

Научная статья

УДК 621.391

DOI:10.31854/1813-324X-2023-9-3-28-41



Application Layer Cooperative Automatic Repeat Request Method for Data Streaming over UAVs Network

✉ **Mohammed Amin Lamri**¹, lamri.amin@istu.ru
Albert Abilov², abilov.av@sut.ru
Alexandr Presnetsov³, alexandrpresnetsov@gmail.com

¹Kalashnikov Izhevsk State Technical University,
Izhevsk, 426069, Russian Federation

²The Bonch-Bruevich Saint-Petersburg State University of Telecommunications,
St. Petersburg, 193232, Russian Federation

³Neftemash LLC
Izhevsk, 426062, Russian Federation

Abstract: In this article, an evaluation study was conducted on an Application Layer Cooperative Automatic Repeat Request Algorithm ALC-ARQ designed for data streaming over a Wi-Fi Unmanned Aerial Vehicles standalone networks. A simulation model in NS-3 was implemented to investigate the performances of the method and conduct a comparative analysis with well-known routing protocols such like Ad hoc On Demand Distance Vector (AODV) and Optimized Link State Routing Protocol (OLSR) in terms of declaring link state information, rapidity of relaying and transmission range. The QoS metrics measured was Packet Loss Rate (PLR) and One-way Transmission Delay. Results show that the proposed method consistently outperform the classical routing protocols it terms of rapidity of relaying and transmission range. In addition, the results show that the method maintains its stability it terms of packet recovery along the relay-node transmission range and keeps the QoS metrics under the permissible rates.

Keywords: packet loss, error control, delay, data streaming, automatic repeat request

Funding: This research was funded by Russian Foundation for Basic Research according to the research project No. 19-29-06076.

For citation: Lamri M.A., Abilov A., Presnetsov A. Application Layer Cooperative Automatic Repeat Request Method for Data Streaming over UAVs Network. *Proc. of Telecommun. Univ.* 2023;9(3):28–41. DOI:10.31854/1813-324X-2023-9-3-28-41

Метод автоматического повторного запроса прикладного уровня для потоковой передачи данных по сети БПЛА

✉ **Мохаммед Амин Ламри**¹, lamri.amin@istu.ru
Альберт Винерович Абилов², abilov.av@sut.ru
Александр Михайлович Преснецов³, alexandrpresnetsov@gmail.com

¹Ижевский государственный технический университет имени М.Т. Калашникова,
Ижевск, 426069, Российская Федерация

²Санкт-Петербургский государственный университет телекоммуникаций им. М.А. Бонч-Бруевича,
Санкт-Петербург, 193232, Российская Федерация

³ООО «Нефтемаш»
Ижевск, 426062, Российская Федерация

Аннотация: В работе предложен и исследован метод кооперативного автоматического повторного запроса прикладного уровня ALC-ARQ, предназначенный для потоковой передачи данных по автономным сетям БПЛА. Имитационная модель в сетевом симуляторе NS-3 реализована для оценки производительности алгоритма и проведения сравнительного анализа с известными протоколами маршрутизации – AODV и OLSR. Измеренными показателями качества обслуживания являются коэффициент потери пакетов и односторонняя задержка передачи. Результаты показывают, что предложенный метод превосходит указанные протоколы таким параметрам, как: объявление информации о состоянии канала, скорость кооперативной ретрансляции и дальность передачи. Также он улучшает метрики качества обслуживания, сохраняет стабильность передачи с точки зрения восстановления потерянных пакетов в диапазоне передачи узла-ретранслятора и удерживает показатели односторонней задержки ниже допустимых значений.

Ключевые слова: потеря пакетов, контроль ошибок, задержка, потоковая передача данных, автоматический повторный запрос

Источник финансирования: Исследование выполнено при финансовой поддержке Российского фонда фундаментальных исследований № 19-29-06076.

Ссылка для цитирования: Ламри М.А., Абилов А.В., Преснецов А.М. Метод автоматического повторного запроса прикладного уровня для потоковой передачи данных по сети БПЛА // Труды учебных заведений связи. 2023. Т. 9. № 3. С. 28–41. DOI:10.31854/1813-324X-2023-9-3-28-41

I. Introduction

The theory of cooperative data streaming over Unmanned Aerial Vehicles (UAVs) networks has been studied in depth during the past few years [1–3]. Many researches, experimental and simulation studies were conducted for different transmission scenarios and network topologies. However, most of the existing research studies are intend to customize or improve the existed MAC protocols with or without additional support of physical layer techniques [4, 5]. In fact, the implementation of such methods and methods consider being complex and costly since it requires optimizations and changes of existed standardized protocols that may lead to incompatibility issues. Moreover, there is a high possibility of lake of effectiveness since OSI model can fail to meet the practical expectations.

In general, the goal behind cooperative transmission in UAVs networks is to improve the system performances in terms of reliability of link, network coverage and energy efficiency [6, 7]. In UAVs standalone network, because of their specific characteristics from high mobility of nodes to unreliability of transmission link, it is very crucial to address the two key issues: 1) Cooperative condition (since the quality of transmission links and the density of the network is constantly in change, a source node may not always need help from relay node; the condition by which the transmission switch to relay node is considered as sensible); 2) How to guarantee cooperative transmission (regardless the possibility of existence of error-control mechanisms on the down layers stack, the cooperative transmission must include a control mechanism to protect all ongoing transmission flows from potential packet loss and guaranty the data delivery at certain rate [8]).

Scholars have been working on cooperative data transmission for reliability and network coverage

purposes. In [9] the authors proposed a Multi-relay Cooperative Automatic Repeat ReQuest (MC-ARQ) MAC protocol based on the IEEE 802.11 DCF access scheme and CFC frames where they defined three schemes inspired by the use of different bakeoff time before transmission to choose optimal relay. However if two or more relays have the same T_i start timer, they will transmit simultaneously to the channel and a collision will occur as a consequence, which in turn leads the cooperative retransmission attempt to failure. In [10] the authors developed a selective retransmission scheme for multi-channel systems. The idea was to deliver the required data successfully using the available power and bandwidth within a limited number of transmissions using four schemes. In the Selective Automatic Repeat Request with Fixed Band scheme, the maximum number of transmission was limited. In the Selective Chase Combining with Fixed Bandwidth, data was detected using less number of transmission but implement a collection buffer to collect the unsuccessfully detected observations along with newly received. In Limited Selective Chase Combining with Fixed Band, a single buffer per channel can preserve only one previous observation per channel. And last, Selective Automatic repeat request with Variable Band was design similar to SARQ-Fixed Band but with the use of a single detector and the ability of increasing the channel bandwidth for each retransmission to achieve reliable transmission. Other works were also proposed for error control over cooperative transmission [11–13] where error correcting code, priority position inside Group of Pictures and, retransmission decision based on a given threshold are the key highlighted proposed features.

A long with the novel methods and models of cooperative reliable transmission that can be proposed for a specific scenario, a verification approach must be included to check their correctness and feasibility. The

most well-known approaches is the use of experiments in real-life, the use of simulation environments, the use of formal methods and the use of model checking. The use of simulation environment is one of the most optimal methods for verification, which consists of constructing a simulation model for the proposed method and schedule the main features and behaviors that produce the execution of the scenario proposed by entering their values as a set of parameters. The results of the simulation scenario represent a numerous abstraction level analysis of the method behavior; produce a reachability graph that includes all the simulation paths and compare it with the method expected results.

For a specific scenario of data streaming over UAVs standalone network, where a dynamic single relay-node is used to guarantee the cooperative transmission and the cooperative condition, which is dynamically, depend on the link state between the sender and receiver, an method Application Layer Automatic Repeat ReQuest (ALC-ARQ) is proposed. This method is inspired by the selective-repeat Automatic Repeat ReQuest error-control model, with a customized packet header on the application layer and a link-state routing scheme where all sending nodes are already aware about the map connectivity and they will route data transmission based on packet loss rate metrics.

The paper is organized as follows. Section II introduces the system model of cooperative transmission of ALC-ARQ and its error-control model as well as the QoS metrics measurement methods. In Section III we present the implemented simulation model in NS-3 simulation environment and describe in detail its components. The results of simulation model and the comparative analysis are conducted in section IV. The conclusions are drawn in section V.

II. System model and ALC-ARQ method design

In this section, the proposed method ALC-ARQ is described in detail, starting from cooperative transmission with relay deployment, criteria of deployment and link quality measurement.

A. System model for cooperative transmission

To ensure coverage and communication, lower-cost relay node may be inserted to forward data from each individual sending node in the network in a multi-hop transmission way to the server or data sink. Nodes locations are considered optimal if the resulting links can satisfy transmitting and communication coverage to a predefined packet delivery rate PDR value. In case of deterioration of the link, the measurements provided by the implementation of the measurement model (described in II.C) will be used to dynamically deploy additional transmission relay node or activate already deployed one.

For each sending node N_i we define two types of transmission link:

- Direct link D_i , between source and destination end nodes (N_i-BS) without relay;
- Relay link R_i , for the communication composition (N_i-R-BS) with relay.

Assuming that the relay node has the same transmission range as the sending nodes. The single-tiered network method proposed aims to ensure connectivity for each communication pair (N_i-BS) by dividing the unreliable long hop between N_i and BS to two short hops by deploying relay node. This relay will stand as a node helper to retransmit all the packets coming from each end node. In the first scenario, we consider one relay node in the network. When the transmission starts, each sending node will transmit data packets to the base station. For each transmission window, the reliability of the link will be measured by calculating packet loss rate (PLR).

Fig. 1 shows the scenario of deploying, activating and deactivating a relay node between source node and Base Station.

The deployment of relay will be fully automatic by the network without external interactions. Consequently, the topology of the network will be changed depending on the relay node state. To describe the system model, we define the following states.

Standby mode (Fig. 1a). BS starts to receive data without the interaction of the relay node after UAVs have been launched (P2P mode). During this period, the quality of D_i link will be measured. The window size used for metrics measurements is defined based on the parameters of the scenario proposed (Wi-Fi standard, channel bandwidth, node velocity and reordering buffer size on the application layer). Because packet loss rate is dependent on link quality and network interference, loss events will occur which in turn invoke the application layer ALC-ARQ error control method [14]. The constant mobility of source nodes away from BS during the transmission will decrease the PLR measured on the application layer on BS . If PLR value for a given link D_i exceeds a predefined threshold, the BS will broadcast a control message packet to all nodes in the network to notify the correspondent source node about the D_i link state. Threshold value is defined by the system's user when starting the application. The source node in its turn after receiving control message will switch to multicast transmission including relay node R as intended destination (node helper) on its transmission table. Consequently, another transmission link R_i will be created between N_i and BS as a combination of two sub-links (N_i-R , $R-BS$) and the relay node will switch to active mode.

Active mode (Fig. 1b). After receiving the control message, the relay node starts by creating a playback buffer for the newly created link (N_i-R). This buffer will be used for error control purposes like packet recovering and reordering. Buffer parameters such as reordering buffer offset and reordering late time will be

chosen as input parameters when starting the relay application. Relay node will listen to packets coming from both source-node and BS, and determine the type of each packet depending on packet_id field in ALC-ARQ header. If the packet contains video data and does not exist in the playback buffer, the relay will keep a copy of it in the corresponding buffer, and then forward it to the BS. Since the source node is multicasting data packets, the BS will continue to check the reliability of links *DI* and *RI* by measuring PLR for each flow for the same sending window size. This measurement allows the BS to define the most reliable link from which packets will be accepted.

As a result, packets coming from less reliable link will be ignored. During transmission and on both *BS* and *R*, loss events will be recognized by checking out-of-order received packets using the sequence numbers (*SM*) assigned in ALC-ARQ header. Consequently, the lost packets will be requested by broadcasting a Negative Acknowledgment (NACK) message. For (*Ni-BS*) link, source node will attempt to retransmit the requested packet(s) if the NACK packet was successfully received. For (*Ni-R*, *R-BS*) link, both source-node and *R* will attempt to retransmit the requested packet(s). However, for each requested packet, *BS* will accept only the first correct received packet by one link and ignore the others. If the attempt of packet recovery fails, the *BS* will broadcast the same NACK again for each Round Trip Time RTT value until Retransmission Timeout RTO is reached [15] or the requested packet no longer exists in the buffer. Since playback buffers on source-node and *R* node are set up with different parameters' values, if the requested packet does not exist in the relay buffer, the relay node will forward this request to the source node in an attempt to recover the maximum number of lost packets. The average delay for all packets and average delay for only retransmitted packets on both links will be measured.

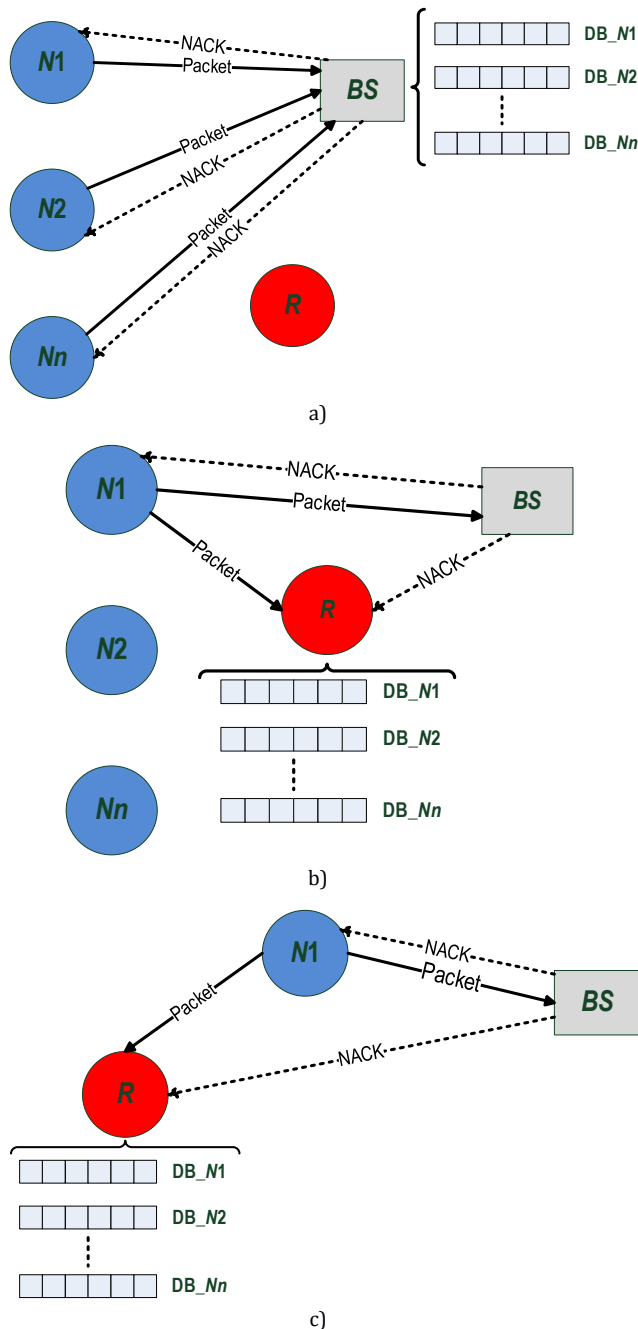


Fig. 1. System Model of Cooperative Transmission: a) Transmission without Relay; b) Relay is Deployed; c) Relay Out of Range (Background Mode)

While the reliability of links is measured, the choice of receiving link will be chosen as follow:

$$DI \text{ if } PLR_{DI} \geq \text{Threshold} \vee PLR_{DI} > PLR_{RI}, \quad (1)$$

$$RI \text{ if } PLR_{DI} \leq PLR_{RI} \wedge PLR_{DI} < \text{Threshold}. \quad (2)$$

If the first condition is satisfied, the relay node will switch to background (inactive) mode.

Background mode (Fig. 1c). The condition for which relay node switch to background mode is when PLR of link *DI* is greater than or equal to the threshold value or the packet loss rate of link *RI* (1). The *BS* then will send a control message to notify both source and relay that data will be received by *DI* only. However, the source node will continue to multicast data packets to ensure that the relay node is still receiving data packets but without involving in the packet delivery process. The reason behind this is to keep *RI* link active with the minimum consumption of network and memory resources (bandwidth and buffer). This can be achieved by saving only sequence numbers of the received packets on relay's buffer. Thereafter, based on these sequence numbers, *BS* will keep checking the quality of each link till the condition in equation (2) is satisfied to switch to state II (Active mode).

B. Cooperative error-control ALC-ARQ method description

The ALC-ARQ transmission block scheme is represented in Fig. 2. This scheme is inspired by the selective-repeat Automatic Repeat ReQuest error-control model with a customized packet header on the application layer and a link-state routing scheme where all sending nodes are already aware about the connectivity map and they will route data based on packet loss rate metrics.

The proposed scheme defines two error-control phases.

1 phase: Relay cooperative error-control

The performance of ALC-ARQ error control is measured by its capacity to alleviate packet loss. The relay node experiences a loss event whenever $SN_b - SN_a > 1$ where: SN_b and SN_a represent the sequence numbers of two consecutive received packets. For a window size denoted n_bit , let $P(L)$ denote the probability of a loss event. To guarantee the reliability of data transmission, the probability $P(L)$ has to be very small. The performance of ALC-ARQ packet recovery is dependent on what type of loss events may occur during transmission.

For this purpose, we define the following probability:

– P_n is the probability for n_bit to be received without loss;

– P_s is the probability for n_bit to be received with n series of losses with burst size equal to one;

– P_b is the probability for n_bit to be received with n series of losses with burst size greater than one.

Accordingly $P(L) = P_n + P_s + P_b$. P_n represents the ideal link state between the source-node and the relay node where there is no interference or channel errors. Assuming that the transmission link is not ideal ($P_n = 0$) and P_s is a special case of P_b , then the probability $P(L)$ for loss event for n_bit is equal to P_b and define as:

$$P_b = 1 - \exp\left(\frac{1}{P_i \cdot SNR(\theta_i)} \cdot \left(2^{\frac{r}{w}} - 1\right)\right). \quad (3)$$

In this function, the probability of loss event is a function of transmission power P_i used to transmit the i -th packet and the normalized expected Signal-to-noise Ratio $SNR(\theta_i)$ given the fading level θ_i [16], where r and w represent the transmission rate (bits/sec) and bandwidth respectively. In section III, the values of P_i , θ_i , r and w are determined for the simulation model on the network device layer.

The customization that was made on the application layer was designed to handle more than one loss event using one NACK packet (loss event is defined by SN of the first lost packet and burst length – BL). According to Fig. 2, the relay node keeps asking for the lost packets for each RTO while receiving data packets. In other words, another loss event(s) may occur and the previous event has not been handled yet. In this situation, instead of sending another NACK message to request the lost packets of the last loss event, the same NACK will be edited to include SN and burst length of the second event. Fig. 3 represents the format of NACK packet.

In this way, one NACK packet can handle more than one loss event by adding SN_i and BL_i of the occurred loss. Furthermore, if not all the packets are successfully recovered, the burst can be break up into N sub-bursts with the corresponding SN and BL .

As mentioned before, the process of sending NACK messages is repeated if RTO value is exceeded until the

requested packets are no longer relevant to the application. RTO value is defined by default when starting the application to 70 ms.

However, it will be updated for each loss event based on the approximated time value of Round Trip Time RTT in 4 phases [17]:

1) the current smoothed time is calculated Smoothed Round Trip Time ($SRTT_{curr}$) which uses the smoothed round trip time previous value:

$$SRTT_{curr} = \frac{7}{8} \cdot SRTT_{prev} + \frac{1}{8} \cdot RTT_{curr}, \quad (4)$$

where $SRTT_{prev}$ – previous smoothed round trip time; RTT_{curr} – current round trip time;

2) the current deviation is determined DEV_{curr} :

$$DEV_{curr} = |RTT_{curr} - SRTT_{prev}|; \quad (5)$$

3) the current smoothed deviations value is calculated:

$$SDEV_{curr} = \frac{3}{4} \cdot SDEV_{prev} + \frac{1}{4} \cdot DEV_{curr}, \quad (6)$$

where $SDEV_{prev}$ – previous smoothed deviations value;

4) the retransmission timeout is calculated using the following formula:

$$RTO = SRTT_{curr} + 2 \cdot SDEV_{curr}. \quad (7)$$

2 phase: Source-node error control

Source node implements the same mechanism of error control. On relay node, if the requested packets do not exist in the playback buffer, the received NACK will be forwarded to the source node. The probability of recovering these packets depends on its relevance to the application on BS which in turn is related to byte offset and late time buffer parameters that are set by the application user. As a result, if the requested packets no longer exist in the buffer, the source node multicasts a cancellation message to inform BS that the requested packets no longer exist. BS after receiving the cancellation message will delete the sequence numbers of the requested packets from its waiting list.

To analyze the performances of ALC-ARQ on the application layer using a predefined packet loss rate and burst lengths, an integrated method for generating artificial packet loss events is implemented based on the classical 2-state Markov model introduced by Gilbert and Elliott [18] in Fig. 4. The proposed method is implemented on the source-node side, where after generating data packets, the Gilbert – Elliott average loss probability P will decide if the packet will be sent or dropped, and the average burst size L will define the sequence of dropped packets. P and L values define the transaction probabilities p and q by which the system moves from state G (send packet) to state B (drop packet) by the following formulas:

$$p = \frac{1}{L}, \quad q = \frac{P}{L(1-P)}. \quad (8)$$

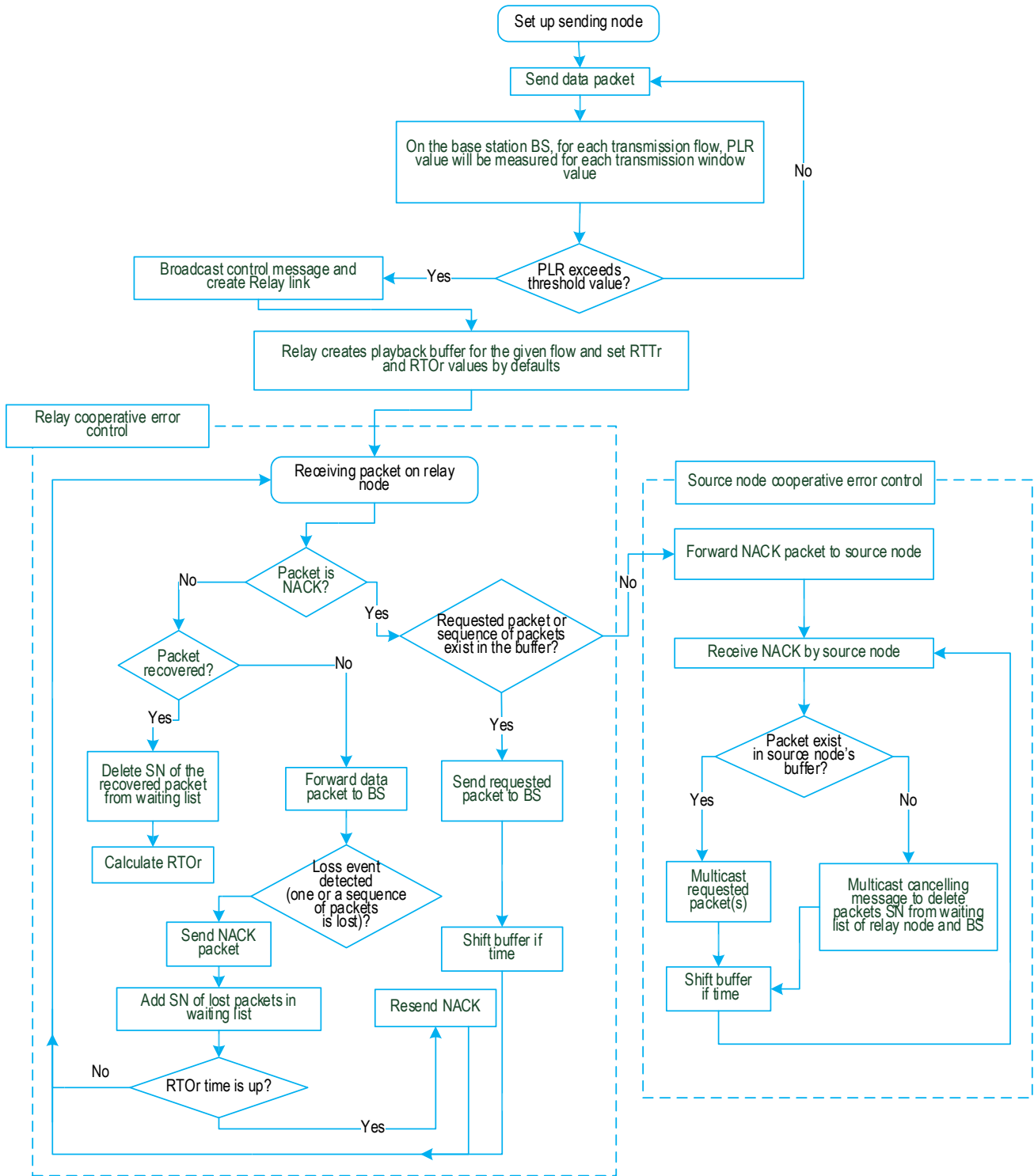


Fig. 2. ALC-ARQ Transmission Block Schema



Fig. 3. NACK Packet; ID) Packet Identifier 4 Bits; NR) Node Identifier 4 Bits; N) Number of NACK Repeats 8 Bits; SN) Sequence Number of the First Lost Packet in Succession 32 Bits; BL) Burst Length 8 Bits

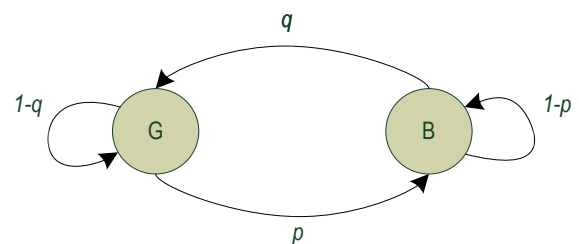


Fig. 4. 2-State Markovian Model for Artificial Packet Loss Generation

The method was implemented in C++ programming language. The implementation is provided on the appen-dix below.

```

Appendix
HEADER FILE gilbert_elliott.h

#ifndef GILBERT_ELLIOTT_H
#define GILBERT_ELLIOTT_H
#include "socket_io.h"
#define MAX BURST LENGTH GEC 1000
class gilbert Elliott
{
    int p,q; // transition probabilities
    bool state;
public:
    short stat[MAX BURST LENGTH GEC];
    unsigned char bl; //burst length
public:
    gilbert_Elliott();
    void initGilbert Elliott(double plr, double lb avr);
    bool getState(void);
    bool getCurrentState(void);
    int randomNumber(int hi);
    //int write_stat_to_file(FILE *file);
};
#endif // GILBERT ELLIOTT H

CLASS FILE gilbert_elliott.cc

#include "gilbert_elliott.h"
gilbert Elliott::gilbert Elliott()
{
    p = q = 0;
    bl = 0;
    for (int i=0; i<MAX_BURST_LENGTH_GEC; i++)
        stat[i]=0;
    state = true;
}

void gilbert Elliott::initGilbert Elliott(double plr, double lb_avr)
{
    srand( (unsigned)time( NULL ) );
    p = (1 / lb_avr) * 10000;
    q = (plr / (lb_avr * (1 - plr)))*10000;
    state = true;
}

bool gilbert Elliott::getState()
{
    if (state)
    {
        if (randomNumber(10000) < q)
        {
            state = false;
            bl++;
        }
    }
    else
    {
        if (randomNumber(10000) < p)
        {
            state = true;
            if ((bl)&&(bl <
MAX BURST LENGTH GEC))
            {
                stat[bl]++;
                bl=0;
            }
        }
        else
            bl++;
    }
    return state;
}

bool gilbert Elliott::getCurrentState()
{
    if (!state)
        bl++;
    return state;
}

int gilbert Elliott::randomNumber(int hi)
{
    float scale = rand()/float(RAND_MAX);
    return int((float)scale*hi);
}

```

C. QoS Measurement Model

In this section, we will analyze the performances of ALC-ARQ method in terms of packet loss rate PLR and one-way transmission delay on the application layer.

1) Packet Loss Rate

Packet reordering and recovering on the *BS* side is handled by the receiving buffer. For each dataflow f_i the packet loss rate will be measured for both transmission links Dl and Rl as the ratio of the number of lost packets to the total number of sent packets as:

$$PLR(f_i) = 1 - \frac{N_g}{N_r}, \quad (9)$$

where N_g and N_r represent the number of transmitted packets and the number of received packets respectively.

It should be noted that erroneous data that has failed to be detected by Forward Error Correction (FEC) mechanism of data link layer [19] and has been delivered to upper layers are out of scope since ALC-ARQ error control considers error events as a defect in the succession of sequence numbers of the received packets. The performance of ALC-ARQ will be investigated in relation to node velocity and distance between end nodes (source-node and *BS*). The traffic generator model in source-application uses fixed intervals to schedule the next packet transmission, the value is set by default to 1.7 ms for all different velocity values. Based on this, the receiving window for which PLR will be measured is defined dynamically and depends on the given velocity.

2) One-way transmission delay

The proposed method for calculating a one-way transmission delay requires a registered timestamp (sending time) for each sent packet in the application layer header. As the simulation study is handled on one machine, there was no need to distribute a clock sync signal to handle synchronization between the network units [20]. Based on this, the one-way transmission delay will be measured as presented in (10):

$$D_i = T_{reception} - T_{transmission}, \quad (10)$$

where $T_{reception}$ and $T_{transmission}$ are respectively the reception time by *BS* and the transmission time packet fixed on the application layer.

III. NS-3 simulation model of ALC-ARQ

To conduct performance evaluation, NS-3 is used, as it is a packet-level network simulator where the main unit of modeling is packets and entities that exchange packets. The modelling process in NS-3 follows the workflow presented in Fig. 5.

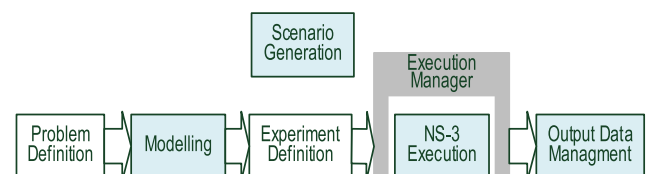


Fig. 5. NS-3 Workflow for the Modelling Process

A. ALC-ARQ packet generation in NS-3

Data packets are generated as special data buffers with space for headers, trailers, tags and packet metadata. At the application layer, the packets are created with dummy data bytes. We choose a 1250 bytes size for packets that are considered to carry video data. The headers that contain information about packet type, application layer *SN*, transmission Node ID, transmission timestamp, node IP address are created using the implemented class `PacketDataTag.cc` that is derived from the class `Tag.cc` of NS-3 as shown in Fig. 6. As the packet moves down the stack, the stack layers will add the corresponding headers and append it to the packet, and the reverse will happen in the receive process.

```
Ptr<Packet> packet = Create<Packet>(MTU_SIZE);
PacketDataTag tag;
tag.SetNumberOfRepeat (0);
if(gal_pn == MAX_PN) gal_pn = 0; else gal_pn++;
tag.SetSeqNumber (gal_pn);
tag.SetNodeId (GetNode ()->GetId ());
tag.SetPacketId (IDM_UDP_ARQ_VIDEO);
tag.SetTimestamp (Simulator::Now ());
tag.SetTreeNumber (0);
m_my_addr.Serialize (tag.sourceAddr);
packet->AddPacketTag (tag);
```

Fig. 6. Code Snippet for Data Packet Creation in NS-3

B. ALC-ARQ application model in NS-3

NS-3 simulator was not designed to provide so much fidelity on the application layer, because it typically uses a packet generator models such as Net devices (https://www.nsnam.org/docs/release/3.29/doxygen/classns3_1_1_net_device.html) that aims to characterize the traffic generated by the real applications. In other words, it does not run real applications but just models of how to generate packets. The lack of such functionality led us to create an NS-3 custom component for the ALC-ARQ method to implement all the functionalities proposed in it. We started by creating 3 applications (source-application, destination-application and relay-application) as subclasses of NS-3 application class (https://www.nsnam.org/docs/release/3.16/doxygen/classns3_1_1_application.html) then, we used the predefined UDP socket API on NS-3 which is based on BSD socket API (https://www.nsnam.org/docs/release/3.19/doxygen/classns3_1_1_socket.html#details) for communication between end nodes. The packets are generated with a custom header as shown in Fig. 6, and the traffic management is implemented as a C++ Client/Server structured application. All the measurements (PLR calculation, RTO, RTT and delay) are measured on the application layer. The metrics include all the down stack OSI layers values.

C. End nodes Net Devices

This class represents the API, which the IP and ARP layers need to access to manage the instance of a

network device layer (https://www.nsnam.org/docs/release/3.29/doxygen/classns3_1_1_net_device.html#details). Typically, it is installed on the nodes using a net device helper (https://www.nsnam.org/docs/release/3.19/doxygen/classns3_1_1_net_device_container.html#details). This class is used to extend the MAC layer functions and it hides as many MAC-level details to allow a single layer three to work with any kind of MAC layer. For ALC-ARQ to configure the net devices, we used `WifiHelper.cc` class to create the Wi-Fi Net Device object and configure its attributes. To create and manage the Wi-Fi physical layer we used the `YansWifiPhyHelper.cc` class and `YansWifiChannelHelper.cc` class to create and manage Wi-Fi channel objects for the created model. The configuration method for 802.11ax 5GHz is presented in Fig. 7.

```
NetDeviceContainer
NetDeviceSetup::ConfigureDevices (NodeContainer&
nodes)
{
int channelWidth = 80;
std::string Vht = "7";

WifiHelper wifiHelper;
wifiHelper.SetStandard(WIFI_PHY_STANDARD_80211ax_5GHz);
YansWifiPhyHelper wifiPhyHelper =
YansWifiPhyHelper::Default ();
wifiPhyHelper.SetPcapDataLinkType (YansWifiPhyHelper::DLT_IEEE802_11_RADIO);
wifiPhyHelper.Set ("TxPowerStart",
DoubleValue(13));
wifiPhyHelper.Set ("TxPowerEnd", DoubleValue(13));
wifiPhyHelper.Set ("RxNoiseFigure", DoubleValue
(7));
wifiPhyHelper.Set ("ChannelWidth", UIntegerValue
(channelWidth));
wifiPhyHelper.Set ("Frequency",
UIntegerValue(5180));
wifiPhyHelper.Set ("Antennas", UIntegerValue(1));
wifiPhyHelper.Set ("ChannelNumber",
UIntegerValue(42));

YansWifiChannelHelper wifiChannelHelper;
wifiChannelHelper.SetPropagationDelay
("ns3::ConstantSpeedPropagationDelayModel");
wifiChannelHelper.AddPropagationLoss ("ns3::FriisP
ropagationLossModel", "SystemLoss",
DoubleValue(1), "Frequency", DoubleValue(5.18e9));
wifiPhyHelper.SetChannel
(wifiChannelHelper.Create ());
wifiHelper.SetRemoteStationManager ("ns3::Constant
RateWifiManager", "DataMode", StringValue
("VhtMcs" + Vht), "ControlMode", StringValue
("VhtMcs0"));
WifiMacHelper macHelper =
HtWifiMacHelper::Default ();
macHelper.SetType ("ns3::AdhocWifiMac");
NetDeviceContainer devices = wifiHelper.Install
(wifiPhyHelper, macHelper, nodes);

return devices;
}
```

Fig. 7. Code Snippet for Net Device Configuration for ALC-ARQ

D. Mobility model In NS-3

Mobility models are used to track and maintain the current Cartesian position and speed of a node/object (<https://www.nsnam.org/docs/models/html/mobility.html>). It is composed of a set of helper classes, which can be used to place nodes and set up their mobility. In our proposed scenario, the sending nodes are flying away from the BS with a specific velocity while relay node is located on a fixed position away from the BS. The transmission link between relay node and BS is guaranteed as ideal. The two mobility models ConstantVelocityMobilityModel.cc and ConstantPositionMobilityModel.cc were used to set the Cartesian position of the sending node and both BS and relay node respectively. The mobility model configuration is presented in Fig. 8.

```
MobilityHelper mobility;
mobility.SetMobilityModel("ns3::ConstantVelocityM
obilityModel");
mobility.Install(nodes);

for (uint32_t i=1 ; i<nodes.GetN() - 1; i++)
{
Ptr<ConstantVelocityMobilityModel> cvmm =
DynamicCast<ConstantVelocityMobilityModel>(nodes.
Get(i)-> GetObject<MobilityModel>());
cvmm->SetPosition( Vector (distance,0,0));
cvmm->SetVelocity( Vector (0.2,0,0));
}

MobilityHelper relay_mobility;
Ptr<ListPositionAllocator> positionAlloc =
CreateObject<ListPositionAllocator>();
positionAlloc->
Add(Vector(relay_position,2.0,0.0));
positionAlloc->Add (Vector (0.0, 0.0, 0.0));

relay_mobility.SetPositionAllocator(positionAlloc);
relay_mobility.SetMobilityModel("ns3::ConstantPos
itionMobilityModel");
mobility.Install (nodes.Get (nNodes - 1));
```

Fig. 8. Code Snippet for ALC-ARQ Mobility Configuration

IV. Simulation experiments and results

To understand the difference of packet delivery efficiency of ALC-ARQ between point-to-point and multi-hop (one relay node between source node and destination) topology, the testbed shown in Fig. 9 is used to set up the technical and execution requirements for the simulation stand where flying source nodes stream video data to the base station on the ground. The relay node and BS are placed symmetrically 25 m apart from each other where the path between them is ideal. The source nodes during transmission are flying away from the base station location toward the relay node, the distance between BS and source node when starting the simulation is 0 m, by the end of simulation, the distance is dependent on the Wi-Fi standard transmission capabilities and its configuration parameters. Other simulation parameters are listed in Tab. 1.

The performance of ALC-ARQ while using the Gilbert – Elliott model for artificial packet loss is investigated. The dependence of Packet Loss Ratio on the distance between the source and destination node is explored, in addition to the position of the source-node, the mobility of nodes and the time required to switching to a relayed transmission. The improvements which ALC-ARQ has given related to the Packet Loss Ratio and topology change compared to classical well known routing protocols such as Optimized Link State Routing Protocol (OLSR) and Ad hoc On Demand Distance Vector (AODV) were shown. The effect of recovery process on the application level one-way transmission delay was derived.

TABLE 1. Simulation Parameters

Parameter	Setting
Channel delay model	ConstantSpeedPropagationDelayModel
Propagation loss model	FriisPropagationLossModel
Transmission protocol	UDP
Application data rate	6 Mbps
Packet size	1250 bytes
Mobility speed	[1; 60] km/h
Simulation topology	End to End; Multi-hop
Number of transmission nodes	[1; 10] nodes
PLR measurement window	[60; 740] packets
Playback buffer size	248 packets for source nodes; 148 for both relay and BS

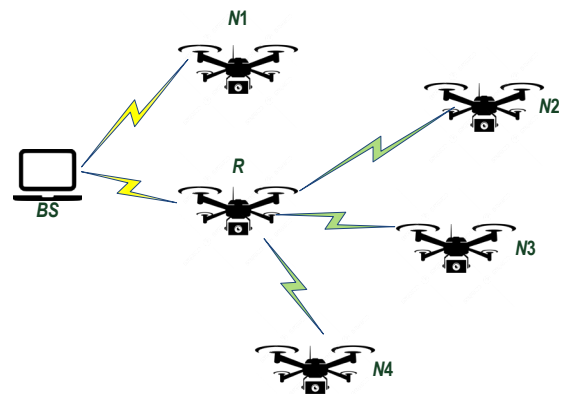


Fig. 9. Testbed for Simulation Scenario

Fig. 10 shows the results of packet delivery simulation using artificial packet loss and burst length for ALC-ARQ method. The simulation results present the theoretical analysis for the efficiency of the method and its ability to recover lost packets on the application layer while using a non-reliable transmission protocol on the transport layer. It is obvious that the packet delivery is enhanced by the use of ALC-ARQ where the path is in pure conditions (0.15–0.6 in Given PLR axis). The reason is the lightweight packet request scheme on the application level that uses dynamic NACK packets to request lost packets of different burst size; this scheme does not require a high competition for media access, processing time and buffer management for packet

recovering. We further investigate the cooperative transmission of ALC-ARQ by measuring the packet loss rate for mobility scenario. Fig. 11 represents PLR measurements for cooperative transmission. The threshold value after which the transmission is switched to cooperative is set to 0.1.

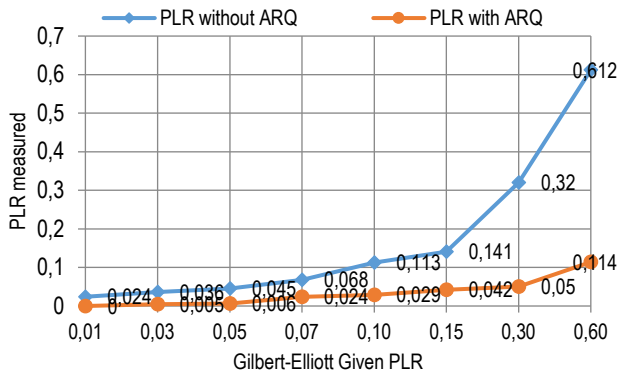


Fig. 10. Packet Loss Rate Measurement for Artificial Packet Loss Model. Burst Length Equals 18 Packets

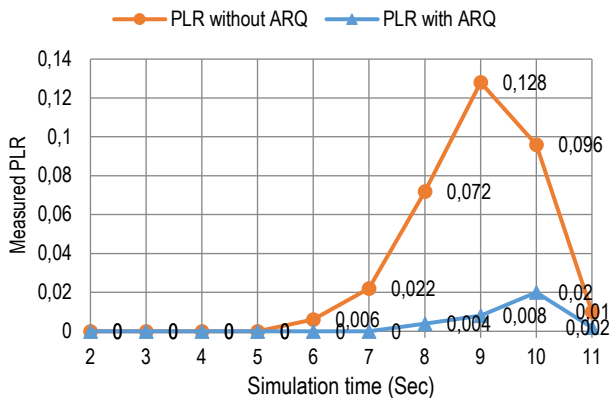


Fig. 11. Packet Loss Rate for Cooperative Transmission for Threshold Value Equals 0.1 (10%) and Nodes Velocity Equals 14 km/h

As we see in Fig. 11, while the source node is flying away from the BS (node velocity = 14 km/h), for each time value, ALC-ARQ calculates the packet loss rate and compares it to the threshold value. On second 9, the transmission was switched to cooperative. This change in the transmission topology is noticed by the sharp decrease in the value of packet loss rate.

For this simulation, it should be noticed that the threshold value was set to the PLR values measured without ALC-ARQ packet recovery. However, the system user can also set the threshold value for PLR of ALC-ARQ.

To justify the effectiveness of the ALC-ARQ proposed method, a comparative analysis with some well known routing protocols such as OLSR and AODV should be conducted in terms of declaring the link state information, rapidity of relaying, and transmission range. In Fig. 12, we present the simulation results of the packet loss rate measurement for OLSR, AODV, and ALC-ARQ at different threshold values. As we can see from this figure, because ALC-ARQ does not have to periodically announce the information about the relay's state, it

means less number of transmission is required, the overhead of flooding messages is in its minimal value compared with routing protocols. From the point of view of declaring link state, rapidity of relaying, and knowing that routing protocols do not provide any error control processes, our method provides better rapidity of relaying and better transmission range. From the graph, it is shown that AODV takes 2 measurements (38 and 39 m) to maintain routing tables, get information about nodes before switching to relay transmission. For OLSR it is shown that switching to relay transmission takes longer by 4 measurements (37, 38, 39 and 40) to maintain routing tables, select the path and carry out the handshake procedure before switching to relay transmission.

Fig. 13 represents the results of PLR measured for different source-node velocity values. As we can see, the method maintains its stability in terms of relaying process and packet recovery along the range of transmission through the relay node, the threshold value defined for each simulation was 0.1 (10%). The difference between the results was the distance at which the relay node was activated: 37 (Fig. 13a), 36 (Fig. 13b) and 36 m (Fig. 13c). For packet recovery effectiveness for the second phase (where the relay node is active and starting from position 58 m), we recorded an increase in PLR values measured on the same node position for different velocities. For example, for position 61 m, 15 km/h \approx 0.041; 30 km/h \approx 0.08; 60 km/h \approx 0.22. This is related to signal strength and the transmission range for physical and data link layers.

Fig. 14 represents the analyses of a sequence of 700 successfully received packets by BS by order. The PLR value measured for the given sample was 0.09 (9%) for both graphs; transmission without relay (transmission delay for *DI* link) and with relay (transmission delay for *RI* link). It is shown that if the Transmission experiences some loss events, and then the one way-transmission delay for the recovered packets increases for both graphs. For *DI* link, if the transmission is lossless, the average transmission delay is between 0.85 and 0.96 ms. and the average delay for recovered packets is between 1.67 and 1.77 ms. For *RI* link, if the transmission is lossless, the average transmission delay is between 1.89 and 1.95 ms and average delay for recovered packets is between 5.51 and 5.54 ms. For comparative results, if the relay node is active, the lossless transmission delay increases by 1.96 times the value of transmission delay without relay node. The increasing rate totally conforms to the assumptions of the theoretical study since the sending packets are sent through two sub-links passing by the same process. Although, if the transmission experiences loss events then the average delay for recovered packets is increased by 3.3 times the value of recovered packets using the direct link. This is because relay node have to forward NACK packet to source nodes if the requested packets no longer exist in its buffer, which needs extra time for retransmission.

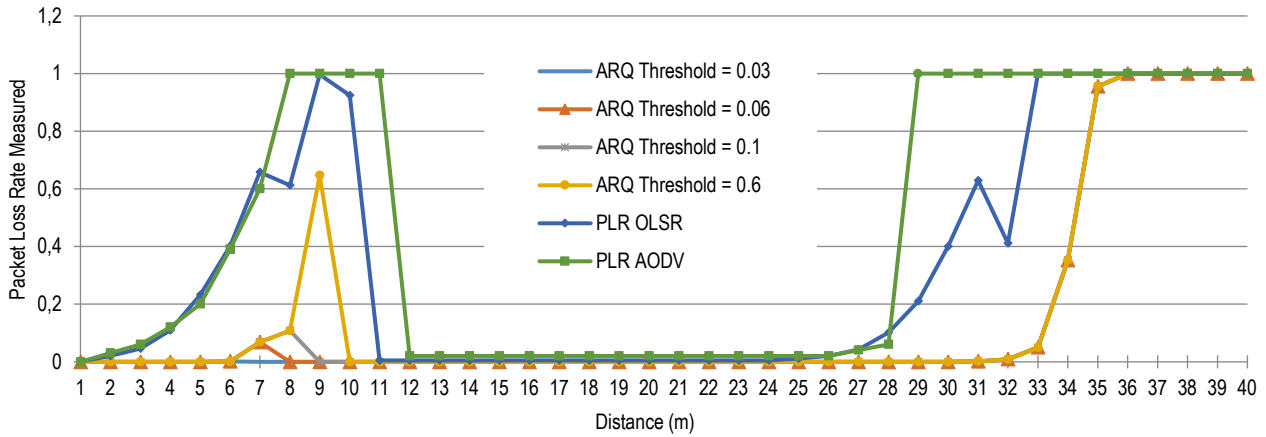


Fig. 12. Comparative Analysis of OLSR, AODV and ALC-ARQ for Different Threshold Values

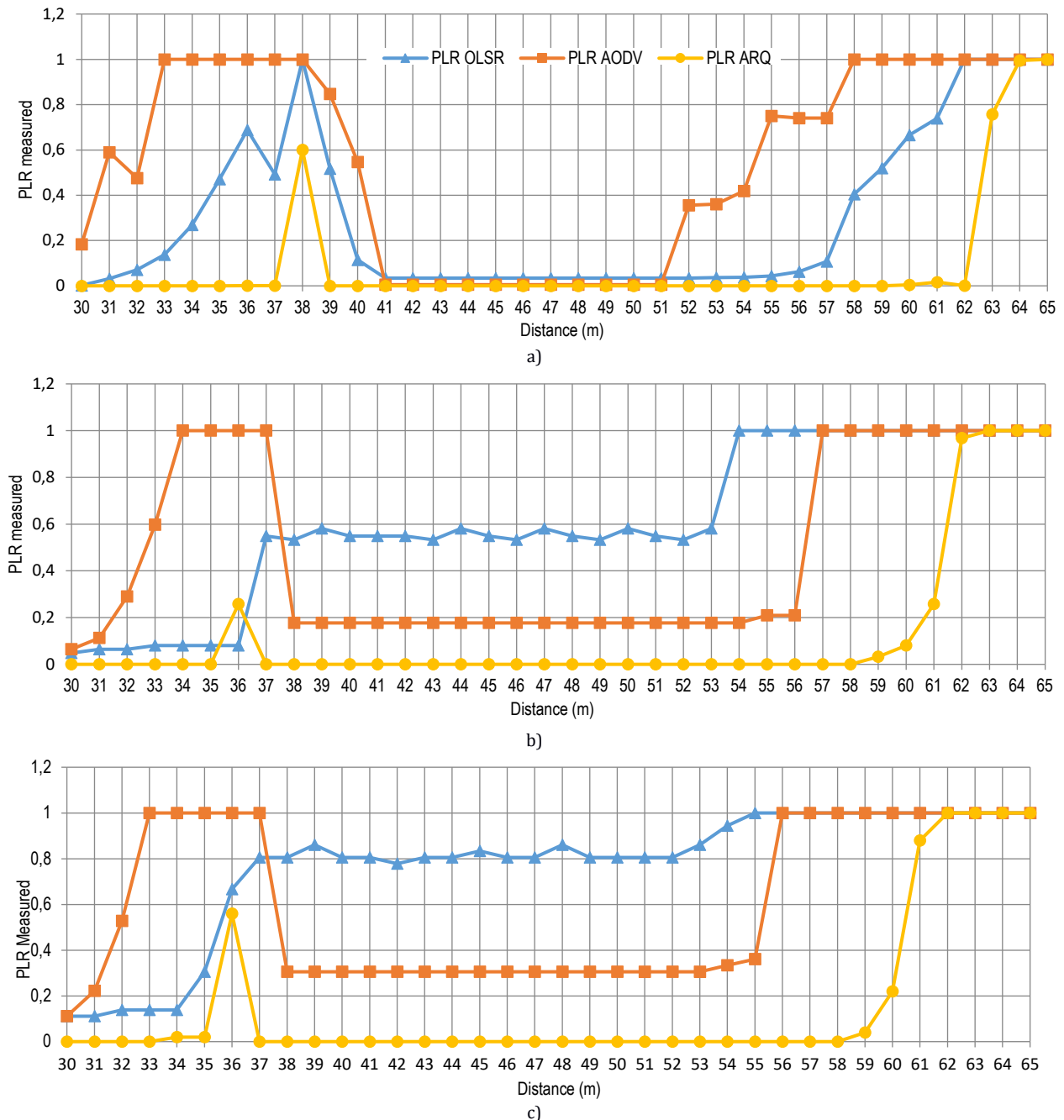


Fig. 13. Comparative Analysis for OLSR, AODV and ALC-ARQ for Different Velocity Values: 15 (a); 30 (b) and 60 km/h (c)

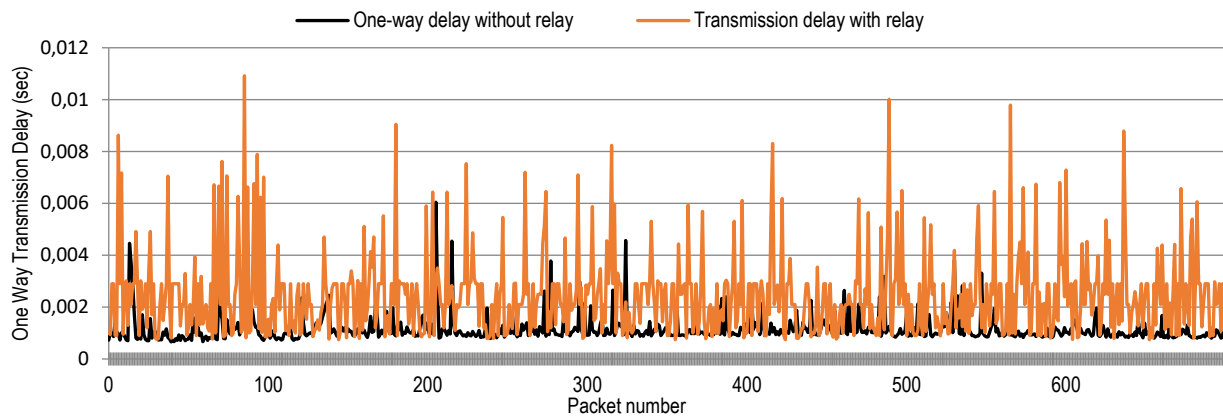


Fig. 14. One-Way Transmission Delay for ALC-ARQ Packet Loss Rate PLR = 0.09

V. Conclusion

In this work the effectiveness of the ALC-ARQ proposed method was justified in comparison with OLSR and AODV routing methods in terms of the link state information, rapidity of relaying, and transmission range. The method has the ability to recover lost packets on the application layer while using a non-reliable transmission protocol on the transport layer. The packet delivery is enhanced where the path is in pure conditions as a result of the lightweight packet request scheme on the application level that uses dynamic NACK packets to request lost packets of different burst size, this scheme does not require a high competition for media access, processing time and buffer management for packet recovering. The cooperative transmission of ALC-ARQ was investigated by measuring the packet loss rate for mobility scenario.

While the source node is flying away from the *BS* at certain time, the transmission was switched to cooperative, where the change in the transmission topology is noticed by the sharp decrease in the value of packet loss rate. The threshold value was set to the PLR values measured without ALC-ARQ packet recovery; however, it can also be set to the threshold value for PLR of ALC-ARQ. Because ALC-ARQ does not have to periodically announce the information about the relay's

state, it means less number of transmission is required, the overhead of flooding messages is in its minimal value compared with routing protocols.

From the point of view of declaring link state, rapidity of relaying, and knowing that routing protocols do not provide any error control processes, our method provides better rapidity of relaying and better transmission range. The method maintains its stability in terms of relaying process and packet recovery along the range of transmission through the relay node. Transmission experiences some loss events, and then the one way-transmission delay for the recovered packets increases. For comparative results, if the relay node is active, the lossless transmission delay increases by 1.96 times the value of transmission delay without relay node. The increasing rate totally conforms to the assumptions of the theoretical study since the sending packets are sent through two sub-links passing by the same process. If the transmission experiences loss events then the average delay for recovered packets is increased by 3.3 times the value of recovered packets using the direct link. This is because relay node have to forward NACK packet to source nodes if the requested packets no longer exist in its buffer, which needs extra time for retransmission.

References

1. Ngo H.A., Hanzo L. Hybrid Automatic-Repeat-reQuest Systems for Cooperative Wireless Communications. *IEEE Communications Surveys and Tutorials*. 2013;16(1):25–45. DOI:10.1109/SURV.2013.071913.00073
2. He X., Kumar R., Mu L., Gjørseter T., Li F.Y. Formal verification of a Cooperative Automatic Repeat reQuest MAC protocol. *Computer Standards and Interfaces*. 2012;34(4):343–354. DOI:10.1016/j.csi.2011.12.001
3. Jamshidi A. Efficient cooperative ARQ protocols based on relay selection in underwater acoustic communication sensor networks. *Wireless Networks*. 2019;25:4815–4827. DOI:10.1007/s11276-018-1773-5
4. Goel J., Jagadeesh H. Listen to Others' Failures: Cooperative ARQ Schemes for Low-Latency Communication Over Multi-Hop Networks. *IEEE Transactions on Wireless Communications*. 2021;20(9):6049–6063. DOI:10.1109/TWC.2021.3071504
5. Tutgun R., Aktas E. A Markovian Analysis of Cooperative ARQ with Random Access. *Wireless Personal Communications*. 2022;123:3201–3211. DOI:10.1007/s11277-021-09282-6
6. Goel J., Harshan J. Minimal Overhead ARQ Sharing Strategies for URLLC in Multi-Hop Networks. *Proceedings of the 93rd Vehicular Technology Conference, 25–28 April 2021, Helsinki, Finland*. IEEE; 2021. DOI:10.1109/VTC2021-Spring51267.2021.9448948
7. Mheich Z., Savin V. Cooperative communication protocols with energy harvesting relays. *Proceedings of the Wireless Days, 29–31 March 2017, Porto, Portugal*. IEEE; 2017. p.60–65. DOI:10.1109/WD.2017.7918116

8. Kim S., Kim B.S., Kim K.H., Kim K.I. Opportunistic Multipath Routing in Long-Hop Wireless Sensor Networks. *Sensors*. 2019;19(19):4072. DOI:10.3390/s19194072
9. He X., Li F.Y. A Multi-Relay Cooperative Automatic Repeat Request Protocol in Wireless Networks. *Proceedings of the International Conference on Communications, 23–27 May 2010, Cape Town, South Africa*. IEEE; 2010. DOI:10.1109/ICC.2010.5502169
10. Shafique T., Abdelhady A.M., Amin O., Alouini M. S. Energy Efficiency, Spectral Efficiency and Delay Analysis for Selective ARQ Multichannel Systems. *IEEE Transactions on Green Communications and Networking*. 2018;2(3):612–622. DOI:10.1109/TGCN.2018.2809729
11. Yufeng S. Cross-Layer Techniques for Adaptive Video Streaming over Wireless Networks. *EURASIP Journal on Advances in Signal Processing*. 2005. DOI:10.1155/ASP.2005.220
12. Kang S.H., Zakhor A. Packet Scheduling Algorithm for wireless video streaming. *International Packet Video Workshop*. 2002.
13. Aramvith S., Lin C.W., Roy S. Sunet M.T. Wireless video transport using conditional retransmission and low-delay interleaving. *IEEE Transactions on Circuits and Systems for Video Technology*. 2002;12(6):558–565. DOI:10.1109/TCSVT.2002.800326
14. Lamri M.A., Abilov A., Vasiliev D., Kaisina I., Nistyuk A. Application Layer ARQ Algorithm for Real-Time Multi-Source Data Streaming in UAV Networks. *Sensors*. 2021;21(17):5763. DOI:10.3390/s21175763
15. Zhai F., Eisenberg Y., Pappas T.N., Berry R., Katsaggelos A.K. Joint source-channel coding and power allocation for energy efficient wireless communications. *Proceedings of the 41st Allerton Conference on Communication, Control and Computing, 1–3 October 2003, Monticello, USA*. 2003.
16. Kondi L.P., Ishtiaq F., Katsaggelos A.K. Joint source channel coding for motion-compensated DCT-based SNR scalable video. *IEEE Transactions on Image Processing*. 2002;11(9):1043–1052. DOI:10.1109/TIP.2002.802507
17. Paxson V., Allman M., Chu J., Sargent M. Computing TCP's Retransmission Timer. *RFC 6298*. 2011. DOI:10.17487/RFC6298
18. Hasslinger G., Hohlfeld O. The Gilbert-Elliott Model for Packet Loss in Real Time Services on the Internet. *Proceedings of the 14th GI/ITG Conference: Measurement, Modeling and Evaluation of Computer and Communication Systems, 31 March 2008–02 April 2008, Dortmund, Germany*. VDE; 2008.
19. Rondeau E., Lepage F., Georges J. P., Morel G. Chapter 3 – Measurements and Sustainability. In: *Dastbaz M., Pattinson C., Akhgar B. (ed.) Green Information Technology. A Sustainable Approach: Measurements and Sustainability*. Elsevier; 2015. p.29–59. DOI:10.1016/B978-0-12-801379-3.00003-6
20. Hofmann U., Pfeiffenberger T., Hechenleitner B. One-way-delay measurements with CM toolset. *Proceedings of the International Performance, Computing, and Communications Conference, 05–08 February 2000, Phoenix, USA*. IEEE; 2000. p.41–47. DOI:10.1109/PCCC.2000.830300

Список источников




1. Ngo H.A., Hanzo L. Hybrid Automatic-Repeat-reQuest Systems for Cooperative Wireless Communications // *IEEE Communications Surveys and Tutorials*. 2013. Vol. 16. Iss. 1. PP. 25–45. DOI:10.1109/SURV.2013.071913.00073
2. He X., Kumar R., Mu L., Gjøsaeter T., Li F.Y. Formal verification of a Cooperative Automatic Repeat reQuest MAC protocol // *Computer Standards and Interfaces*. 2012. Vol. 34. Iss. 4. PP. 343–354. DOI:10.1016/j.csi.2011.12.001
3. Jamshidi A. Efficient cooperative ARQ protocols based on relay selection in underwater acoustic communication sensor networks // *Wireless Networks*. 2019. Vol. 25. PP. 4815–4827. DOI:10.1007/s11276-018-1773-5
4. Goel J., Jagadeesh H. Listen to Others' Failures: Cooperative ARQ Schemes for Low-Latency Communication Over Multi-Hop Networks // *IEEE Transactions on Wireless Communications*. 2021. Vol. 20. Iss. 9. PP. 6049–6063. DOI:10.1109/TWC.2021.3071504
5. Tutgun R., Aktas E. A Markovian Analysis of Cooperative ARQ with Random Access // *Wireless Personal Communications*. 2022. Vol. 123. PP. 3201–3211. DOI:10.1007/s11277-021-09282-6
6. Goel J., Harshan J. Minimal Overhead ARQ Sharing Strategies for URLLC in Multi-Hop Networks // *Proceedings of the 93rd Vehicular Technology Conference (Helsinki, Finland, 25–28 April 2021)*. IEEE, 2021. DOI:10.1109/VTC2021-Spring51267.2021.9448948
7. Mheich Z., Savin V. Cooperative communication protocols with energy harvesting relays // *Proceedings of the Wireless Days (Porto, Portugal, 29–31 March 2017)*. IEEE, 2017. PP. 60–65. DOI:10.1109/WD.2017.7918116
8. Kim S., Kim B.S., Kim K.H., Kim K.I. Opportunistic Multipath Routing in Long-Hop Wireless Sensor Networks // *Sensors*. 2019. Vol. 19. Iss. 19. P. 4072. DOI:10.3390/s19194072
9. He X., Li F.Y. A Multi-Relay Cooperative Automatic Repeat Request Protocol in Wireless Networks // *Proceedings of the International Conference on Communications (Cape Town, South Africa, 23–27 May 2010)*. IEEE, 2010. DOI:10.1109/ICC.2010.5502169
10. Shafique T., Abdelhady A.M., Amin O., Alouini M. S. Energy Efficiency, Spectral Efficiency and Delay Analysis for Selective ARQ Multichannel Systems // *IEEE Transactions on Green Communications and Networking*. 2018. Vol. 2. Iss. 3. PP. 612–622. DOI:10.1109/TGCN.2018.2809729
11. Yufeng S. Cross-Layer Techniques for Adaptive Video Streaming over Wireless Networks // *EURASIP Journal on Advances in Signal Processing*. 2005. DOI:10.1155/ASP.2005.220
12. Kang S.H., Zakhor A. Packet scheduling algorithm for wireless video streaming // *International Packet Video Workshop*. 2002.
13. Aramvith S., Lin C.W., Roy S. Sunet M.T. Wireless video transport using conditional retransmission and low-delay interleaving // *IEEE Transactions on Circuits and Systems for Video Technology*. 2002. Vol. 12. Iss. 6. PP. 558–565. DOI:10.1109/TCSVT.2002.800326

14. Lamri M.A., Abilov A., Vasiliev D., Kaisina I., Nistyuk A. Application Layer ARQ Algorithm for Real-Time Multi-Source Data Streaming in UAV Networks // *Sensors*. 2021. Vol. 21. Iss. 17. P. 5763. DOI:10.3390/s21175763
15. Zhai F., Eisenberg Y., Pappas T.N., Berry R., Katsaggelos A.K. Joint source-channel coding and power allocation for energy efficient wireless communications // *Proceedings of the 41st Allerton Conference on Communication, Control and Computing*, (Monticello, USA, 1–3 October 2003). 2003.
16. Kondi L.P., Ishtiaq F., Katsaggelos A.K. Joint source channel coding for motion-compensated DCT-based SNR scalable video // *IEEE Transactions on Image Processing*. 2002. Vol. 11. Iss. 9. PP. 1043–1052. DOI:10.1109/TIP.2002.802507
17. Paxson V., Allman M., Chu J., Sargent M. Computing TCP's Retransmission Timer // RFC 6298. 2011. DOI:10.17487/RFC6298
18. Hasslinger G., Hohlfeld O. The Gilbert-Elliott Model for Packet Loss in Real Time Services on the Internet // *Proceedings of the 14th GI/ITG Conference: Measurement, Modeling and Evaluation of Computer and Communication Systems* (Dortmund, Germany, 31 March–02 April 2008). VDE, 2008.
19. Rondeau E., Lepage F., Georges J. P., Morel G. Chapter 3 – Measurements and Sustainability // In: Dastbaz M., Pattinson C., Akhgar B. (ed.) *Green Information Technology. A Sustainable Approach: Measurements and Sustainability*. Elsevier, 2015. PP. 29–59. DOI:10.1016/B978-0-12-801379-3.00003-6
20. Hofmann U., Pfeiffenberger T., Hechenleitner B. One-way-delay measurements with CM toolset // *Proceedings of the International Performance, Computing, and Communications Conference* (Phoenix, USA, 05–08 February 2000). IEEE, 2000. PP. 41–47. DOI:10.1109/PCCC.2000.830300

Статья поступила в редакцию 12.06.2023; одобрена после рецензирования 03.07.2023; принята к публикации 04.07.2023.

The article was submitted 13.06.2023; approved after reviewing 03.07.2023; accepted for publication 04.07.2023.

Информация об авторе:

ЛАМРИ Мохаммед Амин	аспирант кафедры «Радиотехника» Приборостроительного факультета Ижевского государственного технического университета им. М.Т. Калашникова  https://orcid.org/0000-0001-7226-7087
АБИЛОВ Альберт Винерович	кандидат технических наук, первый проректор-проректор по учебной работе Санкт-Петербургского государственного университета телекоммуникаций им. проф. М.А. Бонч-Бруевича  https://orcid.org/0000-0003-2358-4478
ПРЕСНЕЦОВ Александр Михайлович	программист ООО «Нефтемаш»  https://orcid.org/0009-0009-1453-4082