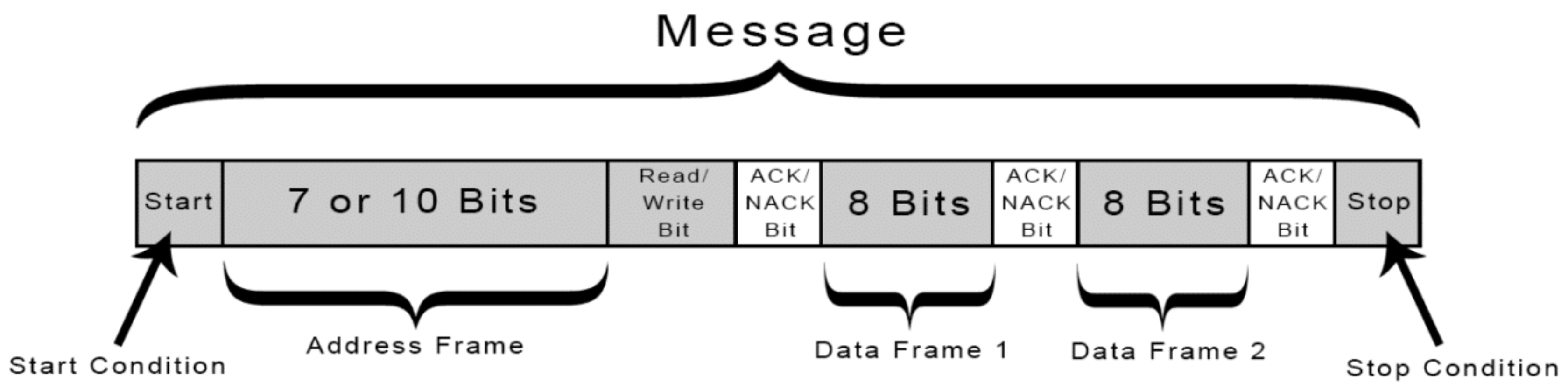
Motivation for I2C Packet Format - CubeRover

Function	Fault and Reg ID	Data Size	Data	CRC8
1 Byte	1 Byte	1 Byte	n Bytes	1 Byte

Standard for I2C Transfer



Implications of CubeRover Packet

- Because the data size bit is 1 byte – we will limit our cases to 255 size of bytes for the data n.
  - In total, we will have 259 Bytes transferred over I2C max for each data packet.
  - Checksum will be helpful for error detection, but not for error correction

Constraints of CubeRover Packet

With each I2C reply, MCU includes a fault bit

CubeRover has a limit of 1 sec for a timeout

- Will at max try 3 times to initiate the same command. If this process fails, Safe Mode is enabled, and the fault register is set

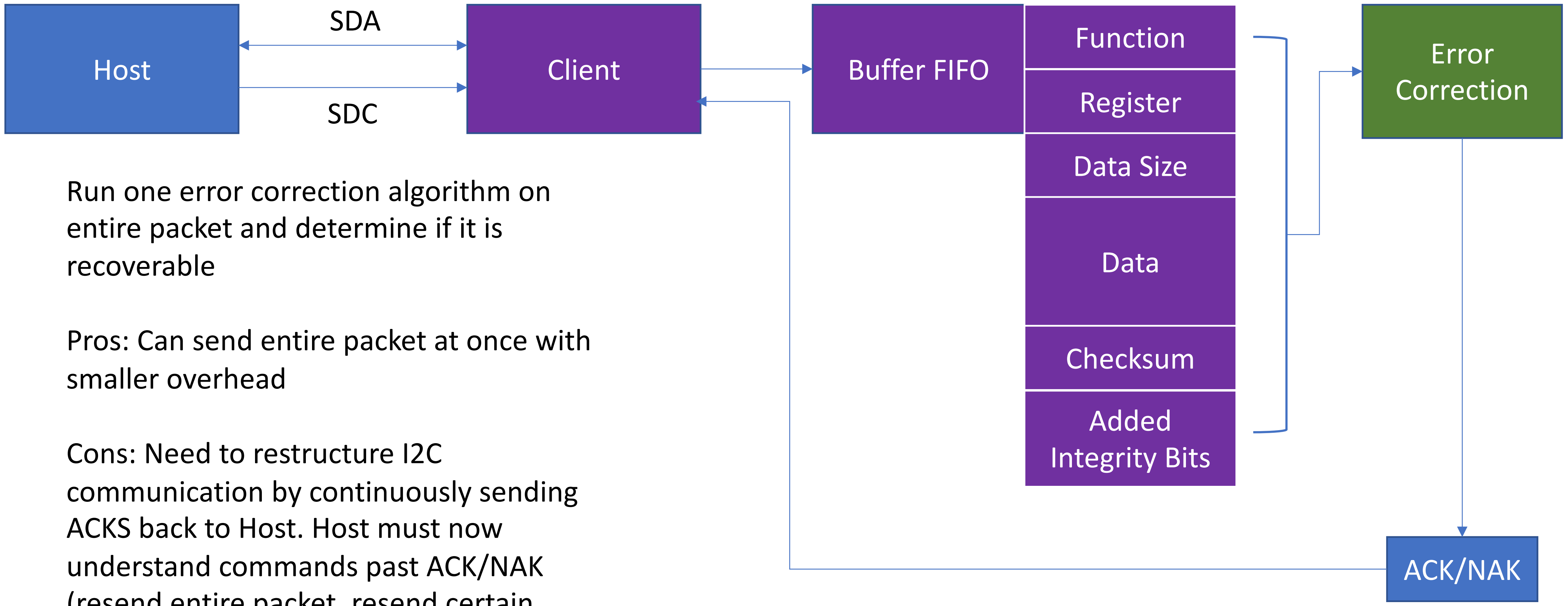
CubeRover limits the total number of retries in any case to 3 tries

- After that it will go into safe mode

Possible Approaches to Avoid Safe Mode

- We know that there is already a system in place in case of failure.
- Checksum is not a guaranteed mechanism and can only do error detection at max.
- Does not take into consideration that the checksum itself can be compromised by radiation.
- We must work within the timeframe specified to error correct the entire packet sent or ask for a retry.

Architecture of I2C Packet Transfer Approach 1

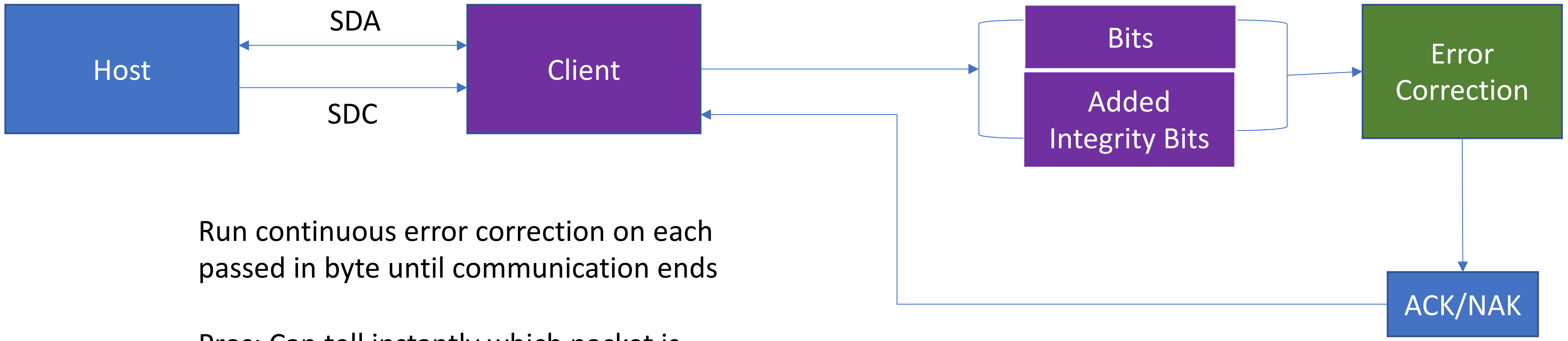


Run one error correction algorithm on entire packet and determine if it is recoverable

Pros: Can send entire packet at once with smaller overhead

Cons: Need to restructure I2C communication by continuously sending ACKS back to Host. Host must now understand commands past ACK/NAK (resend entire packet, resend certain packets, etc.)

Architecture of I2C Packet Transfer Approach 2



Run continuous error correction on each passed in byte until communication ends

Pros: Can tell instantly which packet is malformed/erased. Does not interfere with protocol.

Cons: High overhead because integrity bits might need to be added. Change the way the data is being sent by the hosts.