Transport Protocol to support Congestion Control and Controlled Reliability for Multimedia Transmission over Wireless Sensor Network

By:

Bansi N. Kotecha

10MCES02

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Nirma University

AHMEDABAD

May 2013

Transport Protocol to support Congestion Control and Controlled Reliability for Multimedia Transmission over Wireless Sensor Network

Major Project

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Computer Science & Engineering

By:

Bansi N. Kotecha

(10MCES02)

Guided by:

Prof. Vijay Ukani

Nirma University, Ahmedabad

May 2013

Undertaking for Originality of the Work

I, **Bansi N. Kotecha**, Roll No. **10MCES02**, give undertaking that the Major Project entitled as **"Transport Protocol to support Congestion Control and Controlled Reliability for Multimedia Transmission over Wireless Sensor Network"** submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science and Engineering** of Nirma University, Ahmedabad, is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Bansi N. Kotecha

Date:

Place:

Endorsed By:

Prof. Vijay Ukani

Certificate

This is to certify that the Thesis entitled **"Transport Protocol to support Congestion Control and Controlled Reliability for Multimedia Transmission over Wireless Sensor Network"** submitted by **Bansi N. Kotecha (Roll No : 10MCES02)**, towards the partial fulfillment of the requirements for degree of Master of Technology in Computer Science and Engineering from Institute of Technology, Nirma University, Ahmedabad is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this thesis, to the best of my knowledge, haven't been submitted to any other University or Institution for award of any degree.

Prof. Vijay Ukani
Guide, Associate Professor and PG Coordinator,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Sanjay Garg
Professor and Head,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. K. Kotecha
Director,
Institute of Technology,
Nirma University, Ahmedabad.

Acknowledgements

'Step after step, the ladder is ascended' and I am acutely indebted to **Prof. Vijay Ukani**, Associate Professor and PG - CSE Coordinator, Department of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad and my guide, for the valuable guidance, inspiration and constructive suggestions that helped me through out my Major Project. He has been my true mentor and strength in every context and the success behind this project could not have even been conceived without his support.

I am grateful to **Dr.Ketan Kotecha**, Hon'ble Director and **Dr. Sanjay Garg**, Hon'ble Head of Department, Institute of Technology, Nirma University, Ahmedabad for helping me throughout the project by their unmentionable support and providing basic infrastructure and healthy research environment.

I would also thank my faculty members and my colleagues for supporting me in one or the other different ways.

This project would not have been possible without the positive reception and constant support and sacrifice of my Parents. Most importantly, I would like to thank God, for staying at my back in all the phases of my life.

- Bansi N. Kotecha

10MCES02

Abstract

Wireless Multimedia Sensor Network (WMSN) is a network of wirelessly interconnected sensor nodes equipped with multimedia devices, such as cameras and microphones, and capable to retrieve video and audio streams, still images, as well as scalar sensor data. There are different challenges at each layer of the protocol stack amongst which the major discussion is focused on transport protocols for WMSN which are classified into three categories viz. Reliability Support, Multipath Support and Congestion Control.

Here, the focus is on Congestion Control Transport Protocols which are Standard Protocols like TCP and UDP or Application Specific Protocols like STCP, CODA, CCF, DPCC, UDDP, DCCP, etc. each of which depend on Node level or Link level congestion control mechanisms.

Most of the existing transport protocols for WSN do not take into consideration the multimedia requirements in WMSN. Moreover, majority of the existing protocols do not provide congestion control and reliability simultaneously, i.e. either of them is implemented in the existing protocols.

Therefore, there is a necessity of developing Transport Protocol that gives better performance as compared to the existing ones to control the congestion and provide reliability both, considering the multimedia requirements in WMSN.

The proposed algorithm focuses on providing Controlled Reliability and Congestion Control both at the same time which is the core research area now a days.

Keywords: Wireless Multimedia Sensor Network, Congestion Control, Controlled Reliability, Datagram Congestion Control Protocol

Contents

List of Figures

List of Tables

Abbreviations

WSN	Wireless Sensor Network
WMSN	Wireless Multimedia Sensor Network
I-frame	Intra-coded frame
P-frame	Predicted frame
B-frame	Bi- directionally predicted frame
GOP	Group of Pictures
QOS	Quality of service
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
ECN	Explicit Congestion Notification
PSFQ	Pump Slowly Fetch Quickly
RMST	Reliable Multi-Segment Transport
MRTP	Multi-flow Real-time Transport Protocol
STCP	Sensor Transmission Control Protocol
CODA	COngestion Detection and Avoidance
CCF	Congestion Control and Fairness
DPCC	Dynamic Priority Based Congestion Control
UDDP	A User Datagram Dispatcher Protocol
TFRC	TCP Friendly Rate Control
SCTP	Stream Control Transport Protocol
ROI	Region of Interest
NS	Network Simulator
RA	Rate Adaptive
RTP	Reliable Transport Protocol
RSTP	Reliable Synchronous Transport Protocol
JPEG	Joint Photography Expert Group
PCCP	Priority Based Congestion Control Protocol
QCCP-PS	Queue based Congestion Control Protocol with Priority Support
CDU	Congestion Detection Unit
CNU	Congestion Notification Unit
RAU	Rate Adjustment Unit
RTT	Round Trip Time
TO	Time Out
DCCP	Datagram Congestion Control Protocol
DCCP-CR	Datagram Congestion Control Protocol with Controlled Reliability

Chapter 1

Introduction

Wireless sensor networks (WSN) have drawn the attention of the research community in the last few years, which are the applications enabled by large-scale networks of small devices capable of gathering information from the physical environment phenomena like temperature, pressure, humidity, or location of objects, processing it and transmitting it to remote locations. The speedy development and progress of sensors, MEMS, embedded computing, CMOS (Complementary Metal Oxide Semiconductor) cameras and microphones, signal processing and multimedia source coding techniques, allowed for the emerge of so called Multimedia Sensor Network (WMSN) that has shifted the focus from the typical scalar wireless sensor networks to networks with multimedia devices that are capable to retrieve video, audio, images, as well as scalar sensor data.

As a result, Wireless Multimedia Sensor Network (WMSN) [1] is a network of wirelessly interconnected sensor nodes equipped with multimedia devices, such as cameras and microphones, and capable to retrieve video and audio streams, still images, as well as scalar sensor data.

WMSNs [2] promise a wide range of potential applications in both civilian and military areas which require visual and audio information like Multimedia surveillance sensor networks, Storage of potentially relevant activities like thefts, car accidents, traffic violations and make video/audio streams or reports available for future query, Traffic avoidance, enforcement and control systems, Advanced health care delivery, Environmental and habitat monitoring, Person locator services, Industrial process control, etc.

1.1 Objective

The main objective of the project is to optimize existing Transport Protocol for Wireless Multimedia Sensor Network to provide Congestion Control and Controlled Reliability both, so as to achieve better performance as compared to the existing ones that either support Congestion Control or Reliability.

1.2 Scope

In WMSN, the events are generally event driven and not continuous and so there are more chances of Congestion occurrence. Moreover, dropping of Region of Interest (ROI) packets may cause big loss in the multimedia content. Hence, the scope of the project is to optimize a transport protocol for providing congestion control and reliability both simultaneously in wireless multimedia sensor network along with providing simultaneous streams of data to be sent and Sequenced delivery of packets.

1.3 Motivation of the work

Earlier, sensor network was used to measure scalar physical phenomena, such as temperature, pressure, humidity, or location of objects, etc. But recently, Wireless Multimedia Sensor Network is in higher demand as WMSN is a network of wirelessly interconnected sensor nodes equipped with multimedia devices, such as cameras and microphones, and capable to retrieve video and audio streams, still images, as well as scalar sensor data.

WMSNs have also additional characteristics and challenges, in addition to those of WSNs, because of the nature of the real time multimedia data such as high bandwidth demand, real-time delivery, tolerable end-to-end delay, and proper jitter, frame loss rate, etc. Moreover, there are many different resource constraints in WMSNs involving energy, bandwidth, data rate, memory, buffer size and processing capability because of the physically small size of the sensors and the nature of the multimedia application that is typically producing a huge amount of data [3] [4].

Congestion control is one of the services done by transport layer protocols to mitigate congestion in the network. Congestion, in wireless sensor networks, not only wastes the scarce energy due to a large number of retransmissions and packet drops, but also hinders the event detection reliability and link utilization.

Packets sent to the sink are highly compressed at the source. Compression standards such as the JPEG2000 and the MPEG introduce features such as the region of interest (ROI) and the inter and intra-frames. These special packets carry original content that cannot be retrieved through interpolation. Hence, dropping packets indiscriminately, may cause discernible disruptions in the multimedia content. Thus, we argue that some form of selective reliability, must be introduced for these packets in a WMSN.

Most of the existing transport protocols for WSN do not take into consideration the multimedia requirements in WMSN. Moreover, majority of the existing protocols do not provide congestion control and reliability simultaneously, i.e. either of them is implemented in the existing protocols. Therefore, there is a necessity of developing Transport Protocol that gives better performance as compared to the existing ones to control the congestion and provide reliability both, considering the multimedia requirements in WMSN.

Chapter 2

Literature Survey

2.1 Wireless Multimedia Sensor Network Architecture

The available resources in the network can be efficiently utilized and distributed throughout the network to perform the desired operations of the multimedia content according to the following three reference models of Network architectures in WMSNs as shown in figure 2.1:

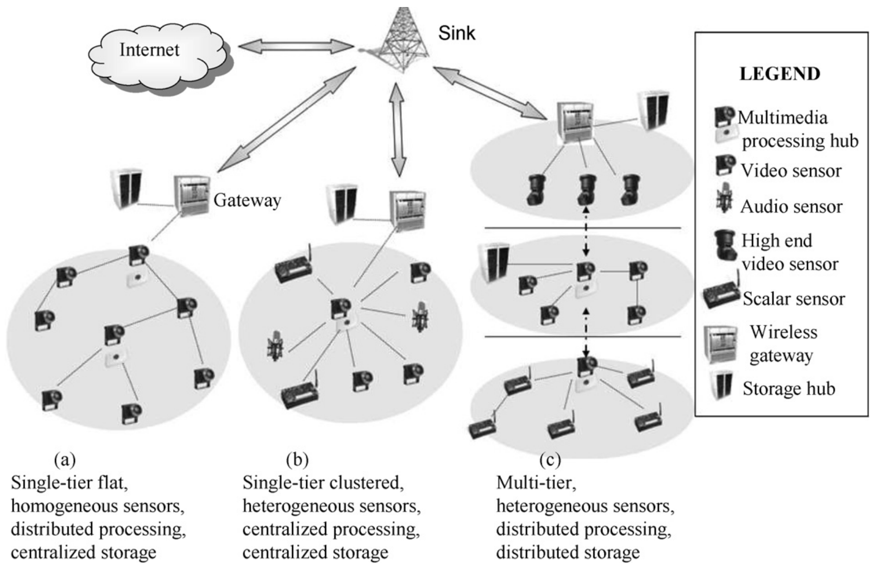

Internet
Sink
Gateway
LEGEND
Multimedia processing hub
Video sensor
Audio sensor
High end video sensor
Scalar sensor
Wireless gateway
Storage hub
(a)
Single-tier flat, homogeneous sensors, distributed processing, centralized storage
(b)
Single-tier clustered, heterogeneous sensors, centralized processing, centralized storage
(c)
Multi-tier, heterogeneous sensors, distributed processing, distributed storage

Figure 2.1: Reference architecture of WMSN [5]

a. Single-tier flat architecture: It consists of homogeneous sensor nodes to perform multimedia functionalities which are easily manageable and multimedia processing is distributed among the nodes, which prolongs network life time [5].

b. Single-tier clustered architecture: It consists of heterogeneous sensors where camera, audio and scalar sensors within each cluster relay data to a cluster head (head is wirelessly connected with the sink or the gateway either directly or through other cluster heads in multi-hop fashion)which has more resources and is able to perform intensive data processing. [5]

c. Multi-tier architecture: It consists of heterogeneous sensors [6] , which provides better balancing of coverage, cost, functionalities, reliability, etc. where the first tier deployed with scalar sensors performs simple tasks, like motion detection, the second tier of camera sensors may perform more complicated tasks as object detection or object recognition, and at the third tier more powerful and high resolution camera sensors to perform more complex tasks, like object tracking. Each tier, connected wirelessly with the sink or the gateway, may have a central hub to perform more data processing and communicate with the higher tier [5].

2.2 Factors Influencing The Design Of Multimedia Sensor Networks

The following are several factors that influence the design of a WMSN:

- Resource Constraints
- Variable Channel Capacity
- Cross-layer Coupling of Functionality
- Application-specific QoS Requirements
- High Bandwidth Demand
- Multimedia Source Coding Techniques
- Multimedia In-network Processing[1]
- Power consumption
- Flexible architecture to support heterogeneous applications

- Multimedia coverage
- Integration with Internet (IP) architecture
- Integration with other wireless technologies[7].

2.3 Internal Organization Of Multimedia Sensor

A multimedia sensor device may be composed of several basic components as shown in the figure 2.2. Sensing units usually are composed of sensors (cameras, microphones, and/or scalar sensors) and analog-to-digital converters (ADCs) which converts the analog signals produced by the sensors to digital signals and is then fed into the processing unit.

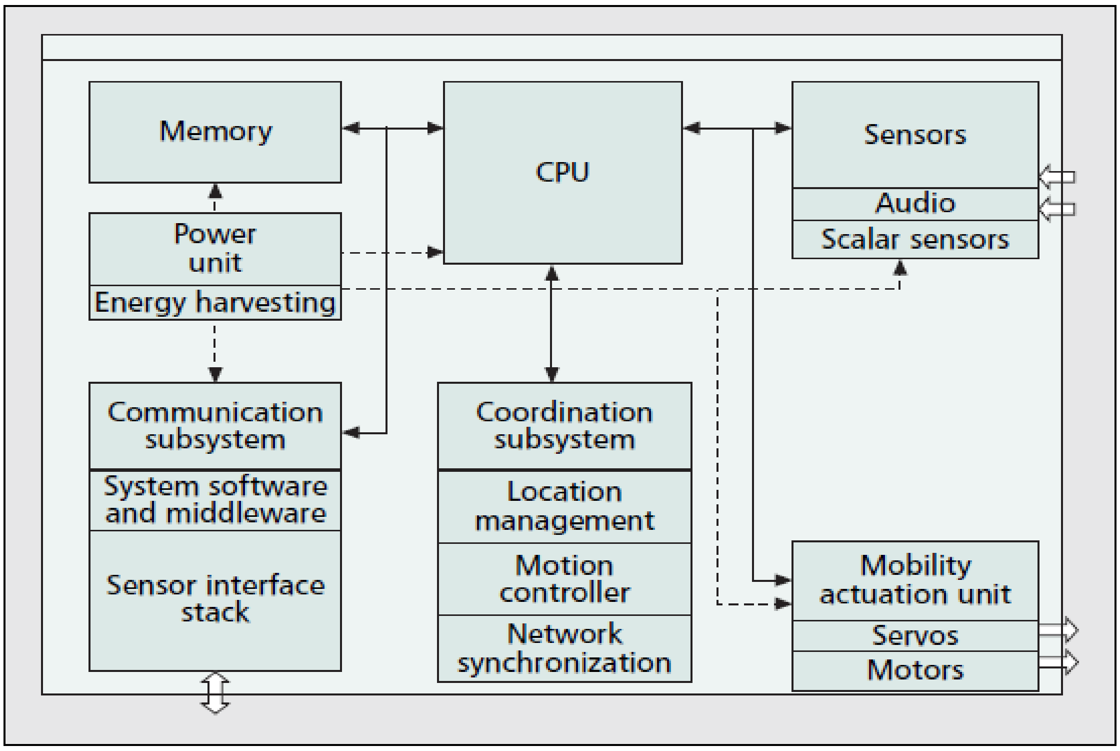


Figure 2.2: Internal Organization Of Multimedia Sensor[7]

The processing unit executes the system software of coordination subsystem that performs operations such as network synchronization and location management. A communication subsystem; that is interfaced with a storage unit includes communication protocol stack and system software, such as middleware, operating systems, virtual machines, etc. An optional mobility/actuation unit can enable movement or manipulation of objects. Finally, the whole system is powered by a power unit that may be supported by an energy scavenging unit, such as solar cells.

2.4 Characteristics Of Real Time Multimedia Communication

There are various characteristics required for the multimedia communication which are as follows[8]:

- Jitter - that is introduced in real-time data by the delay between packets
- Timestamp the packets - to prevent jitter so as to separate the arrival time from the playback time that requires a playback buffer to store the data until they are played back
- Ordering -requires Sequence No. so as to recover from packet loss
- Support for Multicasting - to distribute data in case of applications like video conferencing, etc.
- Translation - changing the encoding of a payload to a lower quality to match the bandwidth of the receiving network
- Mixing - several streams of traffic is combined into one stream

2.5 Challenges At Different Layers Of Protocol Stack

There are various research challenges that are faced at different layers of the communication protocol stack. Some of them are outlined in detail in Table 2.1. I shall be focusing on Transport Layer in this report.

Table 2.1: Research Challenges at different layers of Protocol Stack

LAYERS OF PROTOCOL STACK	**WMSN FUNCTIONALITIES**
Application Layer	Advanced Multimedia Encoding Techniques [7] to give High compression efficiency, Low complexity, Error resiliency, etc. Advanced Multimedia In-networking Processing [1] Traffic Management [1]
Transport Layer	Congestion Control [7] Reliability [1] Guaranteed delivery of I-frames [7] In-order delivery [5] Loss recovery [5] Timeliness [5] Fairness [5]
Network Layer	Energy Optimization [7] Link Quality [7] Multipath and Fault Tolerance [5] Addressing and Localization [7] Routing based on Network Conditions [5],like Position with respect to sink, Radio Characteristics, Error Rate, Residual energy, Backlogged Packets, [1] etc. Routing based on Traffic Classes [or Services] like Event-driven service, Data query service or Stream query service, Dropping rate, Latency, Desired bandwidth, etc. [1] Routing based on support for Streaming like Spatio-temporal character, Probabilistic delay guarantees, etc. [5]
MAC Layer	Reliable, Error-free data transfer with Minimum Retransmissions [1] Maximizing The Network Throughput [7] Enhance Transmission Reliability [1] Minimize Control Overhead [1] Channel Access, Scheduling and Admission Control, Error Control [5] Contention Free Protocols with Single Channel for TDMA, Better control for multimedia parameters, MIMO technology, etc. [1] Contention Free Protocols with Multi- Channel for Better Bandwidth Utilization, etc. [1] Contention Based Protocols to coordinate sleep/wake cycles, bursty nature of scheduling leading to jitters, etc [1]
Physical Layer	Provide good gain over noise [5] Use of time - hopping impulse radio Ultra Wide Band [7] Inference resistance [7] Compatibility with higher layers in the protocol stack [5]

2.6 Transport Layer

2.6.1 Introduction to Transport Layer

Transport layer is a group of protocols that run over the network layer to enable end-to-end message transmission. Transport layer aims to provide several services such as: flow and congestion control, and possibly QoS requirements (e.g., fairness and timing) [5]. The main area of concern in Wireless Multimedia Sensor Network for Transport Protocols would be in-order delivery, loss recovery [5], reliability, Timeliness, Congestion Control [1], etc.

2.6.2 Generic Structure Of Transport Protocol For WMSN

Figure 2.3 shows the Generic Structure of Transport Protocol for WMSN [9], which consists of three main functional modules: (i) congestion control module, (ii) reliability module, and (iii) priority module. Congestion module prevents congestion, thereby reducing the packet drops thus resulting in increased throughput. The reliability module ensures the successful delivery of each segment to the ultimate destination. The priority module differentiates the source traffic based on the importance of the application.

2.6.3 Classification Of Existing Transport Protocols For WSN

The following discussion is based on the classification of WMSN Transport Protocols [5] into (1) Standard Protocols i.e. TCP and UDP (2) Application-specific and non-standardized protocols based on Reliability and Congestion Control.

Standard Transport Protocols:

For real-time applications like streaming media, the User Datagram Protocol (UDP) is preferred over TCP (Transmission Control Protocol) as timeliness is of greater concern than reliability. However, in WMSNs, neither TCP nor UDP can be used due to the following reasons:

a. TCP (Transmission Control Protocol):

- Doesn't support time-stamping and multicasting

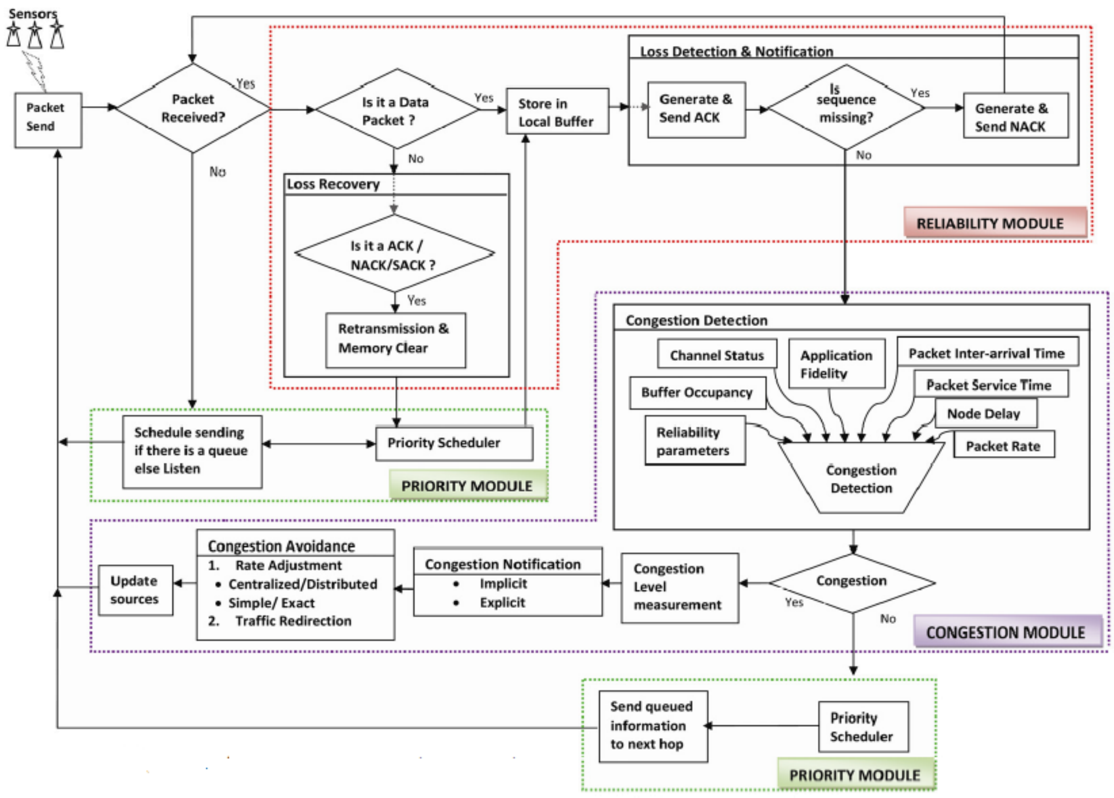


Figure 2.3: Generic Structure of Transport Protocol for WMSN [9]

- Large TCP Header Size
- In case if a packet is lost, TCP gives us a provision of retransmission of a lost or corrupted packet, which cannot be allowed in multimedia as it can be ignored [8]
- Effect of jitter induced by TCP
- Overhead of the connection establishment mechanism in TCP might not be suitable for event-driven applications [9]
- The reliability mechanism in TCP is also based on end-to-end retransmission which consumes more energy and bandwidth than hop-by-hop retransmission [9]
- Distinguish between bad channel conditions and network congestion. : When a packet loss is detected, TCP assumes that it is due to congestion only, hence triggering the rate adjustment process to reduce it which is not always true [9]
- Fairness is an issue in TCP, because congestion control mechanism in TCP can discriminate against sensor nodes that are far away from the sink node[9]

b. UDP (User Datagram Protocol):

   - It is more suitable as compared to TCP as is supports multicasting and has no retransmission policy. Still it is not suitable for multimedia traffic as:
     - It doesn't provide time-stamping, sequencing or mixing [8]
     - It drops the packets during congestion, which may cause loss of i-packets also [9]
     - Does not have provision in header to allow heterogeneous traffic [9]

Application Specific Transport Protocols:

The existing Transport Protocols for WSN can be categorized as protocols for Reliability Support [10], Use of Multi-path and Congestion Control [1].

a. Reliability Support: When a multimedia packet is transmitted, the arrival of I-frame should be guaranteed though an application can withstand moderate loss for P-frame and B-frames. So per-packet reliability has to be enforced by separating the I-frames and letting the transport protocol to ensure their guaranteed delivery to the sink. Such transport protocols include RMST (Reliable Multi-Segment Transport), PSFQ (Pump Slowly Fetch Quickly), etc. which buffers the packets at intermediate nodes for faster retransmission of packets in case of packet loss [1].

b. Use of Multi-Path:
   A large burst of data can be segmented into smaller burst so as to prevent the buffers of the intermediate sensor nodes from getting overloaded. Also, it is possible that certain channels might permit high data rates. In such cases, the use of multi-path becomes a must. The transport protocols which facilitate with this support include MRTP (Multi-flow Real-time Transport Protocol), etc. that splits the packets over different flows but doesn't provide with a provision of retransmission which hence can be an issue for scalar traffic [1].

c. Congestion Control:
   Congestion control service is the solution provided by the Transport layer to mitigate the congestion which not only wastes the energy and resources due to a number of

retransmissions and packet drops but also affects the reliability. There are two types of congestion that are possible: (1) Node level congestion (2) Link level congestion [11]as shown in figure 2.4.

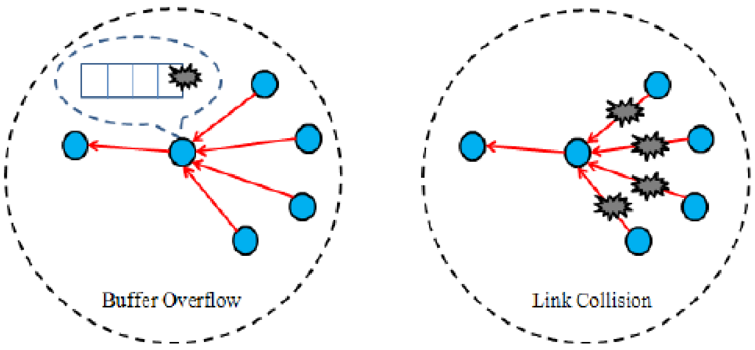


Figure 2.4: Node level and link level congestion [12]

The mechanism for Congestion Control involves three stages: (1) Congestion Detection (2) Congestion Notification (3) Congestion Avoidance.

Congestion Detection means identification of events causing congestion that includes factors like PST (Packet Service Time which is the time required to process each packet), PIAT (Packet Inter Arrival Time which the time interval between two sequentially arriving packets at each node), Node delay (which is the estimated accurate congestion degree), etc [9].

The mechanism to provide information to the relevant neighbors about the detected congestion is called Congestion Notification, which can be done Explicitly using a Control Message or Implicitly by embedding information into the normal data packet [9]. The latter is more power efficient than the former one as it avoids the overhead with the control message.

Congestion avoidance is the phenomena to avoid the congestion, which includes Rate Adjustment that can be implemented either in Centralized fashion at sink or Distributed fashion [9] at each hop of the network. The former controls the aggregate rate of the network while the latter reduces the congestion quickly. There are two types of Rate Adjustment Techniques which are Simple and Exact Rate Adjustment [9]. The former uses single congestion bit to inform it there is congestion or not. Whereas the latter additionally estimates and notifies about the congestion with the additional information like allowable data rate that should be updated in the next data transmission (like in Congestion Control and Fairness [13]), congestion degree (like in Priority-based Congestion Control Protocol), etc.

Another method of Congestion Avoidance is Traffic Redirection [9] where he protocols divert the outgoing traffic to optimal uncongested paths (like Siphon [14] , etc). Further, Polite Group Policy [9] is another method for Congestion Avoidance where each node broadcasts the summary of its data to its local neighbors periodically, but if it hears identical data from any of its neighbors; it politely suppresses down its own broadcasting (like Trickle [15]).

Moreover, there are two types of multimedia flows like Event-driven flow where the base station cannot estimate the arrival times of next packet and Continuous Flow where the base station knows the expected arrival time for the next packet.

2.7 Video Model for MPEG-4

According to [16], for encoding the original pictures in the temporal domain, there are three basic methods: Intra coded frames (I-frames), Predicted frames (P-frames) and Bi-directionally predicted frames (B-frames), introduced in the MPEG-1 standard [50], which are applied on the frame, macroblock or block level, depending on the codec. Group of Pictures (GoP) is the sequence of frames starting with an I-frame, up to but not including the next I-frame as shown in figure 2.5.

1) I-frames:

- Consists exclusively of intra-coded macroblocks
- Independently coded from the other frames

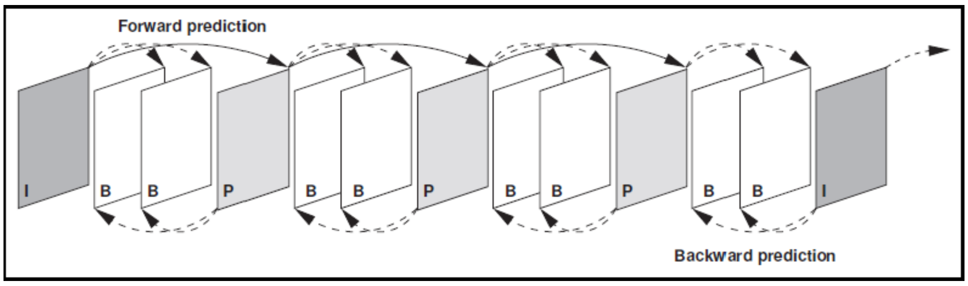


Figure 2.5: I, P and B frames [16]

- Contains the compressed image information (without any prediction information),
- Results in a large frame size (compared to the size of the inter- or bidirectional-coded frames)
- Use motion estimation and compensation techniques relying on the previous inter- or intra-coded frame.
- Do not rely on other video frames and thus are important to stop error propagation

2) P-frames:

- Rely on a previous as well as a following I-frame or P-frame.
- Results in smaller frame sizes for the P-frames as compared to I-frame
- Considers the closest time-preceding frame

3) B-frames:

- Rely on a previous as well as a following I-frame or P-frame.
- Results in smaller frame sizes for the B-frames as compared to P-frame
- When B frames do not have any following I- or P-frames that can be used as reference frames, no encoding or decoding is possible.

Chapter 3

Congestion Control Transport Protocols For WMSN

3.1 Introduction

The QoS of multimedia applications of WMSN mainly focuses on packet loss. There are various transport protocols in WMSN that are specific for controlling packet loss due to inference in wireless environment and bursty traffic. As transport layer focuses on congestion control, there different algorithms developed for the same are STCP[17], CODA [18], CCF[13], UDDP [19], etc. some of which are explained in the further discussion.

The main area of concern in Wireless Multimedia Sensor Network for Transport Protocols are in-order delivery, loss recovery [5], reliability, Timeliness, Congestion Control [1], etc.

3.2 Sensor Transmission Control Protocol (STCP)

STCP [17]is a transport protocol that implements the functionalities like flow type, transmission rate, etc. at base station. STCP adopts the method of explicit congestion notification with some modification. The header of each STCP data packet has a congestion notification bit. Every sensor node maintains two thresholds in its buffer: t-lower and t-higher. When the buffer reaches t-higher, the node will set the congestion notification bit in every packet it forwards.

When the base station receives such packet, the it informs the source of the congested path by setting the congestion bit in the acknowledgement packet. On receiving the congestion notification, the source redirect the successive packets or slow down the transmission rate.

3.3 COngestion Detection and Avoidance (CODA)

Event driven sensor networks remain idle till they detect certain event after which they become active which might cause congestion. To overcome such situations the transport protocol COngestion Detection and Avoidance (CODA) [18] provides with three mechanisms: (i) Receiver-Based Congestion Detection; (ii) Open-Loop Hop-By-Hop Backpressure; And (iii) Closed-Loop Multi-Source Regulation.

Receiver-Based Congestion Detection: as the medium is shared by various channels, state of each channel must be known by the sensor nodes. CODA uses a combination of the present and past channel loading conditions by activating local channel monitoring at the appropriate time to minimize cost and the current buffer occupancy, to detect congestion at each receiver with low cost.

Open-Loop Hop-By-Hop Back-pressure: As shown in figure 3.1 When congestion is detected by a node, it propagates Backpressure messages to the source upstream. local congestion policy (e.g., packet drop, AIMD, etc.). When an upstream node (toward the source) receives the message, it can decrease its sending rate or drop packets and it decides whether or not to further propagate the backpressure upstream, based on its own local network conditions.

Closed-loop, Multi-Source Regulation: It is capable of asserting congestion control over multiple sources from a single sink. When the source event rate is less than some fraction of the maximum theoretical throughput of the channel as shown in the equation 3.1, the source regulates itself. When this value is exceeded, the mechanism is triggered where the

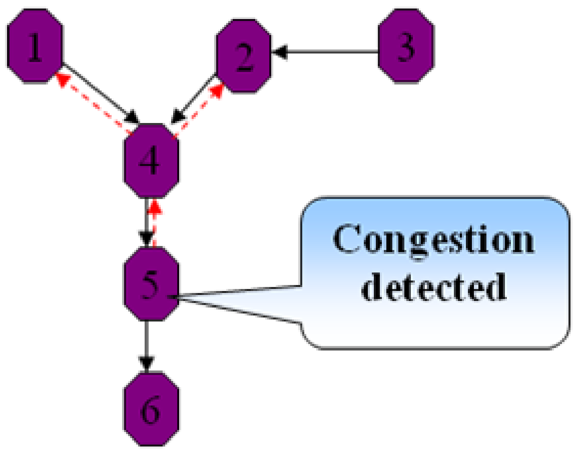


Figure 3.1: Open-Loop, Hop-By-Hop Backpressure [20]

source requires constant, slow time-scale feedback (e.g., ACK) from the sink to maintain its rate, the reception which serves as a self-clocking mechanism allowing sources to maintain their current event rates. In case of failure to receive ACKs forces a source to reduce its own rate.

$$S_{max} \approx \frac{1}{1+2\sqrt{\beta}} \text{ (for } \beta \ll 1), \text{ where } \beta = \frac{\tau C}{L} \tag{3.1}$$

Here, β is a measure of propagation delay, τ is channel idle detection delay, C is the channel bit rate and L is the expected number of bits in a data packet. When the source event rate (r) is greater than some fraction n of the maximum theoretical throughput (S_{max}) of the channel, the source triggers sink regulation by setting a regulate bit in the event packets it forwards toward the sink ,the reception of which forces the sink to send ACKs to regulate all sources associated with a particular data event. The reception of ACKs at sources would serve as a self-clocking mechanism allowing the sources to maintain the current event rate (r).

3.4 Congestion Control and Fairness (CCF)

Here (according to [13]), consider the scenario having many-to-one multihop routing which can easily be extended to unicast or many-to-many routing. Such structures face a problem where sensors closer to the base station experiencing congestion, that cause packets origi-

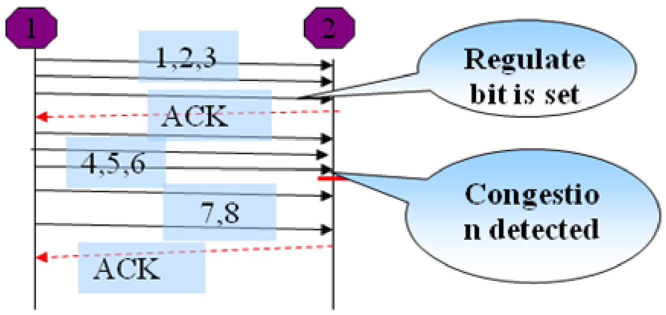


Figure 3.2: Closed-loop, Multi-Source Regulation [20]

nating from sensors and away from the base station to have a higher probability of being dropped. There are two types of congestion possible according to [13]: Type A: in a particular area, many motes within range of one another attempt to transmit simultaneously. Type B: within a particular mote, where the buffer used to hold packets to be transmitted, overflows.

The algorithm to be implemented in every mote according to [13] consists of: Measuring the average rate r at which packets can be sent from this mote, obtaining the generation rate rdata = r/no; the per-node data packet; by dividing 'r' among the number 'no' of children motes downstream and finally comparing the rate rdata with the rate r(data;parent) sent from the parent, thus propagating the smaller rate downstream.

To minimize the Type A congestion let us assume that initially the congestion occurs at a particular node, informs its downstream nodes to reduce their transmission rate which gets propagated through the network emptying the queues of this node and its neighbors resulting into less per node transmission. The node then propagates the new, increased rate resulting into congestion again. The cycle repeats and the rate fluctuates around the actual rate causing phase shift between nodes of different depths thus reducing Type A Congestion.

To minimize the type B congestion, the transmission rates of all downstream nodes can be reduced according to the generation rate when this node's queue is full or about to become full, thus allowing the queue to empty.

3.5 Dynamic Priority Based Congestion Control (DPCC)

DPCC [21] prevents congestion in WMSNs using exact rate control based on dynamic priority supporting three different types of traffic, namely, urgent type (URG) which is important real-time traffic in fact, quick type (QUI) which is real-time but not so important and normal type (NOR).

The algorithm defines Congestion Index as the ratio of average packet scheduling rate over average packet service rate in each senor node i. When the scheduling rate is smaller than the service rate in mote i, then IC (i) is smaller than 1 else greater than 1 resulting into no congestion or queuing up of packets if the packet scheduling rate that goes below the scheduling rate respectively.

Congestion Control is done by Piggybacking the Transmission Rate according to the Global Priority (importance of the total traffic at each node) of the base station and its child nodes and Reducing the same if needed.

3.6 A User Datagram Dispatcher Protocol (UDDP)

In WMSN, the sensor nodes produce snapshots or multimedia streaming which can be encoded to three different types: Intra frame (I), Predictive frame (P) and Bi-directionally predictive frame(B). Since the packet size of WMSN is too small, the video frames are fragmented to different number of packets which belong to the I-frames, P-frames and B-frames. These packets are called I-, P- and B-packets respectively as shown in figure 3.3.

According to [19], the Application layer sends the number of packets in each GOP along with the frame type to transport layer. Moreover, MAC layer sends Maximum Transmission Unit (MTU) of packet to transport layer for fragmenting each video frame to suitable number of packets. A Group of pictures (GOP) contains [8] one I-, multiple P- and B-frames. P and B- frames depend on I-frames and so are the most important among all three frame types. B-frames are least important as no other frames depend on it. Therefore I- and B-

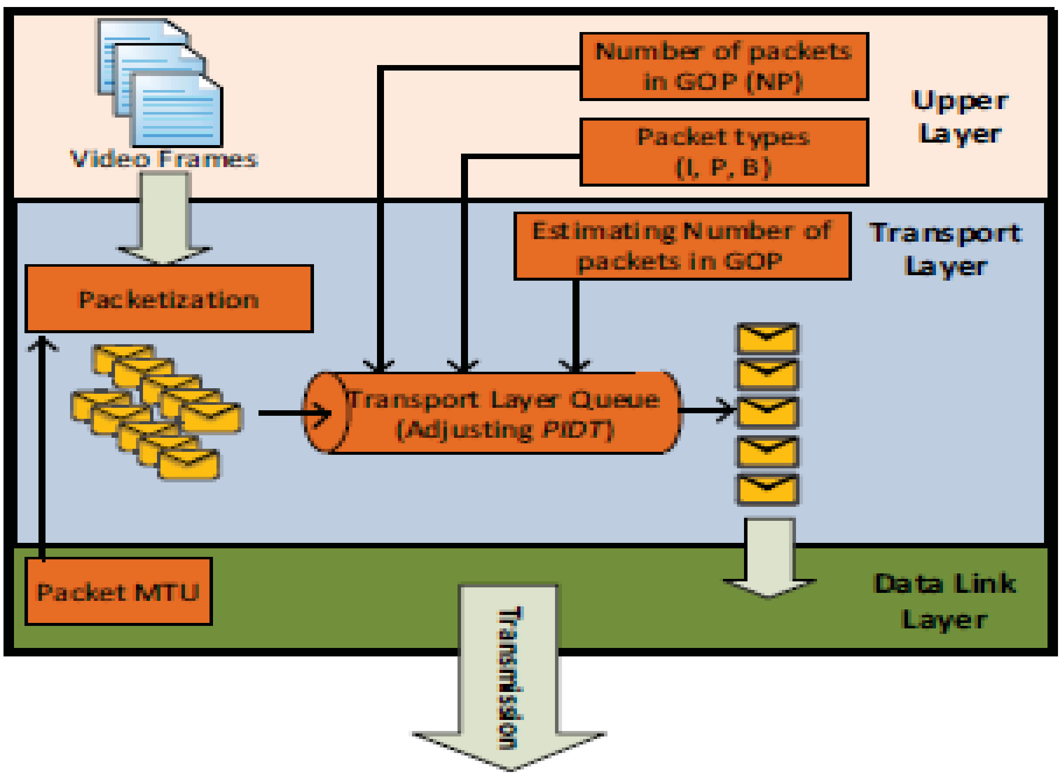


Figure 3.3: Architecture of UDDP [19]

frame loss has the most and the least effect on video quality in multimedia applications respectively. The encoded video frames are then packetized in transport layer and finally transmitted over WMSN.

$$PIDT_j = \begin{cases} \dfrac{NF_j * \frac{1}{FR}}{NP'_j}, & \text{if } OR_s > \frac{PS*NP'_j}{NF_j*\frac{1}{FR}} \quad (3.2a) \\ \dfrac{PS}{OR_s}, & \text{if } OR_s < \frac{PS*NP'_j}{NF_j*\frac{1}{FR}} \quad (3.2b) \end{cases}$$

$$IP_j = NP_j - \frac{OR_s * NF_j * \frac{1}{FR}}{PS} \qquad (3.3)$$

As per the pseudo code stated in [19], for all the GOPs of node i, predict the number of packets in GOP j, find Packet Inter-Departure Time in GOP j which is the inter-departure time of two consecutive packets in transport layer according to equation 3.2, ignoring the packets IP_j with lower priority in GOP j by equation 3.3 and finally sending the packets in GOP j with $PIDT_j$ interval.

In the equations (2) and (3), $PIDT_j$ is packet inter-departure time in GOP j, NP'_j is the number of predicted packets in GOP j, FR is the frame rate of video (fps), NF_j is the number of frames in GOP j, PS is the packet size and OR_s is the maximum output rate of the source s. (PS * NP'_j)/ (NF_j * (1/FR)) shows the predicted sending rate of the sources.

When the predicted output rate is bigger than the maximum output rate, some of the frames in GOP with lower priority (like B-packets and P-packets that are close to the end of GOP) are ignored and not sent by source node to adjust sending rate.

3.7 Datagram Congestion Control Protocol (DCCP)

Datagram Congestion Control Protocol (DCCP)[22] implements a congestion-controlled, unreliable flow of datagrams suitable for use by applications such as streaming media. DCCP provides the following features:

- An unreliable flow of datagrams, with acknowledgements.
- A reliable handshake for connection setup and tear-down.
- Reliable negotiation of features.
- A choice of TCP-friendly congestion control mechanisms, including, initially, TCP-like congestion control (CCID 2) and TCP-Friendly Rate Control(CCID 3). CCID 2 uses a version of TCP's congestion control mechanisms, and is appropriate for flows that want to quickly take advantage of available bandwidth, and can cope with quickly changing send rates; CCID 3 is appropriate for flows that require a steadier send rate.
- Options that tell the sender, with high reliability, which packets reached the receiver, and whether those packets were ECN marked, corrupted, or dropped in the receive buffer.

- Congestion control incorporating Explicit Congestion Notification (ECN) and the ECN Nonce.
- Mechanisms allowing a server to avoid holding any state for unacknowledged connection attempts or already-finished connections.
- Path MTU discovery.

3.7.1 Differences between DCCP and TCP

- Packet stream. DCCP is a packet stream protocol, not a byte stream protocol. The application is responsible for framing.
- Unreliability. DCCP will never retransmit a datagram. Options are retransmitted as required to make feature negotiation and ack information reliable.
- Packet sequence numbers. Sequence numbers refer to packets, not bytes. Every packet sent by a DCCP endpoint gets a new sequence number, even including pure acknowledgements. This lets a DCCP receiver detect lost acks, but introduces some complications with endpoitns getting out of sync; see Sequence Number Validity in [DCCP].
- Copious space for options (up to 1020 bytes).
- Feature negotiation. This is a generic mechanism by which endpoints can agree on the values of "features", or properties of the connection.
- Choice of congestion control. One such feature is the congestion control mechanism to use for the connection. In fact, the two endpoints can use different congestion control mechanisms for their data packets: In an A $\leftrightarrow B$ connection, data packets sent from A $\rightarrow$ B can use CCID 2, and data packets sent from B $\rightarrow$ A can use CCID 3.
- Different acknowledgement formats. The CCID for a connection determines how much ack information needs to be transmitted. In CCID 2 (TCP-like), this is about one ack per 2 packets, and each ack must declare exactly which packets were received (Ack Vector option); in CCID 3 (TFRC), it's about one ack per RTT, and acks must declare at minimum just the lengths of recent loss intervals.

- No receive window. DCCP is a congestion control protocol, not a flow control protocol.
- Distinguishing different kinds of loss. A Data Dropped option lets one endpoint declare that a packet was dropped because of corruption, because of receive buffer overflow, and so on. This facilitates research into more appropriate rate-control responses for these non-network-congestion losses (although currently all losses will cause a congestion response).
- Definition of acknowledgement. In TCP, a packet is acknowledged only when the data is queued for delivery to the application. This does not make sense in DCCP, where an application might request a drop-from-front receive buffer, for example. We acknowledge a packet when its options have been processed. The Data Dropped option may later say that the packet's payload was discarded.
- Integrated support for mobility.
- No simultaneous open.

3.7.2 DCCP Packets

Currently, ten packet types implement DCCP's protocol functions, referring to Table 3.1, the first eight packet types occur during the progress of a typical connection, and the two remaining packet types are used to resynchronize after bursts of loss.

3.7.3 Congestion Control ID

A mechanism, known as Congestion Control Identification (CCID), is implemented in DCCP, enabling it to assign separate CCID for each direction of data flow [23]. CCID defines the nature of congestion control mechanism and the selection of appropriate mechanism by the source and destination is achieved by the so called feature-negotiation.

The connection establishment is done as under:

- Client is in closed state and server is in listening state.

Table 3.1: DCCP Packets

Name	**Use**
DCCP-Request	Sent by the client to initiate a connection (the first part of the three-way initiation handshake).
DCCP-Response	Sent by the server in response to a DCCP-Request (the second part of the three-way initiation handshake).
DCCP-Data	Used to transmit application data
DCCP-Ack	Used to transmit pure acknowledgements.
DCCP-DataAck	Used to transmit application data with piggybacked acknowledgements.
DCCP-CloseReq	Sent by the server to request that the client close the connection.
DCCP-Close	Used by the client or the server to close the connection; elicits a DCCP-Reset in response.
DCCP-Reset	Used to terminate the connection, either normally or abnormally.
DCCP-Sync	Used to resynchronize sequence numbers after large bursts of loss.
DCCP-SyncAck	Used to resynchronize sequence numbers after large bursts of loss.

- Client sends DCCP request, which specifies server and client ports, and server sends DCCP response to specific client, which means the willingness of server to exchange messages.
- Client sends DCCP acknowledgement to server to inform that DCCP response is received.
- Server and client then exchange DCCP-Data, DCCP-Ack and DCCP-DataAck packets, which includes piggybacked acknowledgement.
- Server sends DCCP-CloseReq to client for requesting to close the connection.
- Client acknowledges the request by sending DCCP-Close packet. Server then sends DCCPReset packet and clears its connection state.
- Client receives DCCP-Reset packet and holds the time wait state for two maximum segment lifetimes to allow on transit packets to clear the network.

DCCP connections are bidirectional: data may pass from either endpoint to the other.

This means that data and acknowledgements may flow in both directions simultaneously. Logically, however, a DCCP connection consists of two separate unidirectional connections, called half-connections. Each half-connection consists of the application data sent by one endpoint and the corresponding acknowledgements sent by the other endpoint. We can illustrate this as shown in figure 3.4:

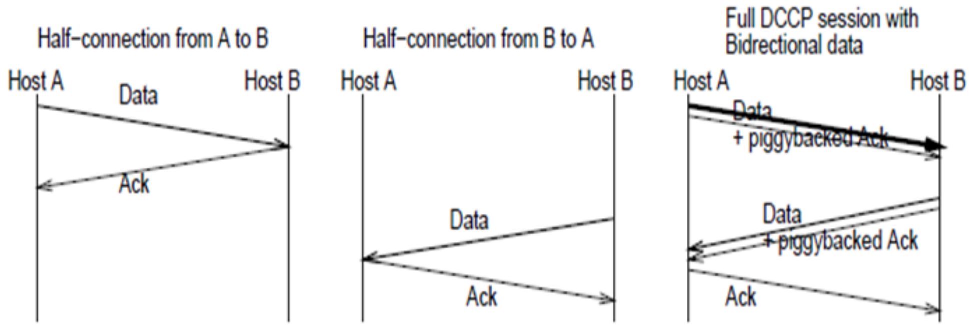


Figure 3.4: DCCP Half Connection

DCCP implements congestion control and the user of the applications can make a choice of congestion control mechanisms. The two hosts agreed on the congestion control mechanism during the initiation of the connection. One byte congestion control identifier called CCID, defines the mechanisms. Among various Congestion Control IDentifier, CCID 2 and CCID 3 are well defined.

3.7.4 CCID - 2 - DCCP TCPlike

CCID -2 implements congestion control through tracking a transmission window by regulating the transmit rate similarly to TCP. It uses Additive Increase Multiplicative Decrease Scheme for Congestion Control. It is best suited for applications which can

- adapt to the changes of congestion control window
- which need as much bandwidth as possible in network

The features of CCID - 2 are:

- Duplicate acknowledgement indicates some loss of data packet.
- Sender has timeout option like TCP sender retransmission timeout.

Every time the congestion event occurs, CCID 2 reduces its cwnd. Every congestion event is indicated by ECN or duplicate acknowledgement.

TCP-like, resembling TCP, uses a congestion window. The window increases when there is no packet loss, and decreases by half when there are losses. The congestion window as well as the bandwidth have abrupt changes and are not appropriate for video streaming.

A typical DCCP half-connection with TCP-like and ECN comprises of the following steps:

a. The connection is initiated as described earlier. Use Ack Vector feature is negotiated on the DCCP-Request packet sent.

b. The sender transmits data packets as governed by the congestion window, cwnd. Each data packet will have either the ECT(0) or ECT(1) codepoint set.

c. The receiver sends a DCCP-Ack packet including an ack vector for every Ack Ratio of data packets received.

d. The receiver's ack vectors include the ECN Nonce Echo so that the sender can verify that the receiver is not misbehaving.

e. The sender receives the acknowledgments and verifies the included ECN Nonce Echo. If it seems correct, the ack vector content is examined for lost or marked packets and cwnd is updated accordingly.

f. The sender also detects missing or marked DCCP-Ack packets and modifies the Ack ratio in response to congestion on the return path.

g. The sender acknowledges the receiver's acknowledgments at least once per congestion window.

  - In case the other half-connection is actively sending data, these acknowledgments are included in the acknowledgments of the receiver's data.

- However, if the other half-connection is quiescent, i.e. not sending data, the sender must do this proactively by occasionally acknowledging the receiver's acknowledgements.

h. The sender estimates the Round Trip Time (RTT) and uses this to calculate a TimeOut (TO) value, much like the Retransmit TimeOut (RTO) is calculated in TCP.

- If TO seconds have passed without receiving feedback from the receiver, all packets in that window are considered lost.

RTT Estimation

The sender saves timestamps for all recently sent data packets in the packet history. When the sender receives an acknowledgment corresponding to a data packet, it compares the current time with the time the packet was sent to obtain a sample of the RTT.

3.7.5 CCID - 3 - DCCP TCP Friendly Rate Control

CCID 3 uses TFRC mechanism [24]. DCCP sender calculates its transmission rate based on the following equation for fair smooth transmission rate which is required for real time applications. The equation is used regularly, for example each RTT. It allows moderate bandwidth changes and is more appropriate to video streaming.

$$T = \frac{s}{R\sqrt{\frac{2bp}{3}} + t_{RTO}(3\sqrt{\frac{3bp}{8}})p(1+32p^2)} \tag{3.4}$$

In equation 3.4, T = transmission rate in bytes/second, s = packet size in bytes, R = round trip time in seconds, b = number of packets acknowledged by a single TCP acknowledgement, p = loss event rate, tRTO = TCP retransmission time out value in seconds.

A typical DCCP half-connection with TFRC and ECN comprises of the following steps:

a. During connection establishment, the Loss Event Rate and Use Ack Vector features are negotiated. The ack vectors are needed to carry the ENE.

- Acknowledgments in TFRC are not generally required to be but the use of ack vectors introduces this demand. An ack-of-ack is sent whenever a feedback packet has been received which ensures that the receiver's acknowledgments are acknowledged approximately once per RTT as required.

b. The sender transmits data packets as governed by the TCP throughput equation. Each data packet will have either the ECT(0) or the ECT(1) codepoint set. The window counter is increased by one for every quarter of RTT that has passed since the last packet was sent.

c. A send timer controls the transmission of data packets. A packet can only be sent upon the expiration of this timer. When the timer expires, a so-called time slot is reached; the timer is rescheduled to match the allowed sending rate. Note that the timer is rescheduled even if no packet was sent when the timer expired. If the application desires to send data while the timer is running, the data is not sent until the next expiration.

d. When the allowed sending rate is updated by the reception of a feedback packet, the timer is cancelled and rescheduled to match the new rate. However, if the sending rate is lowered due to the expiration of the no feedback timer, the update is not carried out until the next time slot is reached.

e. The receiver sends a feedback packet with the calculated loss event rate and the receive rate back to the sender at least once per RTT.

  - Each packet contains the ENE in the form of an ack vector option. Since TFRC does not provide congestion control on acknowledgments, DCCP-Ack packets are marked as ECN-incapable.
  - A loss event is one or more marked or lost packets in a single RTT. The loss event rate is calculated by the weighted average loss interval over the last eight loss intervals. The loss event rate is then the inverse of the average loss interval.

f. The sender receives the acknowledgments and verifies the included ECN Nonce Echo. If the ENE is correct, the sending rate is updated according to the received information.

g. The sender acknowledges the receiver's acknowledgments at least once per RTT if the other half-connection is quiescent.

h. The sender estimates the Round Trip Time (RTT) and uses this to calculate a TimeOut (TO) value. If TO seconds have passed without receiving feedback from the receiver, the sending rate is halved.

RTT estimation

The receiver requires an approximation of the current RTT to initialise the loss interval history after the first loss event.

- In CCID 3, the RTT is estimated from the window counter and the arrival times of the received data packets.
- The calculation requires a minimum of two data packets with a window counter difference larger than four to provide an accurate estimate of the current RTT. So, the RTT is estimated from the packets exchanged during the handshake.
- The client estimates the RTT from the send time of the DCCP-Request packet and the receive time of the following DCCP-Response packet.
- The server uses the send time of the DCCP-Response packet and the receive time of the DCCP-Ack packet that finalises the handshake.

Explicit congestion notification

Traditionally, routers drop packets when their buffers overflow to indicate congestion in the network. With the introduction of Active Queue Management (AQM), a router can detect congestion before the queue overflows and explicitly notify the end nodes by other means than dropping packets. A router that supports ECN may set the Congestion Experienced (CE) codepoint on ECN capable packets when it detects congestion as an alternative to dropping the packets. A codepoint is a combination of the two ECN bits reserved in the IP header. The possible codepoints are presented as

- Not-ECT for Not-ECN-Cable Transport

- ECT(0) for ECN-Cable Transport (Nounce 0)
- ECT(1) for ECN-Cable Transport (Nounce 1)
- CE for Congestion Experienced - Marked

A malicious receiver might ignore to report marked packets to keep the sender from lowering the send rate. ECN Nonces introduce a way for the sender to verify that the receiver is not misbehaving. The sender includes either Nonce 0 or Nonce 1 on sent packets. The receiver calculates the sum of the received nonces, the ECN Nonce Echo (ENE), and returns it to the sender. When a packet is marked in the network, the nonce information is lost. Hence, to report a lost packet as received, the receiver must guess the correct ENE to return. Both TCP-like and TFRC can make use of ECN and the ECN Nonce. In ns, the ECN bits are located within the flags header. Most queuing mechanisms have been altered to recognise the codepoints from table 1. The DCCP agent automatically generates and sets random nonces on outgoing packets if ECN is enabled.

3.8 Summary of the Protocols

Table 3.2: Summary of the Protocols

Protocols	**C. Detection**	**C. Notification**	**C. Avoidance**	**Reliability**
STCP [9]	Queue Length	Implicit	AIMD	Limited using ACK and NACK
CODA [10]	Queue Length & Channel Status	Explicit	AIMD	None
CCF [6]	Packet Service Time	Implicit	Exact	None
DPCC [12]	Packet Service Time	Explicit	Exact	None
UDDP[14]	Packet Inter-Departure Time	Implicit	Exact	None
DCCP[15]	Duplicate Ack	Explicit	AIMD	None

Chapter 4

Proposed Algorithm

Congestion control is one of the services done by transport layer protocols to mitigate congestion in the network. Congestion, in wireless sensor networks, not only wastes the scarce energy due to a large number of retransmissions and packet drops, but also hinders the event detection reliability and link utilization.

Packets sent to the sink are highly compressed at the source. Compression standards such as the JPEG2000 and the MPEG introduce features such as the region of interest (ROI) and the inter and intra-frames. These special packets carry original content that cannot be retrieved through interpolation. Hence, dropping packets indiscriminately, may cause discernible disruptions in the multimedia content. Thus, we argue that some form of selective reliability, must be introduced for these packets in a WMSN.

Most of the existing transport protocols for WSN do not take into consideration the multimedia requirements in WMSN. Moreover, majority of the existing protocols do not provide congestion control and reliability simultaneously, i.e. either of them is implemented in the existing protocols.

Therefore, according to [1], there is a necessity of developing Transport Protocol that gives better performance as compared to the existing ones to control the congestion and provide reliability both, considering the multimedia requirements in WMSN.

In the previous chapter, the Datagram Congestion Control Protocol was described in detail.CCID defines the nature of congestion control mechanism. The two congestion controlled mechanisms followed are CCID - 2 - DCCP TCPlike and CCID - 3 - DCCP TCP Friendly Rate Control.

CCID 2 uses a congestion window. The window increases when there is no packet loss, and decreases by half when there are losses.

CCID 3 uses TFRC mechanism. DCCP sender calculates its transmission rate based on the following equation for fair smooth transmission rate which is required for real time applications.

But, DCCP faces the limitation that it does not provide Reliability. [22]

4.1 Proposed Algorithm

The proposed algorithm focuses on providing **Controlled Reliability** and Congestion Control both at the same time which is the core research area now a days.

Step - 1 Begin.

Step - 2 DCCP is implemented for **Wireless Sensor Network**

Step - 3 Connection is established. During connection establishment the Loss Event Rate is negotiated.

Step - 4 At the Sender endpoint,

4.1 The sender transmits the data as governed by the throughput equation 5.

$$T = \frac{s}{R\sqrt{\frac{2bp}{3}} + t_{RTO}(3\sqrt{\frac{3bp}{8}})p(1+32p^2)} \tag{4.1}$$

- T = Transmission rate in bytes/second,
- s = packet size in bytes,
- R = round trip time in seconds,
- b = number of packets acknowledged,
- p = loss event rate,
- tRTO = TCP retransmission time out value in seconds.

Step - 5 The following process is followed at the Receiver endpoint.

5.1 The receiver sends a feedback with the calculated loss event rate and ENE at least once per RTT.

5.2 The receiving end acknowledges all data received, even if there are gaps in the sequence.

Step - 6 The sender receives the ack and verifies the included ECN Nonce Echo.

6.1 If the ENE is correct, the sending rate is updated accordingly.

6.2 The sender estimates (RTT) and uses this to calculate a TimeOut value. If TO seconds have passed without receiving feedback from the receiver, the sending rate is halved.

Step - 7 If the sender does not receive the acknowledgement of any of the frame, it performs the following steps to provide **Controlled Reliability**:

7.1 The sender waits till the TimeOut value TO.

7.2 If it still does not receive the acknowledgement, it retransmits the unacknowledged frame ONLY IF IT WAS Intra-coded Frame (I-Frame).

7.3 If it was Inter-coded Frame (B-Frame or P-Frame), it does not resend the unacknowledged frame.

Step - 8 The connection termination is done by closing request from the either endpoints.

Step - 9 End.

4.2 Details of the Proposed Algorithm

The above Proposed Algorithm follows the following approach:

In the first step, the connection is established using three way handshake. In Request state, a client use an exponential-backoff timer to send new Request packets if no response is received. If no response after some predefined time, the client will give up.

The second step describes the processes that are followed at the Sender endpoint as follows:

The data is divided into a stream which is a unidirectional logical channel established between the two endpoints, within which all user messages are delivered in sequence. The sender transmits streams as governed by the throughput equation 4.2. The Sender assigns a stream sequence number to each message passed to it by the endpoint.

$$Ti = \frac{s}{R\sqrt{\frac{2bp}{3}} + t_{RTO}(3\sqrt{\frac{3bp}{8}})p(1+32p^2)} \tag{4.2}$$

- Ti = Ideal transmission rate in bytes/second,
- s = packet size in bytes,
- R = round trip time in seconds,
- b = number of packets acknowledged by a single TCP acknowledgement,
- p = loss event rate,
- tRTO = TCP retransmission time out value in seconds.

Each data packet will have either the ECT(0) or the ECT(1) codepoint set.

Explicit congestion notification Traditionally, routers drop packets when their buffers overflow to indicate congestion in the network. With the introduction of Active Queue

Management (AQM), a router can detect congestion before the queue overflows and explicitly notify the end nodes by other means than dropping packets. A router that supports ECN may set the Congestion Experienced (CE) codepoint on ECN capable packets when it detects congestion as an alternative to dropping the packets. A codepoint is a combination of the two ECN bits reserved in the IP header. The possible codepoints are presented as

- Not-ECT for Not-ECN-Cable Transport
- ECT(0) for ECN-Cable Transport (Nounce 0)
- ECT(1) for ECN-Cable Transport (Nounce 1)
- CE for Congestion Experienced - Marked

A malicious receiver might ignore to report marked packets to keep the sender from lowering the send rate. ECN Nonces introduce a way for the sender to verify that the receiver is not misbehaving. The sender includes either Nonce 0 or Nonce 1 on sent packets. The receiver calculates the sum of the received nonces, the ECN Nonce Echo (ENE), and returns it to the sender. When a packet is marked in the network, the nonce information is lost. Hence, to report a lost packet as received, the receiver must guess the correct ENE to return. Both TCP-like and TFRC can make use of ECN and the ECN Nonce. In ns, the ECN bits are located within the flags header. Most queuing mechanisms have been altered to recognise the codepoints from table 1. The DCCP agent automatically generates and sets random nonces on outgoing packets if ECN is enabled.

The receiver sends a feedback packet with the calculated loss event rate and the receive rate back to the sender at least once per RTT.

A loss event is one or more marked or lost packets in a single RTT. The loss event rate is calculated by the weighted average loss interval over the last eight loss intervals. The loss event rate is then the inverse of the average loss interval.

The receiver requires an approximation of the current RTT to initialize the loss interval history after the first loss event.

- In CCID 3, the RTT is estimated from the window counter and the arrival times of the received data packets.
- The calculation requires a minimum of two data packets with a window counter difference larger than four to provide an accurate estimate of the current RTT. So, the RTT is estimated from the packets exchanged during the handshake.
- The client estimates the RTT from the send time of the DCCP-Request packet and the receive time of the following DCCP-Response packet.
- The server uses the send time of the DCCP-Response packet and the receive time of the DCCP-Ack packet that finalizes the handshake.

In the next step, the receiver sends a feedback packet with the calculated loss event rate and the receive rate back to the sender at least once per RTT. The receiving end acknowledges all the frames received. If the sender does not receive the acknowledgement of any of the frame, it performs the following steps to provide **Controlled Reliability**:

- The sender waits till the TimeOut value TO.
- If it still does not receive the acknowledgement, it retransmits the unacknowledged frame ONLY IF IT WAS Intra-coded Frame (I-Frame).
- If it was Inter-coded Frame (B-Frame or P-Frame), it does not resend the unacknowledged frame.

The Proposed Algorithm gives Controlled Reliability along with Congestion Control - TFRC by retransmitting only Intra-coded frames due to their importance, thereby not creating an overhead in the network.

Finally, the connection termination uses a handshake consisting of a Closing Request from either side and a Close Confirmation from the opposite side. Once the Termination is successfully accomplished, the an endpoint quietly preserves a socket for some specific time after its connection has closed for ensuring that no connection duplicating the current connection's source and destination addresses and ports can start up while old packets might remain in the network.

Thus, Controlled Reliability and Sequenced Delivery of Data can be achieved along with Congestion control according to the proposed algorithm.

Chapter 5

Simulators

5.1 Introduction

In the network research area, it is very costly to deploy a complete test bed containing multiple networked computers, routers and data links to validate and verify a network protocol or a specific network algorithm. The network simulators in these circumstances save a lot of money and time in accomplishing this task. Network simulators are also particularly useful in allowing the network designers to test new networking protocols or to change the existing protocols in a controlled and reproducible manner. Thus we use a network simulator to test the proposed protocol.

Various simulators were tried so as to find the best suited for implementing Transport Protocol for Congestion Control in WMSN, whose overview are described in the following section.

5.2 NS-2

ns-2 [25]in its different versions is one of the most popular simulation environments for research. It has a hybrid approach to programming simulations with both C++ and an object-oriented version of Tcl scripting called OTcl. This duality can lead to confusion when not familiar with the system, but it proves to be very convenient once the user becomes acquainted with it. The modules are developed using C++, in order to provide higher simulation speeds by the use of compiled code.

C++ modules are configured and executed via OTcl scripts, which provide the description of the simulation environment and the configuration parameters for each module involved. This OTcl scripts are not compiled but interpreted by the ns software. This makes the set-up of simulations very easy and convenient to batch, as there is no compilation needed to run the scripts, and these contain all the required configuration parameters for the C++ modules.

This duality becomes critical when it comes to develop or modify modules. The modules have two parts: one programmed using C++ and other OTcl. This is required to provide the usability features previously mentioned.

There is an All-in-One package available for most of the releases. These versions include the network simulator, network animator –NAM– and xGraph in the latest version available at the moment of the creation of the package. The installation is not quite straightforward if you are not using one of the systems supported out-of-the-box for that version, but is easy to find community-developed scripts to compile and install the software properly.The installation of extra modules may require additions and modifications in the configuration files in order to work, being usually simple and well documented.

There is an extensive documentation for the network simulator and its modules, in addition to *.tcl example files provided in the distribution in order to both validate the installation of the simulator and learn how to script for the different areas of application of the simulator.

The disadvantage of ns-2 is mainly the limited scalability in terms of number of nodes being simulated, which is not a fixed limit, but it depends on the simulation parameters. This fact is related with the lack of memory management of ns-2: it may require multiple times the amount of memory than some of its alternatives for similar simulations. This is in part a consequence of the use of interpreted software (OTcl), which in 1989 when the ns project was born was a very convenient method to improve the simulation work-flow. However, at present, when the compilation process is not time-consuming, it is considered an unnecessary legacy burden when conducting large simulations.

Another important disadvantage has already been introduced when stated that ns-2 is in its different versions the most used software: not all modules are updated and valid for all the versions. There have been different points in the development where a number of modules stopped performing properly, so there is a considerable number of research executed with older versions as those are able to execute the modules required by the developers. Outdated versions lack general improvements and patches on different parts of the software which may influence the simulation results and their validity.

5.3 NRL-Sensorsim

SensorSim, [26] which has been built on NS-2, is a simulation framework for sensor networks. It provides sensor channel models, energy consumers, lightweight protocol stacks for wireless micro sensors, scenario generation and hybrid simulation. The sensor channel models the dynamic interaction between the sensor nodes and the physical environment. At each node, energy consumers are said to control the power in multiple modes to efficiently use the power and prolong the nodesŠ lifetime. But this simulator is no longer developed, therefore, no more available.

5.4 Evalvid

Evalvid-RA (Rate Adapive) [27]is the first tool to create realistic ŞonlineŤ rate adaptive streaming media traffic. It includes:

- a simulation time rate controller to modulate the quantizer scale used by a real codec
- realistic frame packetizing Ţ the ability (through ns-2) to choose network complexity, protocol and queue management support
- a framework that is scalable to a large number of simultaneous video sources

- and finally at the receiver side being able to restore the media files supporting PSNR and other QoS metrics calculation

Due to the trace file approach of Evalvid-RA, absolute delay and delay jitter impairments to the media decoding process can be investigated in a post-process, thus decoupling network and receiver media player constraints. Although we recognize the importance of mathematical models for traffic and queue statistics analysis, we believe that the complexity of the heterogeneous networks makes realistic simulation a better tool, especially when being able to compute end-user QoS metrics such as PSNR, or even perform human subjective tests.

Evalvid-RA is a framework and tool-set to enable simulation of rate adaptive VBR video. Evalvid-RAŠs main capability is the generation of true rate adaptive MPEG-4 VBR traffic, i.e. the codec output is dependent of the aggregate traffic passing through the network bottlenecks. In addition, the received media traces are used to restore true media files that can be visually inspected and PSNR and MOS scores can be calculated when comparing with the original material. The tool-set includes an online (at simulation time) rate controller that, based on network congestion signals, chooses video quality and bit rates from corresponding pre-processed trace files.

Chapter 6

Installation and Error Solving

Various simulators were tried so as to find the best suited for implementing Transport Protocol for Congestion Control in WMSN. According to the requirements of the project, ns-2, Evalvid and NRL-Sensorsim were found to be the best suited ones. Also, different versions of ns2 from ns2.26 to ns2.35 were implemented, but none of them, except ns2.35, worked properly due to the compatibility issues among NRL-Sensor patch and DCCP Patch. Finally, ns2.35 was taken as the platform to implement the project.

6.1 Installing ns2.35

The following steps are needed to install ns2.35 in Fedora 17, 32-bit version.

a. Pre-installations for ns2.35

- su
- yum install gcc gcc-c++ make binutils
- yum update kernel
- yum install kernel-headers kernel-devel
- yum install autoconf
- yum install automake
- yum install libX11-devel
- yum install xorg-X11-proto-devel

- yum install libXmu-devel
- yum install libXt-devel
- exit

b. Installing ns2.35

- tar -xvfz ns-allinone-2.35.tar.gz
- cd ns-allinone-2.35
- Install using ./install
- Set the path of Library, TCL, SGRAPH, NS, NAM for ns2.35 in .bashrc

6.2 Installing Evalvid-Rate Adaptive

The following steps are needed to install Evalvid-Rate Adaptive in Fedora 17, 32-bit version.

a. On the destination system, make a new folder (evalvid-ra) and copy the appropriate files into that folder.

b. Make appropriate changes in the following lines so as to implement the functionality of Evalvid - RA

- Makefile
- common/packet.h
- tcl/lib/ns-default.tcl
- Copy the files to tcp folder

c. Run from home ./configure; make clean; make

6.3 Installing NRL-SENSOR SIM

The following steps are needed to install NRL-SENSOR SIM in Fedora 17, 32-bit version.

a. Copy the folders phenom, nrlolsr and sensornets-NRL into the directory ns-allinone2.35/ns-2.35

b. Make appropriate changes in the following lines so as to implement the functionality of NRL-SENSOR

   - Makefile
   - common/packet.h
   - mac/mac.cc
   - mac/wireless-phy.cc
   - mac/wireless-phy.h
   - queue/priqueue.cc
   - tcl/lib/ns-lib.tcl
   - tcl/lib/ns-mobilenode.tcl
   - tcl/lib/ns-namsupp.tcl
   - trace/cmu-trace.cc
   - trace/cmu-trace.h

c. Run from home ./configure; make clean; make

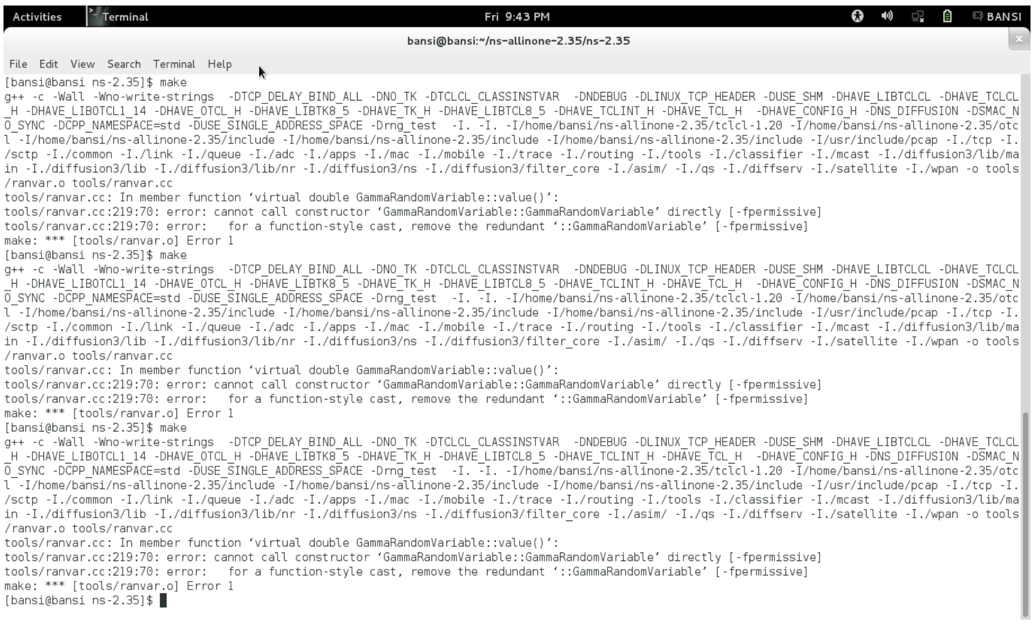
6.4 Solving Errors in NS-2.35

During the installation of ns2.35, there were many errors encountered and solved, some of which are as under:

- Error in file tools/ranvar.cc

  Solution :

  - Edit the file tools/ranvar.cc.
  - Remove GammaRandomVariable:: from line no. 219


Activities
Terminal
Fri 9:43 PM
BANSI
bansi@bansi:~/ns-allinone-2.35/ns-2.35
File Edit View Search Terminal Help

```
[bansi@bansi ns-2.35]$ make
g++ -c -Wall -Wno-write-strings  -DTCP_DELAY_BIND_ALL -DNO_TK -DTCLCL_CLASSINSTVAR  -DNDEBUG -DLINUX_TCP_HEADER -DUSE_SHM -DHAVE_LIBTCLCL -DHAVE_TCLCL
_H -DHAVE_LIBOTCL1_14 -DHAVE_OTCL_H -DHAVE_LIBTK8_5 -DHAVE_TK_H -DHAVE_LIBTCL8_5 -DHAVE_TCLINT_H -DHAVE_TCL_H  -DHAVE_CONFIG_H -DNS_DIFFUSION -DSMAC_N
O_SYNC -DCPP_NAMESPACE=std -DUSE_SINGLE_ADDRESS_SPACE -Drng_test  -I. -I. -I/home/bansi/ns-allinone-2.35/tclcl-1.20 -I/home/bansi/ns-allinone-2.35/otc
l -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/usr/include/pcap -I./tcp -I.
/sctp -I./common -I./link -I./queue -I./adc -I./apps -I./mac -I./mobile -I./trace -I./routing -I./tools -I./classifier -I./mcast -I./diffusion3/lib/ma
in -I./diffusion3/lib -I./diffusion3/lib/nr -I./diffusion3/ns -I./diffusion3/filter_core -I./asim/ -I./qs -I./diffserv -I./satellite -I./wpan -o tools
/ranvar.o tools/ranvar.cc
tools/ranvar.cc: In member function ‘virtual double GammaRandomVariable::value()’:
tools/ranvar.cc:219:70: error: cannot call constructor ‘GammaRandomVariable::GammaRandomVariable’ directly [-fpermissive]
tools/ranvar.cc:219:70: error:   for a function-style cast, remove the redundant ‘::GammaRandomVariable’ [-fpermissive]
make: *** [tools/ranvar.o] Error 1
[bansi@bansi ns-2.35]$ make
g++ -c -Wall -Wno-write-strings  -DTCP_DELAY_BIND_ALL -DNO_TK -DTCLCL_CLASSINSTVAR  -DNDEBUG -DLINUX_TCP_HEADER -DUSE_SHM -DHAVE_LIBTCLCL -DHAVE_TCLCL
_H -DHAVE_LIBOTCL1_14 -DHAVE_OTCL_H -DHAVE_LIBTK8_5 -DHAVE_TK_H -DHAVE_LIBTCL8_5 -DHAVE_TCLINT_H -DHAVE_TCL_H  -DHAVE_CONFIG_H -DNS_DIFFUSION -DSMAC_N
O_SYNC -DCPP_NAMESPACE=std -DUSE_SINGLE_ADDRESS_SPACE -Drng_test  -I. -I. -I/home/bansi/ns-allinone-2.35/tclcl-1.20 -I/home/bansi/ns-allinone-2.35/otc
l -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/usr/include/pcap -I./tcp -I.
/sctp -I./common -I./link -I./queue -I./adc -I./apps -I./mac -I./mobile -I./trace -I./routing -I./tools -I./classifier -I./mcast -I./diffusion3/lib/ma
in -I./diffusion3/lib -I./diffusion3/lib/nr -I./diffusion3/ns -I./diffusion3/filter_core -I./asim/ -I./qs -I./diffserv -I./satellite -I./wpan -o tools
/ranvar.o tools/ranvar.cc
tools/ranvar.cc: In member function ‘virtual double GammaRandomVariable::value()’:
tools/ranvar.cc:219:70: error: cannot call constructor ‘GammaRandomVariable::GammaRandomVariable’ directly [-fpermissive]
tools/ranvar.cc:219:70: error:   for a function-style cast, remove the redundant ‘::GammaRandomVariable’ [-fpermissive]
make: *** [tools/ranvar.o] Error 1
[bansi@bansi ns-2.35]$ make
g++ -c -Wall -Wno-write-strings  -DTCP_DELAY_BIND_ALL -DNO_TK -DTCLCL_CLASSINSTVAR  -DNDEBUG -DLINUX_TCP_HEADER -DUSE_SHM -DHAVE_LIBTCLCL -DHAVE_TCLCL
_H -DHAVE_LIBOTCL1_14 -DHAVE_OTCL_H -DHAVE_LIBTK8_5 -DHAVE_TK_H -DHAVE_LIBTCL8_5 -DHAVE_TCLINT_H -DHAVE_TCL_H  -DHAVE_CONFIG_H -DNS_DIFFUSION -DSMAC_N
O_SYNC -DCPP_NAMESPACE=std -DUSE_SINGLE_ADDRESS_SPACE -Drng_test  -I. -I. -I/home/bansi/ns-allinone-2.35/tclcl-1.20 -I/home/bansi/ns-allinone-2.35/otc
l -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/usr/include/pcap -I./tcp -I.
/sctp -I./common -I./link -I./queue -I./adc -I./apps -I./mac -I./mobile -I./trace -I./routing -I./tools -I./classifier -I./mcast -I./diffusion3/lib/ma
in -I./diffusion3/lib -I./diffusion3/lib/nr -I./diffusion3/ns -I./diffusion3/filter_core -I./asim/ -I./qs -I./diffserv -I./satellite -I./wpan -o tools
/ranvar.o tools/ranvar.cc
tools/ranvar.cc: In member function ‘virtual double GammaRandomVariable::value()’:
tools/ranvar.cc:219:70: error: cannot call constructor ‘GammaRandomVariable::GammaRandomVariable’ directly [-fpermissive]
tools/ranvar.cc:219:70: error:   for a function-style cast, remove the redundant ‘::GammaRandomVariable’ [-fpermissive]
make: *** [tools/ranvar.o] Error 1
[bansi@bansi ns-2.35]$ 
```

Figure 6.1: Error in file tools/ranvar.cc

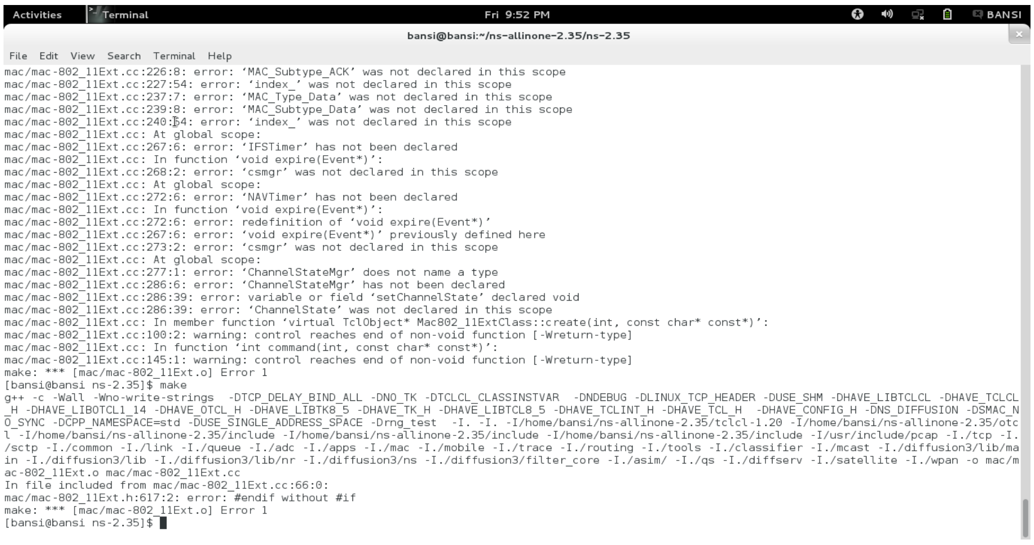
- Error in file mac/mac-802_11Ext.cc

  Solution - Change - 1:

  - Edit the file mac/mac-802_11Ext.cc.
  - #ifndef ns_mac_80211Ext_h
  - #define ns_mac_80211Ext_h
  - #endif
  - #include <cstddef>
  - #include "marshall.h"
  - #include "timer-handler.h"

  Solution - Change - 2:

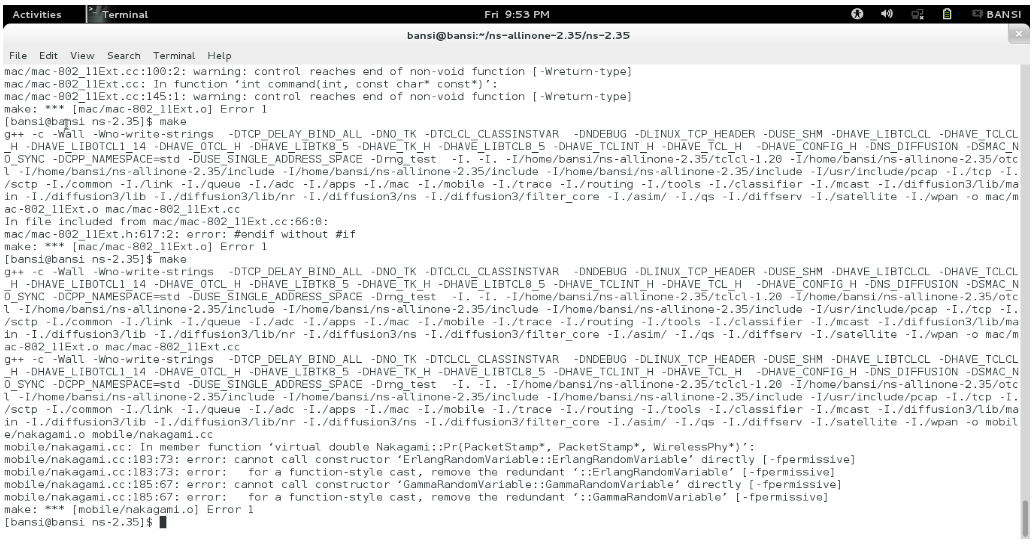
  - Edit the file mac/mac_802_11Ext.h.
  - #include <cstddef>


Activities
Terminal
Fri 9:52 PM
BANSI
bansi@bansi:~/ns-allinone-2.35/ns-2.35
File Edit View Search Terminal Help

```
mac/mac-802_11Ext.cc:226:8: error: ‘MAC_Subtype_ACK’ was not declared in this scope
mac/mac-802_11Ext.cc:227:54: error: ‘index_’ was not declared in this scope
mac/mac-802_11Ext.cc:237:7: error: ‘MAC_Type_Data’ was not declared in this scope
mac/mac-802_11Ext.cc:239:8: error: ‘MAC_Subtype_Data’ was not declared in this scope
mac/mac-802_11Ext.cc:240:54: error: ‘index_’ was not declared in this scope
mac/mac-802_11Ext.cc: At global scope:
mac/mac-802_11Ext.cc:267:6: error: ‘IFSTimer’ has not been declared
mac/mac-802_11Ext.cc: In function ‘void expire(Event*)’:
mac/mac-802_11Ext.cc:268:2: error: ‘csmgr’ was not declared in this scope
mac/mac-802_11Ext.cc: At global scope:
mac/mac-802_11Ext.cc:272:6: error: ‘NAVTimer’ has not been declared
mac/mac-802_11Ext.cc: In function ‘void expire(Event*)’:
mac/mac-802_11Ext.cc:272:6: error: redefinition of ‘void expire(Event*)’
mac/mac-802_11Ext.cc:267:6: error: ‘void expire(Event*)’ previously defined here
mac/mac-802_11Ext.cc:273:2: error: ‘csmgr’ was not declared in this scope
mac/mac-802_11Ext.cc: At global scope:
mac/mac-802_11Ext.cc:277:1: error: ‘ChannelStateMgr’ does not name a type
mac/mac-802_11Ext.cc:286:6: error: ‘ChannelStateMgr’ has not been declared
mac/mac-802_11Ext.cc:286:39: error: variable or field ‘setChannelState’ declared void
mac/mac-802_11Ext.cc:286:39: error: ‘ChannelState’ was not declared in this scope
mac/mac-802_11Ext.cc: In member function ‘virtual TclObject* Mac802_11ExtClass::create(int, const char* const*)’:
mac/mac-802_11Ext.cc:100:2: warning: control reaches end of non-void function [-Wreturn-type]
mac/mac-802_11Ext.cc: In function ‘int command(int, const char* const*)’:
mac/mac-802_11Ext.cc:145:1: warning: control reaches end of non-void function [-Wreturn-type]
make: *** [mac/mac-802_11Ext.o] Error 1
[bansi@bansi ns-2.35]$ make
g++ -c -Wall -Wno-write-strings  -DTCP_DELAY_BIND_ALL -DNO_TK -DTCLCL_CLASSINSTVAR  -DNDEBUG -DLINUX_TCP_HEADER -DUSE_SHM -DHAVE_LIBTCLCL -DHAVE_TCLCL
_H -DHAVE_LIBOTCL1_14 -DHAVE_OTCL_H -DHAVE_LIBTK8_5 -DHAVE_TK_H -DHAVE_LIBTCL8_5 -DHAVE_TCLINT_H -DHAVE_TCL_H  -DHAVE_CONFIG_H -DNS_DIFFUSION -DSMAC_N
O_SYNC -DCPP_NAMESPACE=std -DUSE_SINGLE_ADDRESS_SPACE -Drng_test  -I. -I. -I/home/bansi/ns-allinone-2.35/tclcl-1.20 -I/home/bansi/ns-allinone-2.35/otc
l -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/usr/include/pcap -I./tcp -I.
/sctp -I./common -I./link -I./queue -I./adc -I./apps -I./mac -I./mobile -I./trace -I./routing -I./tools -I./classifier -I./mcast -I./diffusion3/lib/ma
in -I./diffusion3/lib -I./diffusion3/lib/nr -I./diffusion3/ns -I./diffusion3/filter_core -I./asim/ -I./qs -I./diffserv -I./satellite -I./wpan -o mac/m
ac-802_11Ext.o mac/mac-802_11Ext.cc
In file included from mac/mac-802_11Ext.cc:66:0:
mac/mac-802_11Ext.h:617:2: error: #endif without #if
make: *** [mac/mac-802_11Ext.o] Error 1
[bansi@bansi ns-2.35]$ 
```

Figure 6.2: Error in file mac/mac-802_11Ext.cc

- Error in file mobile/nakagami.cc


Activities
Terminal
Fri 9:53 PM
BANSI
bansi@bansi:~/ns-allinone-2.35/ns-2.35
File Edit View Search Terminal Help

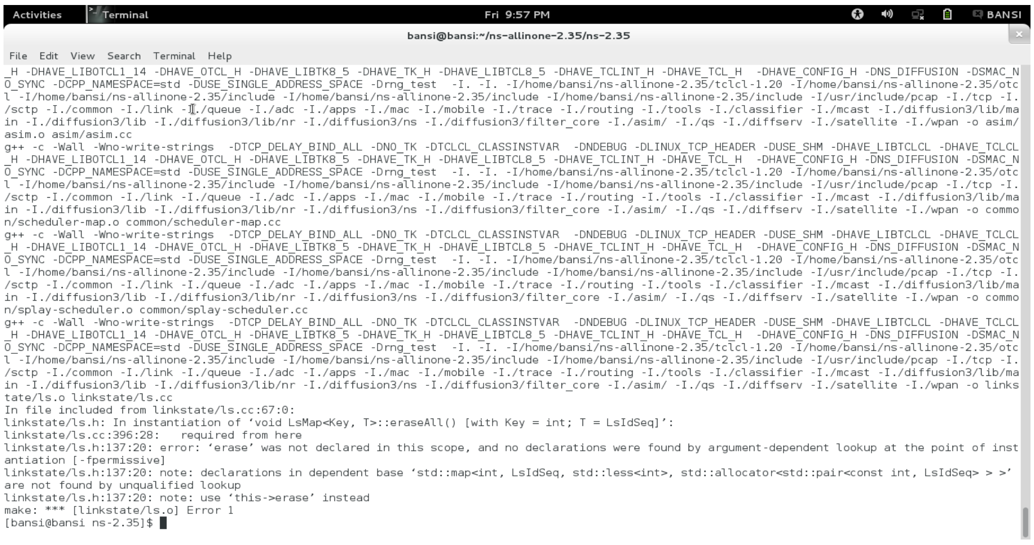
```
mac/mac-802_11Ext.cc:100:2: warning: control reaches end of non-void function [-Wreturn-type]
mac/mac-802_11Ext.cc: In function ‘int command(int, const char* const*)’:
mac/mac-802_11Ext.cc:145:1: warning: control reaches end of non-void function [-Wreturn-type]
make: *** [mac/mac-802_11Ext.o] Error 1
[bansi@bansi ns-2.35]$ make
g++ -c -Wall -Wno-write-strings  -DTCP_DELAY_BIND_ALL -DNO_TK -DTCLCL_CLASSINSTVAR  -DNDEBUG -DLINUX_TCP_HEADER -DUSE_SHM -DHAVE_LIBTCLCL -DHAVE_TCLCL
_H -DHAVE_LIBOTCL1_14 -DHAVE_OTCL_H -DHAVE_LIBTK8_5 -DHAVE_TK_H -DHAVE_LIBTCL8_5 -DHAVE_TCLINT_H -DHAVE_TCL_H  -DHAVE_CONFIG_H -DNS_DIFFUSION -DSMAC_N
O_SYNC -DCPP_NAMESPACE=std -DUSE_SINGLE_ADDRESS_SPACE -Drng_test  -I. -I. -I/home/bansi/ns-allinone-2.35/tclcl-1.20 -I/home/bansi/ns-allinone-2.35/otc
l -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/usr/include/pcap -I./tcp -I.
/sctp -I./common -I./link -I./queue -I./adc -I./apps -I./mac -I./mobile -I./trace -I./routing -I./tools -I./classifier -I./mcast -I./diffusion3/lib/ma
in -I./diffusion3/lib -I./diffusion3/lib/nr -I./diffusion3/ns -I./diffusion3/filter_core -I./asim/ -I./qs -I./diffserv -I./satellite -I./wpan -o mac/m
ac-802_11Ext.o mac/mac-802_11Ext.cc
In file included from mac/mac-802_11Ext.cc:66:0:
mac/mac-802_11Ext.h:617:2: error: #endif without #if
make: *** [mac/mac-802_11Ext.o] Error 1
[bansi@bansi ns-2.35]$ make
g++ -c -Wall -Wno-write-strings  -DTCP_DELAY_BIND_ALL -DNO_TK -DTCLCL_CLASSINSTVAR  -DNDEBUG -DLINUX_TCP_HEADER -DUSE_SHM -DHAVE_LIBTCLCL -DHAVE_TCLCL
_H -DHAVE_LIBOTCL1_14 -DHAVE_OTCL_H -DHAVE_LIBTK8_5 -DHAVE_TK_H -DHAVE_LIBTCL8_5 -DHAVE_TCLINT_H -DHAVE_TCL_H  -DHAVE_CONFIG_H -DNS_DIFFUSION -DSMAC_N
O_SYNC -DCPP_NAMESPACE=std -DUSE_SINGLE_ADDRESS_SPACE -Drng_test  -I. -I. -I/home/bansi/ns-allinone-2.35/tclcl-1.20 -I/home/bansi/ns-allinone-2.35/otc
l -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/usr/include/pcap -I./tcp -I.
/sctp -I./common -I./link -I./queue -I./adc -I./apps -I./mac -I./mobile -I./trace -I./routing -I./tools -I./classifier -I./mcast -I./diffusion3/lib/ma
in -I./diffusion3/lib -I./diffusion3/lib/nr -I./diffusion3/ns -I./diffusion3/filter_core -I./asim/ -I./qs -I./diffserv -I./satellite -I./wpan -o mac/m
ac-802_11Ext.o mac/mac-802_11Ext.cc
g++ -c -Wall -Wno-write-strings  -DTCP_DELAY_BIND_ALL -DNO_TK -DTCLCL_CLASSINSTVAR  -DNDEBUG -DLINUX_TCP_HEADER -DUSE_SHM -DHAVE_LIBTCLCL -DHAVE_TCLCL
_H -DHAVE_LIBOTCL1_14 -DHAVE_OTCL_H -DHAVE_LIBTK8_5 -DHAVE_TK_H -DHAVE_LIBTCL8_5 -DHAVE_TCLINT_H -DHAVE_TCL_H  -DHAVE_CONFIG_H -DNS_DIFFUSION -DSMAC_N
O_SYNC -DCPP_NAMESPACE=std -DUSE_SINGLE_ADDRESS_SPACE -Drng_test  -I. -I. -I/home/bansi/ns-allinone-2.35/tclcl-1.20 -I/home/bansi/ns-allinone-2.35/otc
l -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/usr/include/pcap -I./tcp -I.
/sctp -I./common -I./link -I./queue -I./adc -I./apps -I./mac -I./mobile -I./trace -I./routing -I./tools -I./classifier -I./mcast -I./diffusion3/lib/ma
in -I./diffusion3/lib -I./diffusion3/lib/nr -I./diffusion3/ns -I./diffusion3/filter_core -I./asim/ -I./qs -I./diffserv -I./satellite -I./wpan -o mobil
e/nakagami.o mobile/nakagami.cc
mobile/nakagami.cc: In member function ‘virtual double Nakagami::Pr(PacketStamp*, PacketStamp*, WirelessPhy*)’:
mobile/nakagami.cc:183:73: error: cannot call constructor ‘ErlangRandomVariable::ErlangRandomVariable’ directly [-fpermissive]
mobile/nakagami.cc:183:73: error:   for a function-style cast, remove the redundant ‘::ErlangRandomVariable’ [-fpermissive]
mobile/nakagami.cc:185:67: error: cannot call constructor ‘GammaRandomVariable::GammaRandomVariable’ directly [-fpermissive]
mobile/nakagami.cc:185:67: error:   for a function-style cast, remove the redundant ‘::GammaRandomVariable’ [-fpermissive]
make: *** [mobile/nakagami.o] Error 1
[bansi@bansi ns-2.35]$ 
```

Figure 6.3: Error in file mobile/nakagami.cc

Solution :

- Edit the file mobile/nakagami.cc.
- Remove redundant ::ErlangRandomVariable at line 183
- Remove redundant ::GammaRandomVariable at line 185

- Error in file linkstate/ls.h


Activities
Terminal
Fri 9:57 PM
BANSI
bansi@bansi:~/ns-allinone-2.35/ns-2.35
File Edit View Search Terminal Help

```
_H -DHAVE_LIBOTCL1_14 -DHAVE_OTCL_H -DHAVE_LIBTK8_5 -DHAVE_TK_H -DHAVE_LIBTCL8_5 -DHAVE_TCLINT_H -DHAVE_TCL_H  -DHAVE_CONFIG_H -DNS_DIFFUSION -DSMAC_N
O_SYNC -DCPP_NAMESPACE=std -DUSE_SINGLE_ADDRESS_SPACE -Drng_test  -I. -I. -I/home/bansi/ns-allinone-2.35/tclcl-1.20 -I/home/bansi/ns-allinone-2.35/otc
l -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/usr/include/pcap -I./tcp -I.
/sctp -I./common -I./link -I./queue -I./adc -I./apps -I./mac -I./mobile -I./trace -I./routing -I./tools -I./classifier -I./mcast -I./diffusion3/lib/ma
in -I./diffusion3/lib -I./diffusion3/lib/nr -I./diffusion3/ns -I./diffusion3/filter_core -I./asim/ -I./qs -I./diffserv -I./satellite -I./wpan -o asim/
asim.o asim/asim.cc
g++ -c -Wall -Wno-write-strings  -DTCP_DELAY_BIND_ALL -DNO_TK -DTCLCL_CLASSINSTVAR  -DNDEBUG -DLINUX_TCP_HEADER -DUSE_SHM -DHAVE_LIBTCLCL -DHAVE_TCLCL
_H -DHAVE_LIBOTCL1_14 -DHAVE_OTCL_H -DHAVE_LIBTK8_5 -DHAVE_TK_H -DHAVE_LIBTCL8_5 -DHAVE_TCLINT_H -DHAVE_TCL_H  -DHAVE_CONFIG_H -DNS_DIFFUSION -DSMAC_N
O_SYNC -DCPP_NAMESPACE=std -DUSE_SINGLE_ADDRESS_SPACE -Drng_test  -I. -I. -I/home/bansi/ns-allinone-2.35/tclcl-1.20 -I/home/bansi/ns-allinone-2.35/otc
l -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/usr/include/pcap -I./tcp -I.
/sctp -I./common -I./link -I./queue -I./adc -I./apps -I./mac -I./mobile -I./trace -I./routing -I./tools -I./classifier -I./mcast -I./diffusion3/lib/ma
in -I./diffusion3/lib -I./diffusion3/lib/nr -I./diffusion3/ns -I./diffusion3/filter_core -I./asim/ -I./qs -I./diffserv -I./satellite -I./wpan -o commo
n/scheduler-map.o common/scheduler-map.cc
g++ -c -Wall -Wno-write-strings  -DTCP_DELAY_BIND_ALL -DNO_TK -DTCLCL_CLASSINSTVAR  -DNDEBUG -DLINUX_TCP_HEADER -DUSE_SHM -DHAVE_LIBTCLCL -DHAVE_TCLCL
_H -DHAVE_LIBOTCL1_14 -DHAVE_OTCL_H -DHAVE_LIBTK8_5 -DHAVE_TK_H -DHAVE_LIBTCL8_5 -DHAVE_TCLINT_H -DHAVE_TCL_H  -DHAVE_CONFIG_H -DNS_DIFFUSION -DSMAC_N
O_SYNC -DCPP_NAMESPACE=std -DUSE_SINGLE_ADDRESS_SPACE -Drng_test  -I. -I. -I/home/bansi/ns-allinone-2.35/tclcl-1.20 -I/home/bansi/ns-allinone-2.35/otc
l -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/usr/include/pcap -I./tcp -I.
/sctp -I./common -I./link -I./queue -I./adc -I./apps -I./mac -I./mobile -I./trace -I./routing -I./tools -I./classifier -I./mcast -I./diffusion3/lib/ma
in -I./diffusion3/lib -I./diffusion3/lib/nr -I./diffusion3/ns -I./diffusion3/filter_core -I./asim/ -I./qs -I./diffserv -I./satellite -I./wpan -o commo
n/splay-scheduler.o common/splay-scheduler.cc
g++ -c -Wall -Wno-write-strings  -DTCP_DELAY_BIND_ALL -DNO_TK -DTCLCL_CLASSINSTVAR  -DNDEBUG -DLINUX_TCP_HEADER -DUSE_SHM -DHAVE_LIBTCLCL -DHAVE_TCLCL
_H -DHAVE_LIBOTCL1_14 -DHAVE_OTCL_H -DHAVE_LIBTK8_5 -DHAVE_TK_H -DHAVE_LIBTCL8_5 -DHAVE_TCLINT_H -DHAVE_TCL_H  -DHAVE_CONFIG_H -DNS_DIFFUSION -DSMAC_N
O_SYNC -DCPP_NAMESPACE=std -DUSE_SINGLE_ADDRESS_SPACE -Drng_test  -I. -I. -I/home/bansi/ns-allinone-2.35/tclcl-1.20 -I/home/bansi/ns-allinone-2.35/otc
l -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/home/bansi/ns-allinone-2.35/include -I/usr/include/pcap -I./tcp -I.
/sctp -I./common -I./link -I./queue -I./adc -I./apps -I./mac -I./mobile -I./trace -I./routing -I./tools -I./classifier -I./mcast -I./diffusion3/lib/ma
in -I./diffusion3/lib -I./diffusion3/lib/nr -I./diffusion3/ns -I./diffusion3/filter_core -I./asim/ -I./qs -I./diffserv -I./satellite -I./wpan -o links
tate/ls.o linkstate/ls.cc
In file included from linkstate/ls.cc:67:0:
linkstate/ls.h: In instantiation of ‘void LsMap<Key, T>::eraseAll() [with Key = int; T = LsIdSeq]’:
linkstate/ls.cc:396:28:   required from here
linkstate/ls.h:137:20: error: ‘erase’ was not declared in this scope, and no declarations were found by argument-dependent lookup at the point of inst
antiation [-fpermissive]
linkstate/ls.h:137:20: note: declarations in dependent base ‘std::map<int, LsIdSeq, std::less<int>, std::allocator<std::pair<const int, LsIdSeq> > >’
are not found by unqualified lookup
linkstate/ls.h:137:20: note: use ‘this->erase’ instead
make: *** [linkstate/ls.o] Error 1
[bansi@bansi ns-2.35]$ █
```

Figure 6.4: Error in file linkstate/ls.h

Solution :

- Edit the file linkstate/ls.h.
- use this->erase instead instead of erase at line 137

Chapter 7

Implementation of DCCP - TFRC Protocol for WMSN

After the successful installation of ns2.35, Evalvid-RA and NRL Sensorsim, DCCP-TRFC was implemented for WMSN. The scenerio was as under:

- There is Multimedia Communication between different Wireless Sensor Nodes.
- Certain frames are lost due to collision, congestion, etc.
- As the energy of the nodes reduces due to the communication, after a particular time, their color turns red indicating lack of sufficient energy.

7.1 Parameters of DCCP - TFRC Protocol for WMSN in ns2.35

For introducing Controlled Reliability in DCCP, the following files were modified:

- ../common/packet.h
- ../common/packet.h
- ../trace/cmu-trace.h
- ../trace/cmu-trace.cc

- ../dccp/dccp.h
- ../dccp/dccp.cc
- ../dccp/dccp_packets.h
- ../dccp/dccp_packets.cc
- ../dccp/dccp_tfrc.h
- ../dccp/dccp_tfrc.cc

The following were the parameters that were used for the implementation:

- Channel Type - WirelessChannel
- Radio-Propagation Model - Tworayground
- Network Interface Type - WirelessPhy
- MAC type - 802.11
- Interface Queue Type - Queue/DropTail/PriQueue
- Link Layer Type - LL
- Antenna Model - OmniAntenna
- No. of Nodes Generating Traffic - 1, 3, 5, 7, 9, 11, 13, 20, 27
- Grid - 500 X 500
- Routing Protocol - AODV
- Transport Protocol - DCCP/TFRC
- GOP - 13 - 1 I-frame, 12 B and P frames
- Type of Traffic - Video Traffic
- 10s Video
- Maximum Packet Size - 1024 bytes

The simulation was done for the above mentioned parameters, the snapshots of some of which are as shown in Figure 7.1 and Figure 7.2 :

- Multimedia Communication takes place for 30 nodes in a network out of which Node 6 is Sink and there are 7 Nodes Generating Multimedia Traffic which is shown as shown in Figure 7.1:

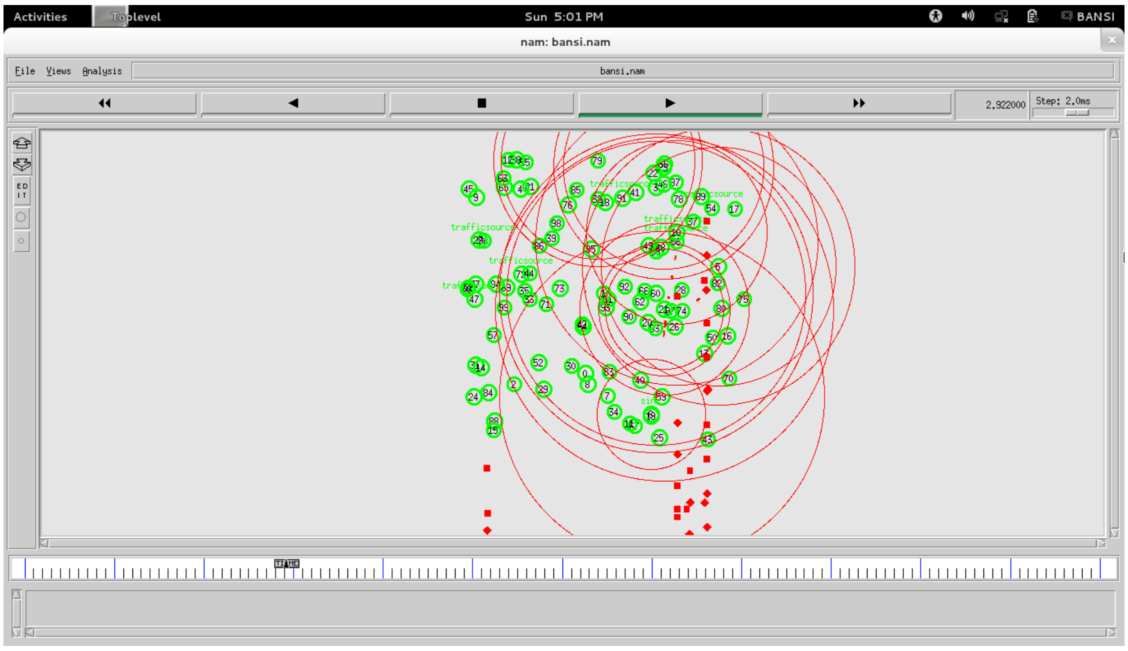


Figure 7.1: Multimedia Communication - Total 30 Nodes in Network, Node 6 as Sink, 7 Nodes Generating Multimedia Traffic

- Multimedia Communication takes place for 30 nodes in a network out of which Node 6 is Sink and there are 20 Nodes Generating Multimedia Traffic which is shown as shown in Figure 7.2:

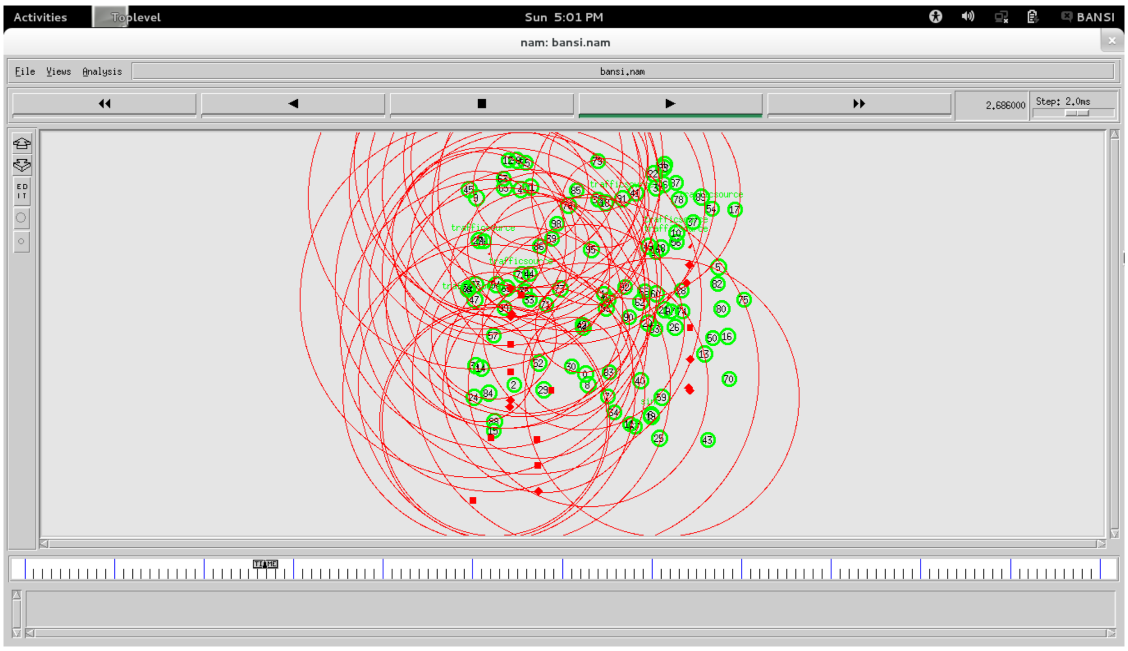


Figure 7.2: Multimedia Communication - Total 30 Nodes in Network, Node 6 as Sink, 20 Nodes Generating Multimedia Traffic

- Multimedia Communication takes place between Wireless Sensor Node 1 and Node 2 with Node 3 as shown in Figure 7.2

Chapter 8

Results and Analysis

8.1 Introduction

To analyze the performance of DCCP and DCCP-CR comparatively, the following parameters were considered:

Packet Delivery Ratio The ratio of No. of Packets received by the receiver with respect to the No. of Packets sent by the sender[8].

Throughput Average rate of successful message delivery over a wired or wireless communication channel (measured in packets per second)[8].

Routing Overhead Total number of routing packets (includes forwarded routing packets as well) transmitted in a network[8].

The implementation was done for DCCP and DCCP-CR and results were taken according to two different scenarios which are explained as under:

Scenario - 1 Here, the Total Number of Nodes of the Network were kept Constant i.e. 30. The nodes generating multimedia traffic were increased i.e. 1, 3, 5, 7, 9, 11, 13, 20, 27.

Scenario - 2 Here, the Total Number of Nodes of the Network were increased i.e. 10, 20, 30, 40, 50. The nodes generating multimedia traffic were kept constant i.e. 7.

The readings were taken and the graphs were plotted, which are described subsequently.

8.2 Simulation Results

As stated earlier, the simulation was done according to two different scenarios. The following shows the result and graphical analysis of Scenario - 1, where Total no. of nodes in the network are kept constant as 30. The nodes generating Multimedia traffic are taken as 1, 3, 5, 7, 9, 11, 13, 20, 27.

Table 8.1: PACKET DELIVERY RATIO for DCCP

NODES IN NETWORK - 30						
Traffic Nodes	**Simulation Runs**					
	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**
1	100	100	100	100	100	100
3	100	100	91.5751	84.6154	100	95.2381
5	68.4329	72.1416	61.0673	59.4513	70.3871	66.29604
7	51.7939	65.7675	43.2695	53.1697	55.3278	53.86568
9	49.7788	51.0217	61.7021	57.9735	43.6276	52.82074
11	41.4504	46.7606	42.6362	41.9316	58.2918	46.21412
13	49.503	48.969	35.6624	43.8956	32.9476	42.19552
20	25.3216	21.2141	27.018	25.436	20.1658	23.8311
24	25.5932	26.9814	20.6618	13.1235	15.5529	20.38256
27	20.1119	19.2978	21.1693	19.0042	21.0406	20.12476

Table 8.2: PACKET DELIVERY RATIO for DCCP with Controlled Reliability

NODES IN NETWORK - 30						
Traffic Nodes	**Simulation Runs**					
	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**
1	100	100	100	100	100	100
3	100	98.1123	94.17	94.0504	98.98	97.86254
5	75.778	72.2318	71.0007	75.9212	77.1	75.40634
7	67.1111	69.1252	67.2425	66.1924	66.0043	67.1351
9	65.7771	61.093	66.807	67.95	64.6785	65.26112
11	54.5559	72.4113	68.9362	49.9016	58.918	60.9446
13	57.9901	57.9908	58.7326	56.196	52.5673	56.69536
20	45.3216	41.44222	37.19	43.785	39.1998	41.38772
24	36.2978	36.62	37.6618	37.2359	36.0912	36.78134
27	35.1106	36.1987	30.9743	35	35.2009	44.4969

Table 8.3: THROUGHPUT for DCCP

NODES IN NETWORK - 30						
Traffic Nodes	**Simulation Runs**					
	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**
1	45.33	45.39	45.33	45.34	45.36	45.35
3	45.4	45.33	45.7	47.28	45.32	45.806
5	123.9	151.28	126.35	130.84	131.64	132.802
7	132.41	192.58	118.38	138.27	149.07	146.142
9	178.67	168.44	227.25	212.8	131.05	183.642
11	129.21	207.95	207.95	171.67	255.62	194.48
13	239.86	229.25	163.02	148.51	95.31	175.19
20	172.02	202.02	123.63	113.88	119.12	146.134
24	136.2	124.49	136.4	167.27	140.55	140.982
27	147.03	151.72	142.68	108.86	151.2	140.298

Table 8.4: THROUGHPUT for DCCP with Controlled Reliability

NODES IN NETWORK - 30						
Traffic Nodes	**Simulation Runs**					
	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**
1	44.4	44.49	44.33	44.32	44.01	44.31
3	44.79	44.9	44.72	44.51	44.43	44.67
5	121.21	124.89	125.75	119.91	119.68	122.288
7	131.13	142.58	138.97	138.01	135.53	137.244
9	164.91	166.99	163.28	164.92	167.23	165.466
11	181.43	175.29	178.55	178.44	176.49	178.04
13	158.9	161.88	158.01	161.46	161.94	160.438
20	157.89	155.1	160.34	158.99	158.94	158.252
24	131.23	134.92	131.59	132.77	131.25	132.352
27	131.92	129.09	130.22	131.82	130.24	130.658

Table 8.5: ROUTING OVERHEAD for DCCP

NODES IN NETWORK - 30						
Traffic Nodes	**Simulation Runs**					
	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**
1	32	30	32	32	31	31.4
3	30	32	102	62	32	51.6
5	580	187	240	1079	191	455.4
7	306	218	1075	769	582	590
9	721	597	572	601	583	614.8
11	590	467	1069	845	410	676.2
13	677	694	1349	467	961	829.6
20	1691	1596	760	952	1615	1322.8
24	1134	1172	1690	2289	2539	1764.8
27	1379	1493	1524	3587	1191	1834.8

Table 8.6: ROUTING OVERHEAD for DCCP with Controlled Reliability

NODES IN NETWORK - 30						
Traffic Nodes	**Simulation Runs**					
	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**
1	41	35	45	49	45	43
3	48	42	61	78	112	68.2
5	259	220	1124	555	300	491.6
7	449	456	892	945	570	662.4
9	742	689	628	689	698	689.2
11	695	697	701	755	724	714.4
13	878	915	899	902	901.8	899.16
20	1599	1368	1487	1251	1579	1456.8
24	1799	2010	1998	1876	1845	1905.6
27	1974	1984	1979	1923	1997	1971.4

Table 8.7: PACKET DELIVERY RATE for DCCP

TRAFFIC NODES - 7						
Nodes in Network	**Simulation Runs**					
	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**
10	51.7939	35.0148	36.3097	65.0661	42.703	46.1775
20	51.76	55.3008	52.3604	46.6954	44.942	50.21172
30	51.7939	65.7675	43.2695	53.1697	55.3278	53.86568
40	58.61	58.0254	53.3315	55.5705	60.9833	57.30414
50	58.0793	48.7726	38.5757	81.7912	60.7769	57.59914

Table 8.8: PACKET DELIVERY RATE for DCCP with Controlled Reliability

TRAFFIC NODES - 7						
Nodes in Network	**Simulation Runs**					
	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**
10	58.0015	58.9843	58.714	59.9942	58.4307	58.82494
20	62.2202	59.0978	64.8634	62.9833	62.7912	62.39118
30	66.1274	65.7983	69.1847	66.9901	66.4404	66.90818
40	71.8956	71.443	71.9462	71.7685	71.8031	71.77128
50	75.1342	71.9835	73.99	72.8757	73.1684	73.43036

Table 8.9: THROUGHPUT for DCCP

TRAFFIC NODES - 7						
Nodes in Network	**Simulation Runs**					
	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**
10	140.68	101.93	94.43	185.36	106.96	125.872
20	137.24	166.47	143.55	124.62	120.2	138.416
30	132.41	192.58	118.38	138.27	149.07	146.142
40	148.7	151.57	161.01	166.2	176.53	160.802
50	163.75	125.93	112.8	248.39	165.63	163.3

Table 8.10: THROUGHPUT for DCCP with Controlled Reliability

TRAFFIC NODES - 7						
Nodes in Network	**Simulation Runs**					
	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**
10	121.22	111.3	96.43	125.36	126.67	116.196
20	127.42	147	135.15	120.62	120.2	130.078
30	142.21	152.58	128.83	128.27	139.65	138.308
40	151.39	157.11	153.94	155.93	154.45	154.564
50	157.63	158.82	158.37	157.46	157.21	157.898

Table 8.11: ROUTING OVERHEAD for DCCP

TRAFFIC NODES - 7						
Nodes in Network	**Simulation Runs**					
	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**
10	83	62	140	84	70	87.8
20	159	140	367	497	868	406.2
30	306	218	1075	769	582	590
40	539	333	407	1300	1588	833.4
50	1652	1268	956	783	577	1047.2

Table 8.12: ROUTING OVERHEAD for DCCP with Controlled Reliability

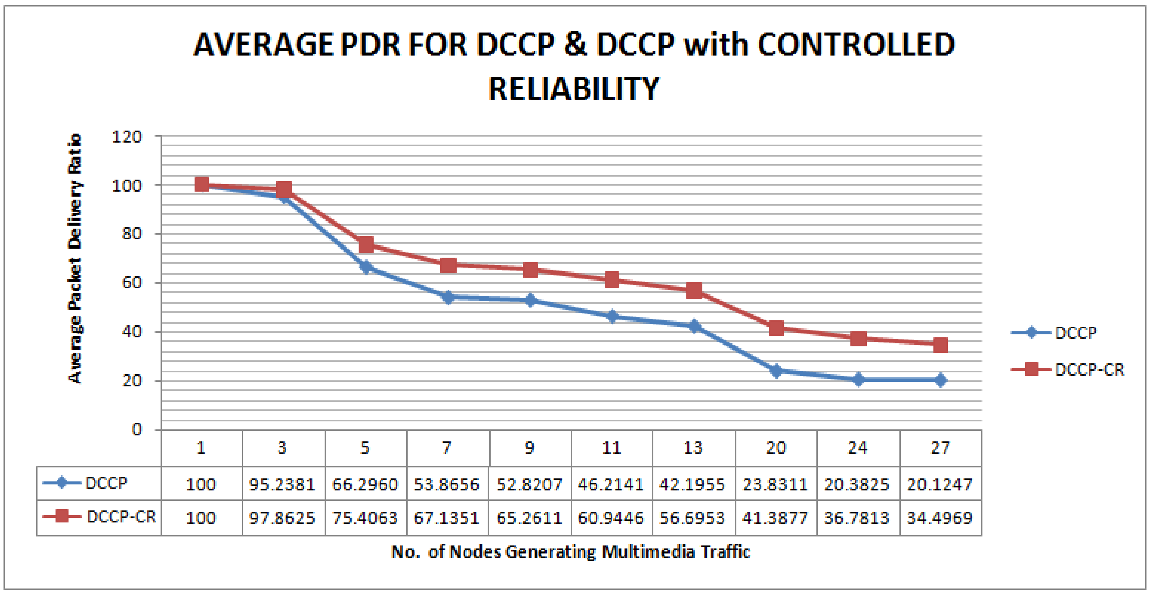
TRAFFIC NODES - 7						
Nodes in Network	**Simulation Runs**					
	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**
10	200	210	220	217	220	213.4
20	492	625	540	564	541	552.4
30	741	727	709	714	765	731.2
40	1024	1100	1035	1009	1021	1037.8
50	1287	1309	1295	1398	1376	1333

8.3 Result Analysis

As stated earlier, the simulation is implemented according to two different scenarios. The following shows the graphical analysis of the Average Throughput for **Scenario - 1**.

Here, the Total Number of Nodes of the Network were kept Constant i.e. 30. The nodes generating multimedia traffic were increased i.e. 1, 3, 5, 7, 9, 11, 13, 20, 27.

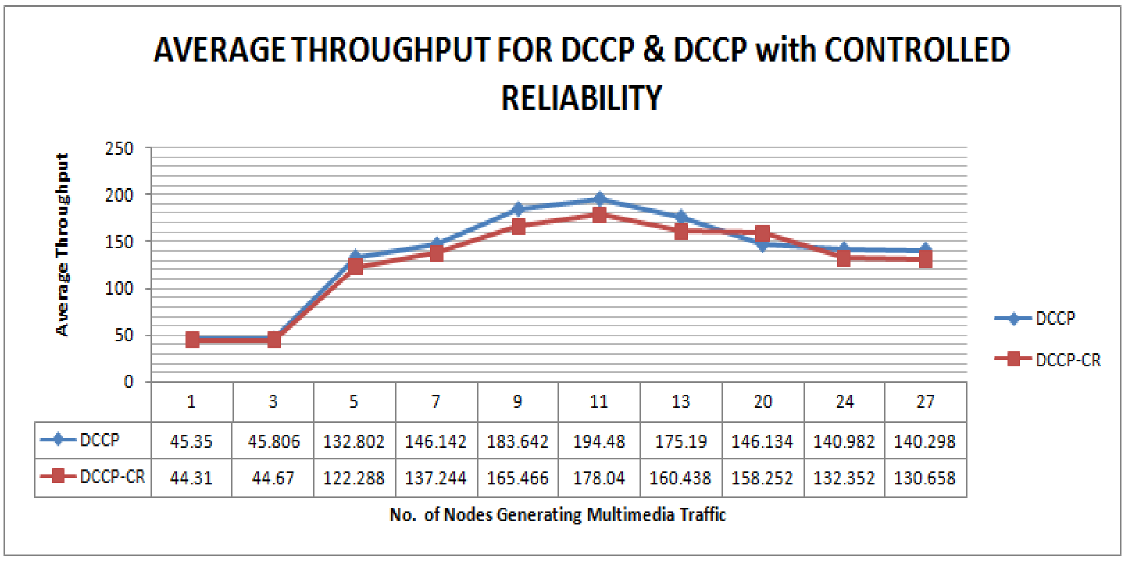
The figure 8.1 shows the graphical analysis of Packet Delivery Ratio.



	1	3	5	7	9	11	13	20	24	27
DCCP	100	95.2381	66.2960	53.8656	52.8207	46.2141	42.1955	23.8311	20.3825	20.1247
DCCP-CR	100	97.8625	75.4063	67.1351	65.2611	60.9446	56.6953	41.3877	36.7813	34.4969

Figure 8.1: Average PDR for DCCP and DCCP-CR

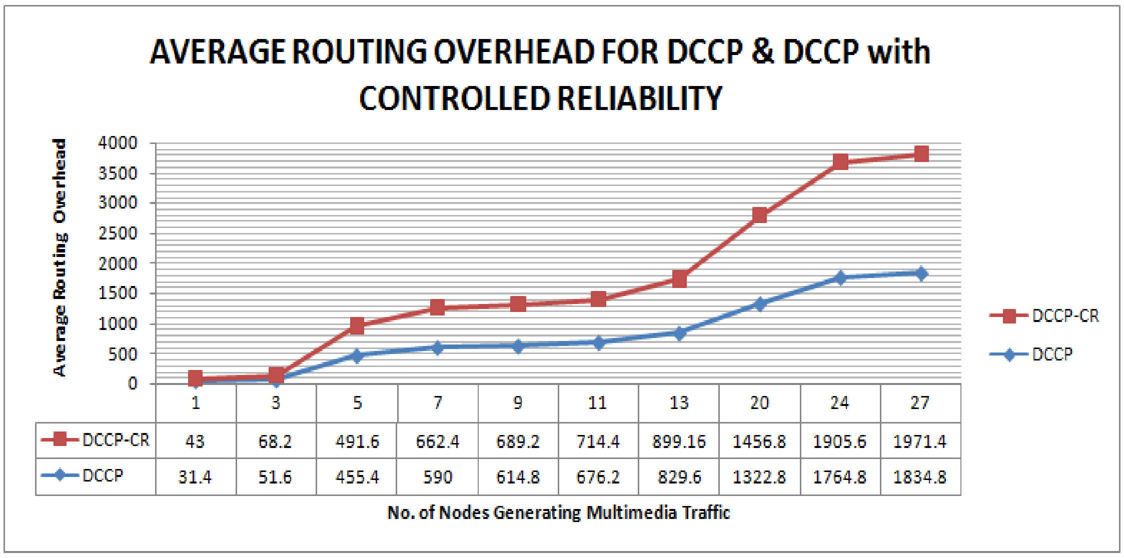
Now, due to retransmission of the lost I frames, the Packet Delivery Ratio of DCCP-CR (DCCP with CONTROLLED RELIABILITY) increases as compared to DCCP. This is the Advantage as the I-frames that were lost in DCCP, are now delivered which is shown by increase in PDR as shown in figure 8.2.



	1	3	5	7	9	11	13	20	24	27
DCCP	45.35	45.806	132.802	146.142	183.642	194.48	175.19	146.134	140.982	140.298
DCCP-CR	44.31	44.67	122.288	137.244	165.466	178.04	160.438	158.252	132.352	130.658

Figure 8.2: Average Throughput for DCCP and DCCP-CR

Here, the throughput of DCCP-CR decreases as compared to DCCP, which is a disadvantage. The reason behind it is that the due to retransmission of the I-frames, the Routing overhead increases in DCCP-CR as compared to DCCP as shown in the figure 8.3:

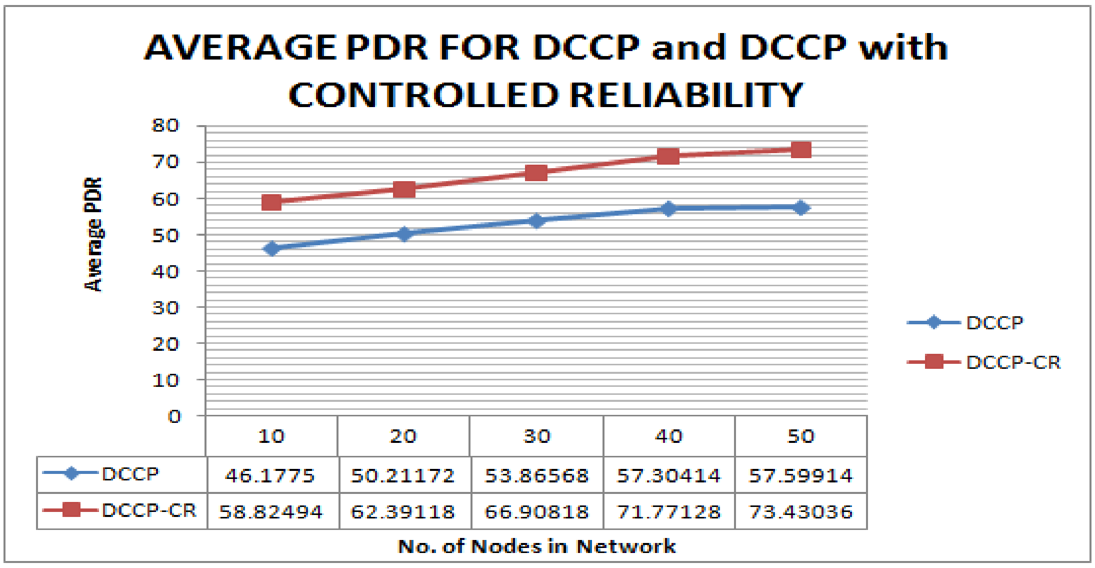


	1	3	5	7	9	11	13	20	24	27
DCCP-CR	43	68.2	491.6	662.4	689.2	714.4	899.16	1456.8	1905.6	1971.4
DCCP	31.4	51.6	455.4	590	614.8	676.2	829.6	1322.8	1764.8	1834.8

Figure 8.3: Average Routing Overhead for DCCP and DCCP-CR

The following shows the graphical analysis of Average Throughput for **Scenario - 2**.

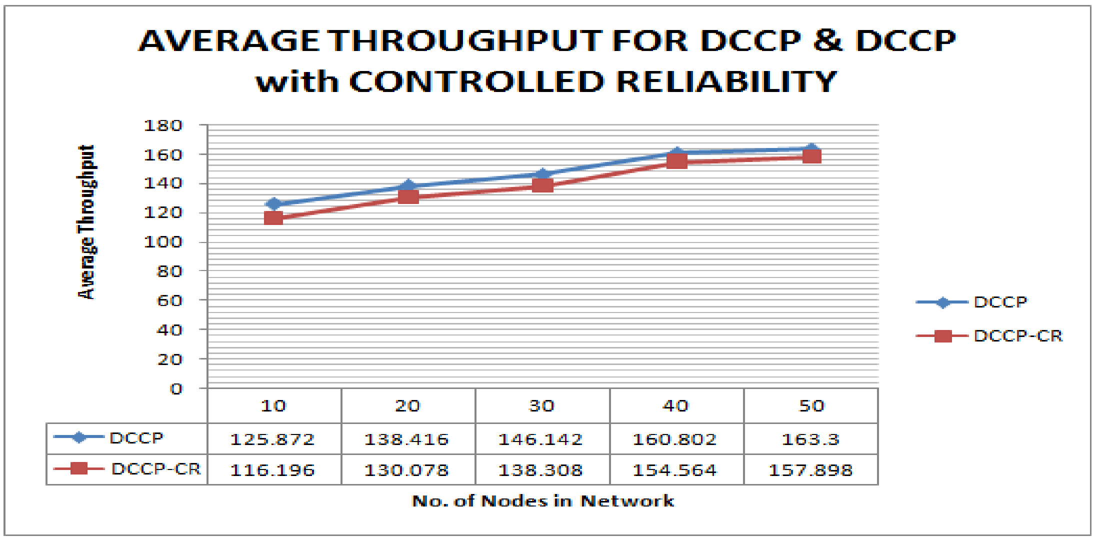
Here, the Total Number of Nodes of the Network were increased i.e. 10, 20, 30, 40, 50. The nodes generating multimedia traffic were kept constant i.e. 7. The figure 8.4shows the graphical analysis of Packet Delivery Ratio.



	10	20	30	40	50
DCCP	46.1775	50.21172	53.86568	57.30414	57.59914
DCCP-CR	58.82494	62.39118	66.90818	71.77128	73.43036

Figure 8.4: Average PDR for DCCP and DCCP-CR

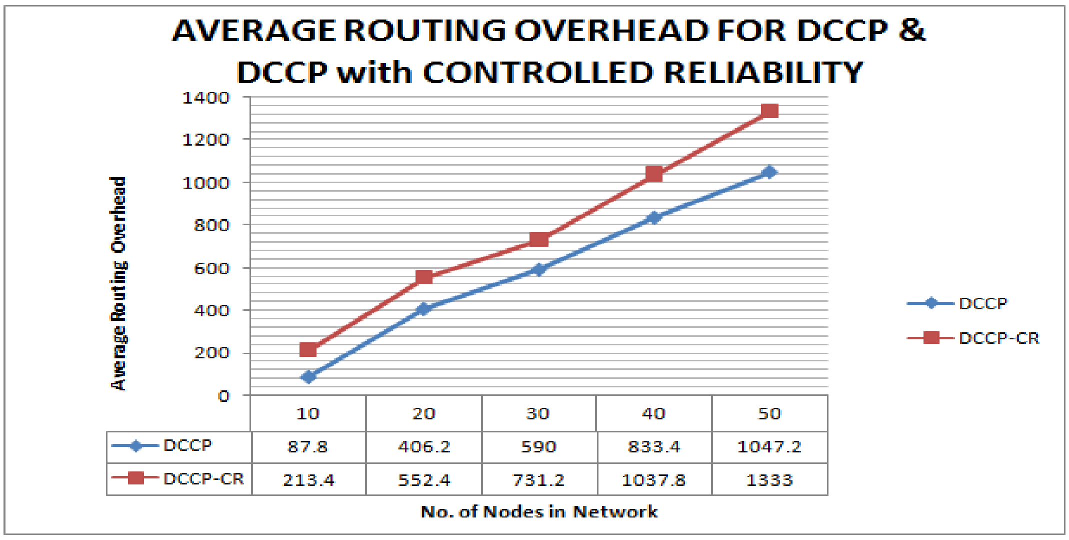
Now, due to retransmission of the lost I frames, the Packet Delivery Ratio of DCCP-CR increases as compared to DCCP. This is the Advantage as the I-frames that were lost in DCCP, are now delivered which is shown by increase in PDR according to figure 8.5.



	10	20	30	40	50
DCCP	125.872	138.416	146.142	160.802	163.3
DCCP-CR	116.196	130.078	138.308	154.564	157.898

Figure 8.5: Average Throughput for DCCP and DCCP-CR

Here, the throughput of DCCP-CR decreases as compared to DCCP, which is a disadvantage. The reason behind it is that the due to retransmission of the I-frames, the Routing overhead increases in DCCP-CR as compared to DCCP as shown in the figure 8.6:



	10	20	30	40	50
DCCP	87.8	406.2	590	833.4	1047.2
DCCP-CR	213.4	552.4	731.2	1037.8	1333

Figure 8.6: Average Routing Overhead for DCCP and DCCP-CR

Chapter 9

Conclusion and Future Scope

9.1 Conclusion

The proposed Algorithm DCCP-CR is an optimized algorithm which is better than the existing ones as it combines the features for congestion control from DCCP using CCID 3 which is using DCCP TCP Friendly Rate Control mechanism, along with providing Controlled Reliability by by retransmitting only Intra-coded frames (I-frame) due to their importance, and not retransmitting Inter-coded frame (B-Frame and P-Frame) thereby not creating an excess overhead in the network.

Due to retransmission of the lost I-frames, the Packet Delivery Ratio of DCCP-CR (DCCP with CONTROLLED RELIABILITY) increases as compared to DCCP. This is an advantage as the I-frames that were lost in DCCP, are now delivered which is shown by increase in PDR.

The throughput of DCCP-CR deccreases as compared to DCCP, which is a disadvantage. The reason behind it is that the due to retransmission of the I-frames, the Routing overhead increases in DCCP-CR as compared to DCCP.

9.2 Future Scope

The future scope of the work is to optimize the algorithm in a way so as to reduce the Routing Overhead and hence get better performance in terms of throughput.

Appendix A

List of Websites

http://www.isi.edu/nsnam/ns/tutorial

http://www.ffmpeg.org

http://mohittahiliani.blogspot.in

http://cs.itd.nrl.navy.mil

http://www.oldnabble.com

Appendix B

List of Papers Prepared

1. "A Survey on Congestion Control Techniques for Wireless Multimedia Sensor Networks".

2. "Transport Protocol to support Congestion Control and Controlled Reliability for Multimedia Transmission over Wireless Sensor Network".

References

[1] A. I. F., M. Tommaso, Chowdhury, and K. R., "A survey on wireless multimedia sensor networks," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 51, pp. 921–960, March 2007.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey, comput. networks," *Elsevier*, vol. 38, no. 4, pp. 393–422, 2002.

[3] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *IEEE Computer*, vol. 37, no. 8, pp. 41–49, 2004.

[4] S. Ehsan and B. Hamdaoui, "A survey on energy-efficient routing techniques with qos assurances for wireless multimedia sensor networks," *IEEE Communications Surveys and Tutorials*, vol. 14, no. 2, pp. 265–278, 2012.

[5] A. I.T., G. Zapata, M. A. Karaki, and J. Morillo-Pozo, "Wireless multimedia sensor networks: Current trends and future directions," *Sensors*, pp. 6662–6717, July 2010.

[6] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu, "Senseye: A multi-tier camera sensor network," *ACM Multimedia*, November 2005.

[7] A. I. F., Melodia, T. Chowdhury, and K. R., "Wireless multimedia sensor networks: A survey," *Wireless Sensor Networking IEEE Wireless Communications*, pp. 32–39, December 2007.

[8] B. A. Forouzan, *TCP/IP Protocol Suite*. New York: Tata Mc Graw Hill Publication, 4 ed., 2010.

[9] A. J. D. Ratnayaka, V. M. Potdar, A. Sharif, S. Sarencheh, and S. Kuruppu, "Wireless sensor network transport protocol - a state of the art," in *IEEE Comppputer Society International Conference on Broadband, Wireless Computing, Communication and Applications*, pp. 812–817, 2010.

[10] S. Kim, "Reliable transfer on wireless sensor networks," in *In SECON*, pp. 449–459, 2004.

[11] U. U. alias Sri Swathiga and C. Chandrasekar, "An efficient fuzzy based congestion control technique for wireless sensor networks," *International Journal of Computer Applications*, vol. 40, pp. 47–55, February 2012.

[12] S. R. Heikalabad, A. Ghaffari, M. A. Hadian, and H. Rasouli, “Dpcc dynamic predictive congestion control in wireless sensor networks,” *IJCSI International Journal of Computer Science Issues*, vol. 8, pp. 472–477, January 2011.

[13] C. T. Ee and R. Bajcsy, “Congestion control and fairness for many-to-one routing in sensor networks,” in *2nd International Conference on Embedded networked sensor systems*, pp. 148–161, 2004.

[14] C. Y. Wan, S. B. Eisenman, A. T. Campbell, and J. Crowcroft, “Siphon: overload traffic management using multi-radio virtual sinks in sensor networks,” in *3rdInternational Conference on Embedded networked sensor systems*, pp. 116–129, 2005.

[15] P. Levis, N. Patel, D. Culler, and S. Shenker, “Trickle a self-regulating algorithm for code propagation and maintenance in wireless sensor networks,” in *1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*, pp. 29–31, 2004.

[16] P. Seeling, F. H. Fitzek, and M. Reisslein, *Video Traces for Network Performance Evaluation.* P.O. Box 17, 3300 AA Dordrecht, The Netherlands: Springer, 1 ed., 2007.

[17] Y. G. Iyer, S. Gandham, and S. Venkatesa, “Stcp a generic transport layer protocol for wireless sensor networks,” in *14th International Conference on Computer Communications and Networks*, pp. 449–454, 2005.

[18] C. Y. Wan, S. B. Eisenman, and A. T. Campbell, “Coda congestion detection and avoidance in sensor networks,” in *1st International Conference on Embedded networked sensor systems*, pp. 266–279, 2003.

[19] S. M. Aghdam, M. Khansari, H. R. Rabiee, and M. Saleh, “Uddp a user datagram dispatcher protocol for wireless multimedia sensor networks,” in *Consumer Communications and Networking Conference (CCNC)*, pp. 765–770, 2012.

[20] G. A. Shah and O. B. Akan, *Chapter in Network and Transport Protocols.* UNESCO Encyclopedia of Life Support Systems (EOLSS), 2012.

[21] Q. min LIN, R. C. WANG, J. GUO, and L. J. SUN, “Novel congestion control approach in wireless multimedia sensor networks,” *The Journal of China Universities of Posts and Telecommunications*, vol. 18, pp. 1–8, April 2011.

[22] E. Kohler, M. Handley, S. Floyd, and J. Padhye, *Datagram Congestion Control Protocol.* Internet Engineering Task Force INTERNET-DRAFT, June 2003.

[23] E. Kohler, “Designing dccp: Congestion control without reliability,” *Computer Communication*, vol. 36, no. 4, p. 12, 2006.

[24] I. S. Chowdhury, J. Lahiry, K. C. Rahman, and S. F. Hasan, “Performance analysis of datagram congestion control protocol (dccp),” *International Journal of Computer Theory and Engineering*, vol. 3, pp. 632–637, October 2011.

[25] K. Fall and K. Varadhan, “The network simulator ns-2 documentation,” 2010.

[26] P. Sung, A. Savvides, and M. Srivastava, “Simulating networks of wireless sensors,” in *Simulation Conference Proceedings of the Winter Volume 2*, p. 1330Ũ1338, 2001.

[27] J. Klaue, B. Rathke, and A. Wolisz, “Evalvid - a framework for video transmission and quality evaluation,” in *13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, 2003.