number of packets we send, the number of frames we send at a time, multiplied by the same basic formula 1/(1+2x), where x is propagation delay divided by transmission time.

The efficiency is calculated as: (sender window size) × 1/(1+2x). Since we can send multiple frames simultaneously, the efficiency is much better than Stop & Wait, and comparable to or better than Go Back N.

Performance Summary

So in case of Selective Repeat protocol, its efficiency is good (due to multiple frame transmission) and retransmissions are minimal (due to selective retransmission). This combination makes Selective Repeat the most desirable protocol from a performance perspective.

Acknowledgement Types Supported

And Selective Repeat uses multiple types of acknowledgement systems: both cumulative and independent acknowledgement. Cumulative acknowledgement means packet number 1, 2, 3, 4 came successfully, so next expected frame is let's say 5. So it will send acknowledgement number 5 to indicate "I have received everything up to 4, now send me 5".

And if receiver wants to send independent acknowledgement, let's say packet number 1, 2, 3 came successfully. If it wants to send different acknowledgement for each individual frame, it can send independent acknowledgement also (like ACK1, ACK2, ACK3 separately). This flexibility allows the protocol to adapt to different network conditions and requirements.

Negative Acknowledgement Feature

Along with this I will tell you one more advanced point, Selective Repeat can also send negative acknowledgement (NAK). Negative acknowledgement means what exactly? Let's say sender sent a packet, a frame and it successfully reaches the receiver physically. Receiver receives the frame and checks it for errors, and it detects there is some error in the received frame. Some 1 bit error or 2 bit error occurred during transmission due to noise or interference.

So what will receiver do in this error case? In this situation it will send a negative acknowledgement back to sender. What will be the major advantage of negative acknowledgement system? The sender will not have to wait for very long timeout period to detect the error.

Time Saving with Negative Acknowledgement

Otherwise how much time will sender wait in normal protocols? The sender will wait up to the complete time-out timer duration. It will wait up to time-out timer expiry, then it will realize that acknowledgement of packet has not yet come back. Then sender will assume the packet was lost and will send the packet again after timeout.

But what will happen in case of negative acknowledgement system? Receiver will inform the sender immediately and beforehand that there is a problem in your packet, please send the packet again immediately. So in that case, instead of waiting for the full timeout period, sender will retransmit the packet immediately upon receiving NAK. So from that we will save considerable amount of time and improve overall efficiency.

NAK Usage in Selective Repeat

So Selective Repeat actually follows and supports negative acknowledgement system as one of its advanced features. This makes error recovery much faster and more efficient compared to timeout-based error detection.

Implementation Complexity Analysis

Implementing Selective Repeat is the most difficult and complex in case of actual coding if we talk about programming implementation. Why is it considered the most difficult to implement? Because both searching and sorting algorithms are used extensively in this protocol implementation. We will have to search for the specific packet which was lost among all transmitted packets, and the same specific packet we have to identify and retransmit.

So the searching algorithms, etc. here become a little more complex and sophisticated sorting algorithms, etc. are required for maintaining the correct order of frames in buffers. That's why its practical implementation is quite complex compared to the other two protocols.

The complexity arises from managing multiple buffers, keeping track of which frames have been received, which are missing, and maintaining the correct sequence order for delivery to higher layers.

Implementation Comparison

Implementation of Stop & Wait is very simple due to its straightforward nature. Go Back N has moderate complexity. But the major evaluation point for protocol selection is related to window size calculations and efficiency performance rather than implementation complexity.

Protocol Selection Considerations

So this is what constitutes the major comparison between these three protocols. In practical network systems, the choice of protocol depends on factors like network reliability, bandwidth-delay product, processing capabilities of devices, and acceptable complexity levels.

Comprehensive Comparison and Final Summary

Efficiency Ranking and Analysis

Stop & Wait protocol has the poorest efficiency among all three protocols because it sends only 1 frame at a time and wastes most of the available bandwidth during waiting periods. The channel remains idle during the entire round trip time, leading to very poor utilization. Go Back N protocol has significantly better efficiency compared to Stop & Wait but still suffers from high retransmission overhead which can reduce effective throughput. Selective Repeat protocol has the best overall efficiency with minimal retransmission overhead, making it the most efficient protocol under most network conditions.

Retransmission Overhead Analysis

Stop & Wait has low retransmission overhead because only 1 packet is retransmitted when errors occur, but this comes at the cost of very poor efficiency. Go Back N has the highest retransmission overhead because entire window needs to be retransmitted if any frame is lost, which can significantly impact network performance in error-prone environments. Selective Repeat has the lowest retransmission overhead because only specifically lost packets need retransmission, making it most suitable for unreliable networks.

Implementation Complexity Summary

Stop & Wait is the easiest to implement requiring minimal programming logic and no complex data structures. Go Back N requires moderate implementation complexity with sliding window management but simpler than Selective Repeat. Selective Repeat requires the most complex implementation with sophisticated buffer management, searching algorithms, and sorting mechanisms for maintaining frame order.

Practical Application Scenarios

These protocols find applications in different scenarios based on their characteristics: Stop & Wait is suitable for simple systems where reliability is more important than efficiency. Go Back N is good for high-speed reliable networks where errors are rare. Selective Repeat is ideal for networks with moderate error rates where both efficiency and reliability are important.

These are the major fundamental differences you need to remember for competitive examinations and practical networking applications.