Software Design Document

Autonomous Intersection Management System

Version 1.0

October 12, 2022

Team: Protocol Pros

Members:

- Ashutosh Mishra
- Prakash Acharya
- Brendan Edgerley
- Julian Villarreal
- Sarah Ryan
- Amado Lazo
- David Schelanko

Table of Contents

Section 1: Overview of Proposed System

The main purpose of this project is to comprehend the possibilities of an intersection, and accordingly analyze, research, develop and execute a networking protocol for autonomous intersection management system. The networking protocol will involve transmission of signals and critical information between entities involved in the system and enable assistance to the agents for deciding their trajectory based on the road conditions.

We are proposing to build a wholesome system to simulate an intersection involving multiple agents such as vehicles, or agents with IoT capabilities, agents without IoT capabilities, AIMS system, smart city management system, etc. wherein the communication shall be operated based on our proposed system of custom protocols.

The proposed system shall operate under the internet layers, and protocols shall be made for functioning in application layer, transport layer and network layer, based on requirements of the system.

The Iterative Waterfall Model includes some modifications to the model to help improve the performance of the software development. The reason why we are using the Iterative Waterfall Model is because it seems more appropriate to the design of the project. It is straightforward to understand and apply the software we are designing.

Section 2: Summary of Requirements

1. Receive the video stream from intersections, **through the defined protocol**.
2. Identify agents from the video, categorize the identified agents and analyze their behavior.
3. Setup priority policy of agents prior to the intersection operation.
4. Generate the instructions to be provided to agents of the current intersection.
5. Ensure that priority policy is maintained while generating instruction.
6. Simulate the intersection behavior assuming the generated instructions to ensure safety.
7. Provide generated set of instructions corresponding to the instruction receptor(agent) to the message sender module.
8. Receive the message digest from the message sender module based on the encryption logic (HMAC, SHA-2, or Fully Homomorphic encryption)
9. Send the encrypted instructions to the corresponding agent, **through the defined protocol**.
10. Receive instruction reception confirmation from each agent.
11. Maintain consistent intersection-status check to ensure absence of accidents.
12. Maintain a periodic video reception confirmation to the video receptor module.
13. Maintain a responsive intersection status provision logic for sending intersection status to the higher authority (Smart City Management), as per requirement, **through the defined protocol**.

Section 3: Design Considerations and Parameters

The overall network is going to replicate the working of the internet and will consider the TCP/IP model, or the internet network layers rather than the OSI model, with major works on the application layer, transport layer and the network layer. The design of physical layer would transcend the scope of our project, however, fragmentation in the transport layer and formation of bits in the network layer would be discussed in the project.

1. **Application layer protocols**

   Application layer is the top layer of any network where the actual applications run and the data to be transmitted are generated. These data then get transferred to the lower level through application layer protocols. Applications are identified by the lower layers of the network models using ports. Thus, port number has to be mandatorily included in the application layer protocols. The application layer of our network has to take care of 4 independent functionalities. Thus, we are supposed to design four different application-level protocols. These 4 protocols would run over the transport layer than we will design subsequently.

   The functionalities to be considered in the application level are as follows:

   i. Video streaming
   ii. Reservation request (High-speed Message Transmission)
   iii. Instruction transmission (Low Latency Message Transmission)
   iv. Long message delivery

2. **Transport Layer protocols**

   The transport layer is responsible for defining the mode of transmission of data and the way this transmission is supposed to take place. Whether the communication requires a pre-setup of the connection, whether it requires a connection throughout the data transmission, whether the connection has to be acknowledged by the destination or not, and few other choices are required to be made in the transport layer. Fragmentation of data into packets takes place in the transport layer. Thus, the logic behind setting up the conversion of data into packets has to be decided in this layer. Connection-oriented or connection-less protocols reside in the transport layer which manipulate the connection between the source and destination of the communication. If there is a pre-setup connection between the two parties, it is termed as handshaking and this protocol paradigm is termed as connection-oriented protocol. Whereas, if the communication happens without a pre-setup of the connection between these two parties, then it is termed as connection-less protocols, where the data packets are simply released on open network, and they find their own way to the destination.

Our network would merely comprise of two transport layer protocols. They would be as follows:

i. Connection-less protocol (No handshaking): For receiving unicast reservation requests and sending intersection status to Smart City Management Systems.

ii. Connection-oriented protocol (Involves handshaking): For video stream and multicast instruction transmission to agents.

3. **Network Layer**

Network Layer is the layer residing above the physical layer in the network protocol where the packets are converted into frames and the data is made ready to be converted into bits, which can finally be transmitted over the physical layer wherein the frames will be converted to bits and get transmitted. Network layer encapsulates the entire metadata and the actual data along with the source and destination addresses into a single block of message, which can be treated as a frame and be transmitted across the physical layer.

Our network demands an efficient network layer protocol that can suffice good amount of data with optimal latency and optimal data size. Source and destination addresses have to be of enough length to accommodate a scalable number of systems with unique addresses.

A protocol similar to the IPv6 is to be designed in such a way that the upper transport layer protocols can function above this protocol, and the frames that this layer results can be converted readily into bits in the physical layer.

4. **Design of the video processing module**

   The video processing module of the AIMS is expected to have a clear and accurate OCR model inculcated in it, which can identify the agent and categorize the class of the agent. An agent is going to have few properties such as: AgentCategory, VehicleNumber, VehicleCategory, and so on. The OCR embedded in the video processing module of the AIMS has to be able to categorize the properties that the video should be able to identify.

5. **Design of security bottlenecks**

   The functionalities that define the security aspects of the AIMS system are as follows:

   i. HMAC authentication mechanism: Used for transmitting instructions from AIMS to agent

   ii. SHA-2 hashing mechanism: Used to hash the message being sent from AIMS to Smart City System.

6. **Design of priority policy of agents in the intersections**

   Priority policy of agents involved in the intersection is based on pre-setup of the AIMS Core Module wherein the priority of the agents are set, and the AIMS processes the instructions to be transmitted to the agents based on priorities of agents.

   For the current scenario, priority of agents are as follows: Agent without IoT capabilities, Emergency agents, Agents with IoT capabilities (non-emergency)

Section 4: Constraints

1. Majority of the agents in the intersection should be IoT capable and compatible with our application.

2. The vehicles should have adequate connection to transmit and receive data from our system.

3. The agents should have inbuilt methodology to decrypt the received encrypted message and properly view the instructions provided by the intersection.

4. AIMS should have pre-trained and accurate OCR (Optical Character Recognition) module which should have the ability to recognize the agents, based on their inherent properties like shape, color, license-plate (if vehicle), priority status (based on vehicle structure), etc.

5. AIMS should have adequate bandwidth and throughput to process and transmit all relevant data to the corresponding entity.

6. Ideally, IoT capable agents should strictly follow the instructions provided by our application.

Section 5: Architectural decisions

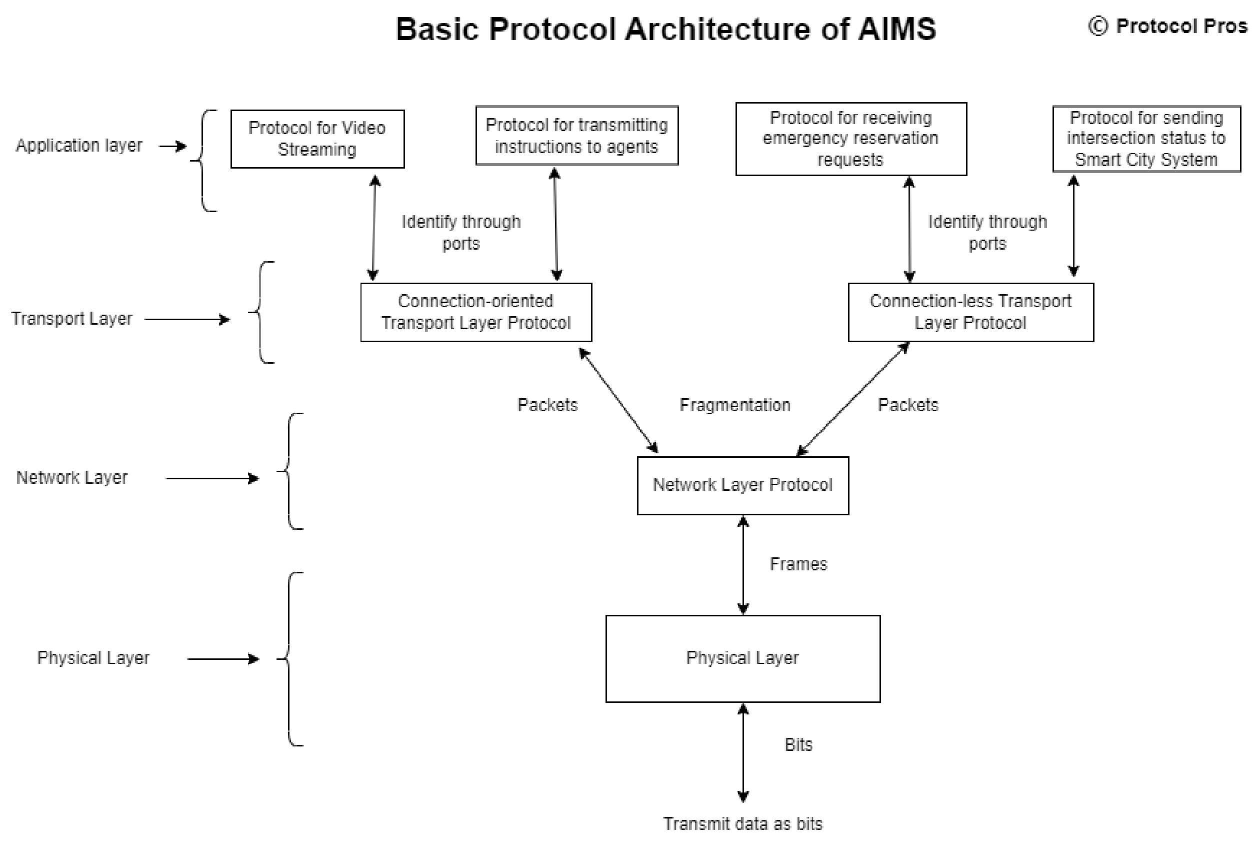

Basic Protocol Architecture of AIMS
© Protocol Pros
Application layer
Protocol for Video Streaming
Protocol for transmitting instructions to agents
Protocol for receiving emergency reservation requests
Protocol for sending intersection status to Smart City System
Identify through ports
Identify through ports
Transport Layer
Connection-oriented Transport Layer Protocol
Connection-less Transport Layer Protocol
Packets
Fragmentation
Packets
Network Layer
Network Layer Protocol
Frames
Physical Layer
Physical Layer
Bits
Transmit data as bits

Fig 1: Protocol Architecture of AIMS

As shown in Fig 1, the overall AIMS protocol operates in four layers, namely application, transport, network and physical layer. These layers are supposed to function independently and interact with each other based on the design of the network. The communication between these layers occurs through the definition of protocols that is instilled within each layer. Likewise, the application layer protocols function independently over the designated transport layer protocol, which is on top of the network layer protocol.

Section 6: Protocol Description

6.1. Application Layer Protocols

The proposed AIMS system contains four application layer protocols covering four different functionalities. These application layer protocols combinedly take care of all the use cases of the AIMS system and handle every communication that the AIMS has to perform.

6.1.1. Video streaming protocol

This protocol takes care of the video stream that will occur in every intersection. The AIMS has to identify the intersection status through a video inspection that concurrently occurs under every moment in order to identify what is happening in the intersection and which agent are passing through. Accordingly, this video stream is processed by the 'Video Processing Module', which identifies the agents and transmits the agent identity and behavior to the 'AIMS Core Processor'.

Thus, the major expectations that this protocol holds are a low-latency, reliable and clear streaming of video of the intersection. However, the interruption of video stream with clarity due to bad weather, bad internet is beyond the scope of this project. It is supposed to be connection-oriented due to reliability consideration in the stream.

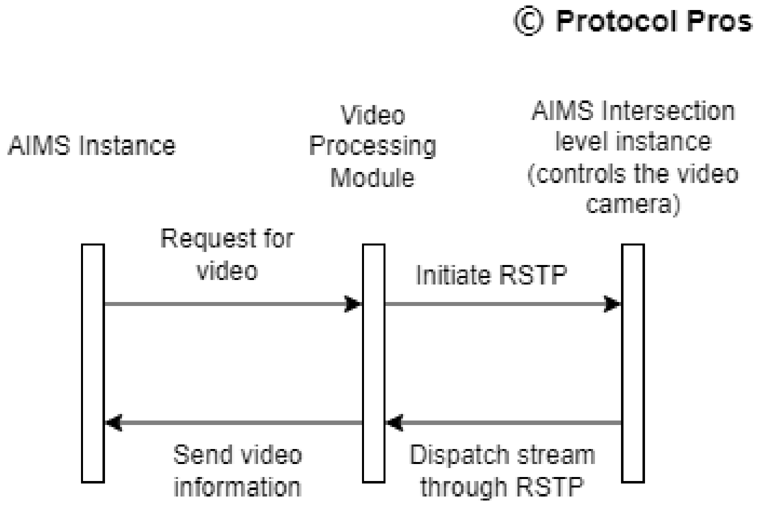
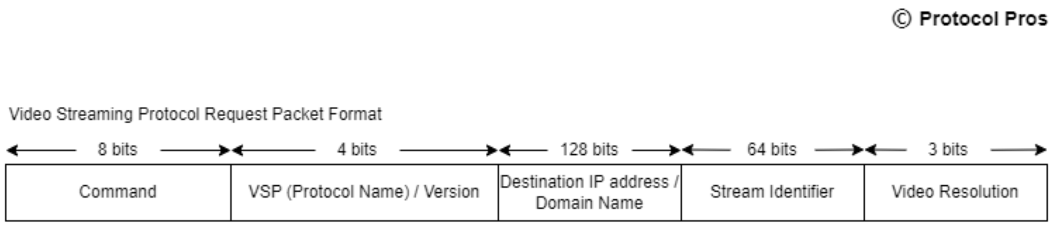
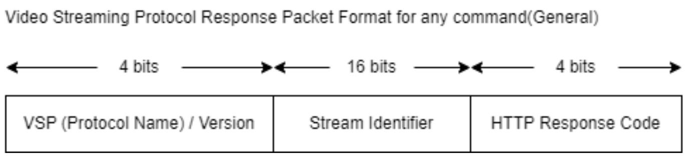


Fig: Basic working of the RSTP protocol


© Protocol Pros
Video Streaming Protocol Request Packet Format
8 bits
4 bits
128 bits
64 bits
3 bits
Command
VSP (Protocol Name) / Version
Destination IP address / Domain Name
Stream Identifier
Video Resolution


Video Streaming Protocol Response Packet Format for any command(General)
4 bits
16 bits
4 bits
VSP (Protocol Name) / Version
Stream Identifier
HTTP Response Code

Request Packet header content

i. **Command:** represents the command that the video module is required to perform. The possible commands are as follows:
   - **START:** start the video stream
   - **PAUSE:** pause the video stream
   - **RESUME:** resume the paused video stream
   - **CHANGERES:** change resolution of the video stream
   - **CHANGEVIEW:** change view of the stream
   - **STOP:** stop the stream

ii. **VSP Version:** version of the VSP Protocol (Video Streaming Protocol)

iii. **Destination IP address / Domain Name:** Domain name or IP address of the destination (AIMS, in our case)

iv. **Stream Identifier:** pointer to identify the current stream

v. **Video Resolution:** resolution of the current stream

Response Packet header content

i. **VSP Version:** version of the VSP Protocol (Video Streaming protocol)

ii. **Stream Identifier:** pointer to identify the current stream

iii. **HTTP Response Code:** represents the response code to the payload (HTTP Response Code format)

6.1.2. Protocol for transmitting instructions to agents

This protocol takes care of sending the processed instruction to each agent in an intersection in real-time, such that the instructions are received by the agents with minimum latency and would not cause any stall in the communication. This protocol is proposed to run on top of a connection-oriented transport layer protocol in order to maintain reliable instruction transmission and ensure the delivery of data to the agents.

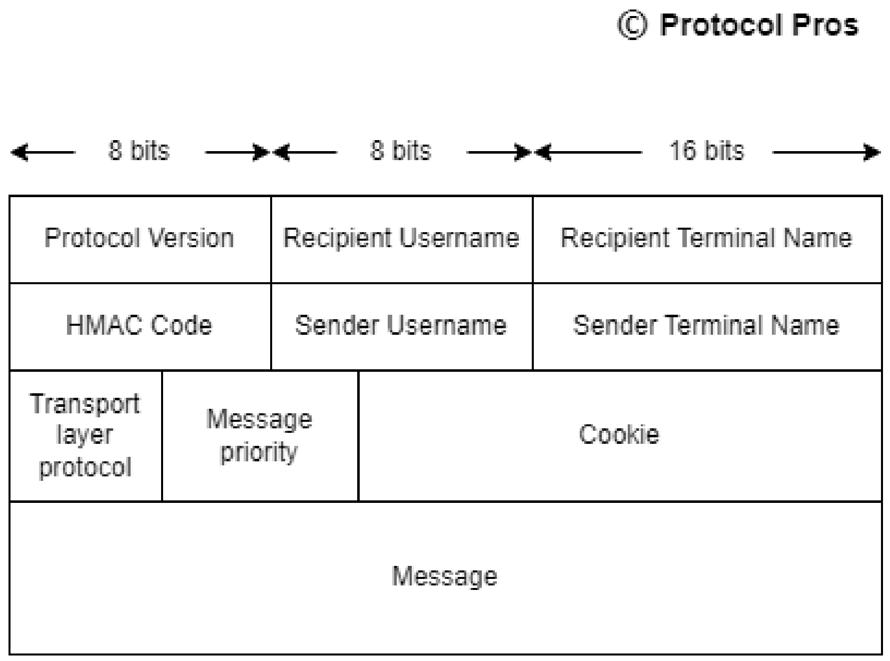


Fig: Simple Message Transmission Protocol

The Simple Message Transmission Protocol works with the following data items within.

i. **Protocol Version:** version of the SMTP (1.0 currently)

ii. **Recipient Username:** User name of the destination application

iii. **Recipient Terminal Name:** The terminal where the destination application is running currently. If left empty, it will get delivered to the center of the network, i.e. the AIMS.

iv. **HMAC Code:** The Hashed Message Authentication Code, if HMAC authentication mechanism is used to maintain the message integrity. Can be left empty if HMAC is not to be used.

v. **Sender Username:** Username of sender application

vi. **Sender Terminal Name:** The terminal/port of the sender application.

vii. **Transport layer protocol:** Since Simple Message Transmission Protocol can be used with both connection-oriented and connection-less transport layer protocols, we need to specify the choice of the current message transmission in this section.

viii. **Cookie:** Used to identify the messages which are already sent. They are expected to be used more in connection-less transport layer protocol. In our case, when the emergency agent requests a reservation to AIMS, the request can be raised multiple times just to ensure the reliability of packet delivery, thereby making an effective reservation request. This cookie enables the receiver to keep track of the messages that are already received.

ix. **Message:** The actual message to be sent. It may be in cipher-text format, may be encrypted as per requirement.

6.1.3. Protocol for transmitting emergency reservation request

There can be situations where any emergency vehicle approaches an intersection and would like to raise a reservation request to the approaching intersection. This has to be taken care as a separate use case. This protocol handles this use case. Since this communication is supposed to be quick and simple, this protocol runs on top of a connection-less transport layer protocol.

6.1.4. Protocol for updating intersection status to Smart City System

Higher authorities often request the AIMS to provide the status of a particular intersection at an instance. The AIMS is supposed to provide the instantaneous intersection status to the Smart City System. This communication between the AIMS and the Smart City System is taken care by this protocol. It runs over a connection-less transport layer protocol.

6.2. Transport Layer Protocols

The proposed AIMS system contains two transport layer protocols covering two different data transmission paradigms. These transport layer protocols support the application layer protocols of AIMS system and handle every communication that the AIMS has to perform. Fragmentation happens at this level, wherein the data to be transmitted is converted into packets and accordingly processed in the further layer.

6.2.1. Connection-oriented Transport Layer Protocol

This transport layer protocol works with a handshaking mechanism and establishes a connection between the two parties involved in the communication. Through this establishment of the connection, it becomes reliable, secured, and involves more complications. SYN/ACK and CON/ACK mechanisms can be inculcated in the connection-oriented protocols. We have two application layer protocols that require this connection-oriented transport layer protocol, namely Video Streaming Protocol and Instruction Sending Protocol.

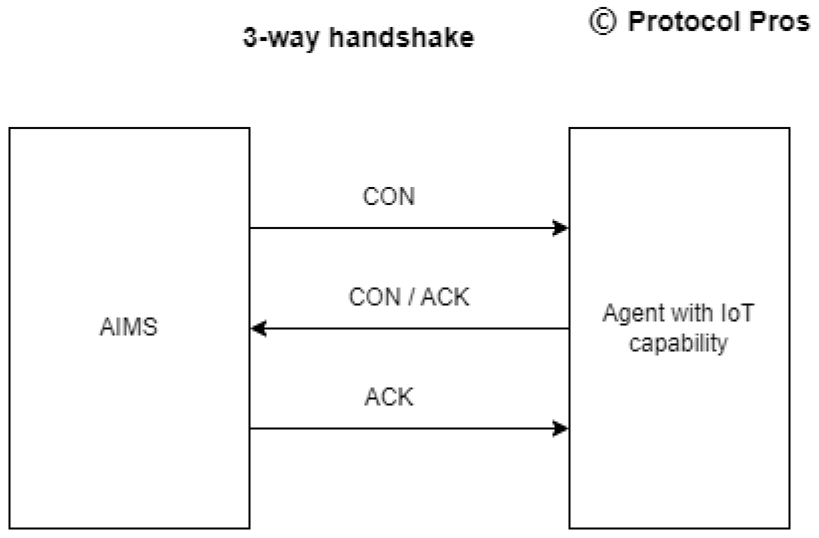


Fig: Three-way handshake for establishing connection for instruction transmission through connection-oriented protocol

Steps for maintaining the 3-way handshake for connection setup through transport layer are as follows.

Step 1: AIMS sends connection request to the agent. (CON Flag is set)

Step 2: Agent responses to the request and acknowledges that the connection is ready to be setup. (CON/ACK Flag is set)

Step 3: AIMS receives the acknowledgement from the agent and responds to the agent as an assertion of the connection setup. (ACK Flag is set)

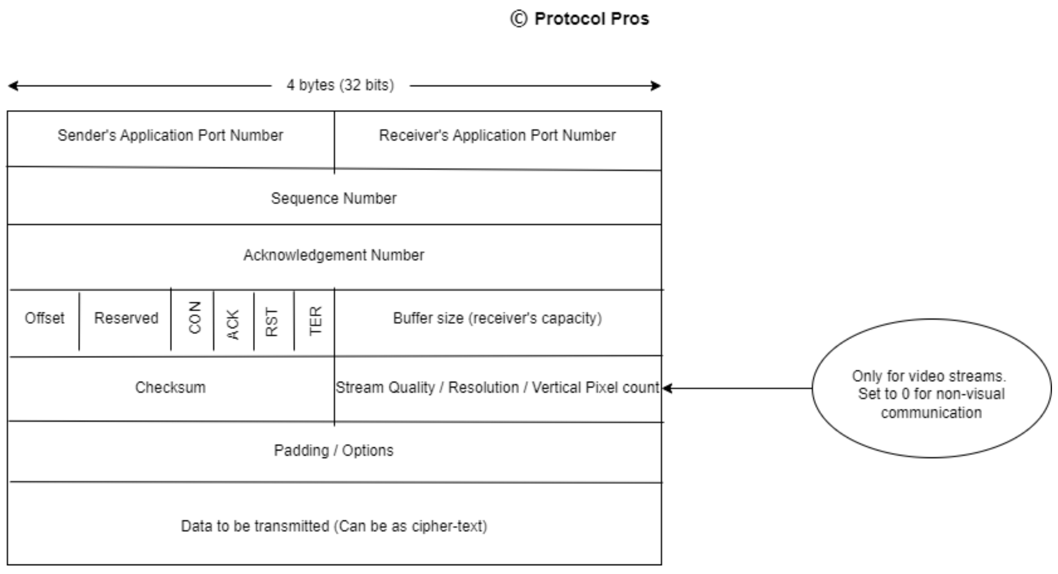

© Protocol Pros
4 bytes (32 bits)
Sender's Application Port Number
Receiver's Application Port Number
Sequence Number
Acknowledgement Number
Offset
Reserved
CON
ACK
RST
TER
Buffer size (receiver's capacity)
Checksum
Stream Quality / Resolution / Vertical Pixel count
Only for video streams. Set to 0 for non-visual communication
Padding / Options
Data to be transmitted (Can be as cipher-text)

Fig: Connection-oriented transport layer protocol

The Connection-oriented transport layer protocol header contains multiple fields that enable reliable and secured transmission of data.

i. **Sender's Application Port Number:** represents the port number of sender application

ii. **Receiver's Application Port Number:** represents the port number of receiver application.

iii. **Sequence Number:**

iv. **Acknowledgement Number:**

v. **Offset:**

vi. **Reserved bits:**

vii. **Flags:** represent bits each with particular indications.

- **CON** – connection flag (used while handshaking)
- **ACK** – acknowledgement flag (used while handshaking)
- **RST –** reset flag (used to start the connection over)
- **TER –** terminate flag (used to terminate the connection)

viii. **Buffer size:** maximum buffer size of the receiver. Used to prevent data overflow.

ix. **Checksum:** represents bits that indicate whether the data received is correct or not.

x. **Stream Quality:**

xi. **Padding:** used to arrange the bits in a proper format.

xii. **Data to be transmitted:** the actual data that comes from the application layer.

6.2.2. Connection-less Transport Layer Protocol

A connection-less transport layer protocol is required for the applications that maintain communications without a pre-establishment of connection. In our case, there are two applications that require this, viz. the emergency reservation request capability and the use case where AIMS has to update intersection status to Smart City Systems.

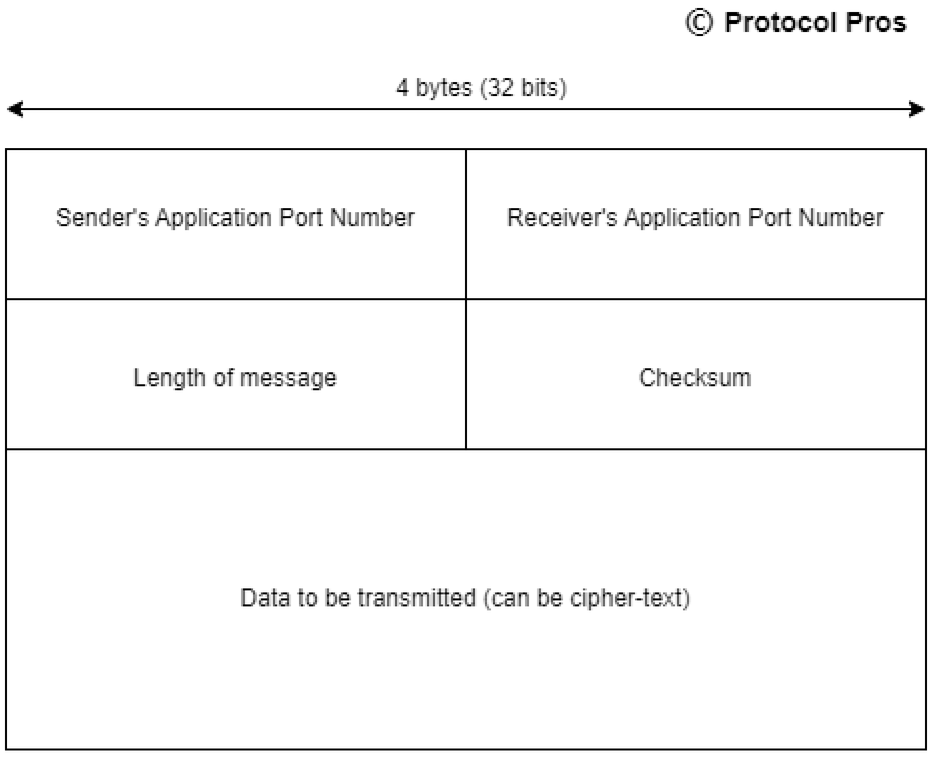

© Protocol Pros
4 bytes (32 bits)
Sender's Application Port Number
Receiver's Application Port Number
Length of message
Checksum
Data to be transmitted (can be cipher-text)

Fig: Connection-less transport layer protocol

i. **Sender's Application Port Number**: represents the port number of the sender application.

ii. **Receiver's Application Port Number**: represents the port number of the receiver application.

iii. **Length of message**: represents the size of the message to be sent

iv. **Checksum**: ensures whether the delivered message is correct or not. It maintains integrity of message throughout the connection.

v. **Data to be transmitted**: Data that is received from the application layer.

6.3. Network Layer Protocol

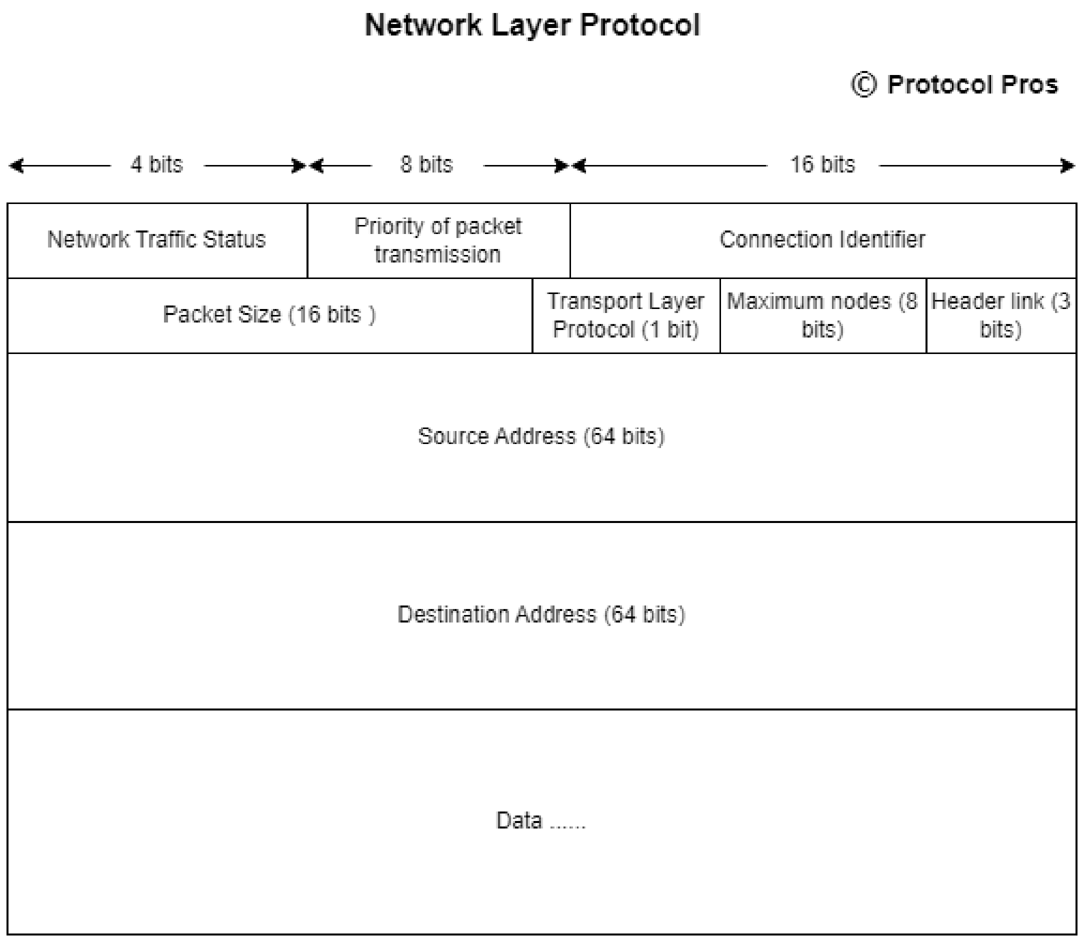


Fig: Network layer protocol

i. **Network Traffic Status:** represents the current status of the network traffic, so that the upper layer (transport layer) can manipulate the data transmission.

ii. **Priority of packet transmission:** holds the priority of the current packet bring transmitted. It helps in emergency and important transmissions (priority based)

iii. **Connection Identifier:** contains the identifier (pointer) to the connection that the current packet belongs to.

iv. **Packet Size:** size of the current packet

v. **Transport Layer Protocol:** Represents which transport layer protocol is being used for the communication.

   0 → connection-less transport layer protocol

1 → connection-oriented transport layer protocol

vi. **Maximum Nodes:** represents the maximum number of nodes that the packet can hop while transmission, even while executing congestion control.

vii. **Header Link:** link to the header of the next packet, so that flow control is maintained.

viii. **Source Address:** address of the source

ix. **Destination Address:** address of the destination

x. **Data:** data to be transmitted

Section 7: Conclusion

The specified design for our protocols that are utilized within the Autonomous Intersection Management System (AIMS) pertaining to road safety, and efficient intersection management are inclusive of all possible conditions. With this document, all stakeholders invested will have a technical and deeper understanding of the protocols that are directing traffic in the intersection. The use of this comprehensive view of the system capabilities, limitations, performance expectations and requirements, adjustments can be made to locate any errors and make the intersection more efficient for all agents. This document can be used effectively to maintain software development cycle and easily conduct the entire project utilizing it. Any changes in the requirements or design will lead the change in version of this document, and the changes will be specified in Section 8 (Document Version).

<u>Section 9</u>: Document Version

This document (version 1.0) is not yet edited.