CHAPTER 1

INTRODUCTION

1.1 Wireless Ad Hoc networks

Ad hoc is a Latin term which means “for this purpose”. A Mobile Ad hoc Network (MANET) is an association of ad hoc mobile nodes interconnected together temporarily on wireless channels . In these networks, since the communication between any two nodes of the network is done through middle nodes if they are not in the direct transmission range of each other, each node acts as a router to forward the packets as well as a host if the destination node is in the direct transmission range of that node. A wireless ad hoc network (WANET) or MANET (Mobile ad hoc network) is a decentralized type of wireless network. The network is ad hoc because it does not rely on a pre-existing infrastructure, such as routers in wired networks or access points in managed (infrastructure) wireless networks. Instead, each node participates in routing by forwarding data for other nodes, so the determination of which nodes forward data is made dynamically on the basis of network connectivity and the routing algorithm in use. In the Windows operating system, ad-hoc is a communication mode (setting) that allows computers to directly communicate with each other without a router. Wireless mobile ad hoc networks are self-configuring, dynamic networks in which nodes are free to move. Wireless networks lack the complexities of infrastructure setup and administration, enabling devices to create and join networks "on the fly" – anywhere, anytime.

MANET is collection of independent mobile nodes that can communicate to each other via radio waves. The communications between these mobile nodes are only possible if they are in radio range. In early days MANETs were called “packet radio” networks, which are sponsored by DARPA in 1970.

1.2 History on packet radio

The earliest wireless data network is called "packet radio" network, and was sponsored by Defense Advanced Research Projects Agency (DARPA) in the early 1970s. Bolt, Beranek and Newman Technologies (BBN) and SRI International designed, built, and experimented with these earliest systems. Experimenters included Robert Kahn, Jerry Burchfiel, and Ray Tomlinson. Similar experiments took place in the amateur radio community with the x25 protocol. These early packet radio systems predated the Internet, and indeed were part of the motivation of the original Internet Protocol suite. Later DARPA experiments included the Survivable Radio Network (SURAN) project, which took place in the 1980s. Another third wave of academic and research activity started in the mid-1990s with the advent of inexpensive 802.11 radio cards for personal computers. Current wireless ad-hoc networks are designed primarily for military utility. Problems with packet radios are:

- Bulky elements
- Slow data rate
- Unable to maintain links if mobility is high.

The project did not proceed much further until the early 1990s when wireless ad hoc networks are born.

1.3 Early work on MANET

In the early 1990s, Charles Perkins from SUN Microsystems USA, and Chai Keong Toh from Cambridge University separately started to work on a different Internet, that of a wireless ad hoc network. Perkins was working on the dynamic addressing issues. Toh worked on a new routing protocol, which was known as ABR – associativity-based routing. Perkins eventually proposed DSDV – Destination Sequence Distance Vector routing, which was based on distributed distance vector routing. Toh's proposal was an on-demand based routing, i.e. routes are discovered on-the-fly in real-time as and when needed. ABR was submitted to IETF as RFCs. ABR was implemented successfully into Linux OS on Lucent WaveLAN 802.11a enabled laptops and a practical ad hoc mobile network was therefore proven to be possible in 1999. Another routing protocol known as AODV was subsequently introduced and later proven and implemented in 2005.In 2007, David Johnson and Dave Maltz proposed DSR – Dynamic Source Routing.

A mobile ad hoc network (MANET) comprises a set of wireless devices that can move around freely and cooperate in relaying packets on behalf of one another. A MANET does not require a fifixed infrastructure or centralized administration. Because mobile nodes have limited transmission range, distant nodes communicate through multi hop paths. Their ease of deployment makes MANETS an attractive choice for a variety of applications. Examples include battleground communications, disaster recovery efforts, communication among a group of islands or ships, conferencing without the support of a wired infrastructure, and interactive information sharing. Unlike typical Internet applications, most applications of MANETS involve one-to-many and many-to-many communication patterns. Efficient support of group communications is critical for most ad hoc network applications. However, MANET group communications issues differ from those in wired environments for the following reasons: The wireless communications medium has variable and unpredictable characteristics and the signal strength and propagation fluctuate with respect to time and environment. Further, node mobility creates a continuously changing communication topology in which routing paths break and new ones form dynamically. Because MANETS have limited bandwidth availability and battery power, their algorithms and protocols must conserve both bandwidth and energy. Wireless devices usually use computing components— processors, memory, and I/O devices—that have low capacity and limited processing power. Thus, their communications protocols should have lightweight computational and information storage needs. MULTICASTING The multicasting communications model can facilitate effective and collaborative communication among groups. Flooding and tree-based routing represent two ends of the multicasting spectrum. Flooding is a simple approach that offers the lowest control overheads at the expense of generating very high data traffic in the wireless environment. The tree-based approach, on the other hand, generates minimal data traffic in the network, but tree maintenance and updates require many control traffic exchanges. Both flooding and tree-based approaches scale poorly. Multicast routing protocols for MANETS vary in terms of route topology, state maintenance, reliance on unicast routing, and other attributes. Instead of using a taxonomic approach to previously proposed multicasting protocols, our approach emphasizes the schemes' salient features. Most proposed multicasting protocols primarily exploit one or more specific characteristics of the MANET environment. These characteristics include variable topology, soft-state and state aggregations, knowledge of location, and communication pattern randomness. For example, mesh-based protocols exploit variable topology; stateless multicasting exploits soft-state maintenance; location-aided multicasting exploits knowledge of location; and gossip-based multicasting exploits randomness in communication and mobility.

Routing in ad hoc networks requires the discovery of multi-hop paths between the wireless mobile nodes in the network that wish to communicate. In multicast routing, packets from one sender must in general be delivered to multiple receivers making up a multicast group; any node can send packets to a group at any time, and any node can join or leave a multicast group at any time. Multicast routing is a hard problem in wired networks, and is even more challenging in ad hoc networks, due to the dynamic topology changes in the network due to node motion and wireless propagation variability, and due to the limited wireless network bandwidth and node energy resources available. Because of these factors, multicast routing protocols designed for the Internet are not well suited for the ad hoc networking environment. In recent years, a number of multicast protocols for ad hoc networks have been proposed some of which rely on reactive (on demand) mechanisms and discover routes only when they are needed for current communication, and others which rely on proactive mechanisms such as periodic neighbor sensing or flooding, or periodic routing table exchanges. In this paper, we present a performance comparison of three on-demand multicast routing protocols for ad hoc networks: the Adaptive Demand-Driven Multicast Routing protocol (ADMR) , the Multicast Ad Hoc On-Demand Distance-Vector protocol (MAODV) , and the On-Demand Multicast Routing Protocol (ODMRP) . All of these protocols contain a significant on-demand (reactive) component, but they differ in how reactive and proactive mechanisms are combined to make the complete protocol: ADMR uses source-based trees and does not utilize any periodic control packet transmissions, MAODV uses a shared group tree and uses periodic Hello messages for link break detection and periodic group leader floods for group information dissemination, and ODMRP uses a group forwarding mesh for packet forwarding and utilizes periodic flood-response cycles for multicast state creation and maintenance. In this performance evaluation, we focus on the effects of changes such as increasing number of multicast receivers or sources, application sending pattern, and increasing number of nodes in the network. We use mobile networks composed of 100 and 200 nodes, with both a single active multicast group and multiple active multicast groups in the network. We study a wide range of multicast scenarios representative of a variety of multicast applications, such as conferencing and single-source vs. multi-source groups. Although some simulation results for these protocols have been published before, the three protocols have not been compared, and prior studies have focused on smaller networks using a small set of simulation scenarios, many with only a single active multicast group. We focus here on the effects of the protocols' relative degree of on-demand behavior and their performance in different multicast scenarios.

A MANET consists of a dynamic collection of nodes with sometimes rapidly changing multi-hop topologies that are composed of relatively low-bandwidth wireless links. Since each node has a limited transmission range, not all messages may reach all the intended hosts. To provide communication through the whole network, a source-to-destination path could pass through several intermediate neighbor nodes. Unlike typical wire line routing protocols, ad-hoc routing protocols must address a diverse range of issues. The network topology can change randomly and rapidly, at unpredictable times. Since wireless links generally have lower capacity, congestion is typically the norm rather than the exception. The majority of nodes will rely on batteries, thus routing protocols must limit the amount of control information that is passed between nodes. The majority of applications for the MANET technology are in areas where rapid deployment and dynamic reconfiguration are necessary and the wire line network is not available. These include military battlefields, emergency search and rescue sites, classrooms, and conventions where participants share information dynamically using their mobile devices. These applications lend themselves well to multicast operation. In addition, within a wireless medium, it is even more crucial to reduce the transmission overhead and power consumption. Multicasting can improve the efficiency of the wireless link when sending multiple copies of messages by exploiting the inherent broadcast property of wireless transmission. However, besides the issues for any ad-hoc routing protocol listed above, wireless mobile multicasting faces several key challenges. Multicast group members move, thus precluding the use of a fifixed multicast topology. Transient loops may form during tree reconfiguration. As well, tree reconfiguration schemes should be simple to keep channel overhead low. Many multicast routing protocols have been proposed for ad-hoc networks. Comparing these protocols is typically done based on extensive simulation studies. Bagrodia et al

1.4 Classification of multicast routing protocols

1.4.1 Application Independent

Most of the routing protocols is application Independent. In this hop count is use as a metric, means if the multiple paths are available to reach the destination then the minimum hop distance will be selected.

1.4.2 Application Dependent

Application Dependent multicast protocols are meant for only specific applications for which they are designed.

1.4.3 Based on Topology

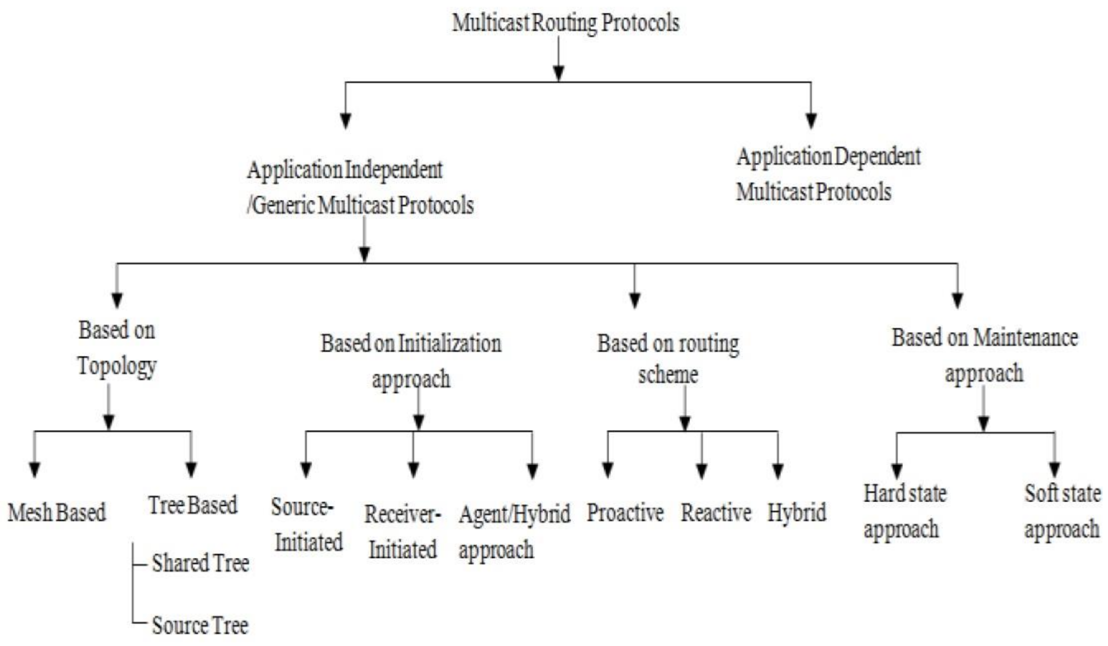


Fig 1.1 Classification of Multicast Routing Protocols

1.4.4 Tree Based

In tree-based multicast routing protocols there is only one path between source and receiver. Tree based routing protocols are not robust to operate in highly mobile environment. A tree-based multicast routing protocol establishes and maintains a shared multicast routing tree to deliver data from a source to receivers of a multicast group. Tree Based multicast routing protocol is further divided into two types-Source Tree Based and Shared Tree Based.

1.4.5 Source Tree Based

In Source Tree Based each source node maintains a separate tree and all the receivers lies under this node.

1.4.6 Shared Tree Based

In this a single tree is shared by all the sources in multicast group.

1.4.7 Mesh Based

In mesh based multicast routing protocols there is more than one path between source and receiver pair. If any link failure occurs then these redundant paths are very useful, and provide higher packet delivery ratio.

1.4.8 Based on Initialization approach

In this formation of multicast group can be initiated by source node as well as receiver node. Under this there are three approaches, namely- Source Initiated, Receiver Initiated and Agent/Hybrid approach.

1.4.9 Source Initiated

In Source Initiated the formation of multicast group is started by source node.

1.4.10 Receiver Initiated

In Receiver Initiated the formation of multicast group is started by receivers of multicast group.

1.4.11 Hybrid approach

In this construction and maintenance of multicast group done by either source node or receiver node. This approach basically combination of these two approaches i.e. Source initiated and receiver initiated.

1.5 Based on Routing scheme

1.5.1 Reactive approach

Reactive approach also known as the on - demand approach. In this route is established on the basis of demand. When a node wants to communicate with other node and there is no route, then these routing protocols will try to establish the route.

1.5.2 Proactive approach

Proactive approach also known as the table driven approach, in which each node maintain the network topology information in the form of routing tables. And these tables are periodically exchange to get the up to date view of the network.

1.5.3 Hybrid approach

Hybrid approach is the combination of both these approaches. It is designed to overcome the limitation of these approaches (Proactive and Reactive approaches).

1.6 Based on Maintenance approach

There are two main approaches which is used for maintenance of multicast topology. This can be done by either soft state approach or either hard state approach.

1.6.1 Soft- State approach

Soft- State(SS) approach is also known as “connectionless approach” .In this the control packets are flooded periodically to refresh the route, which cause a high packet delivery ratio at the cost of more control overhead.

1.6.2 Hard- State approach

Hard- State (HS) approach is also known as “connection oriented approach”. In this the control packets are only transmitted when a link breaks, due to this low control overhead but at the cost of low packet delivery ratio.

1.7 Classification of Topology based Routing Protocols

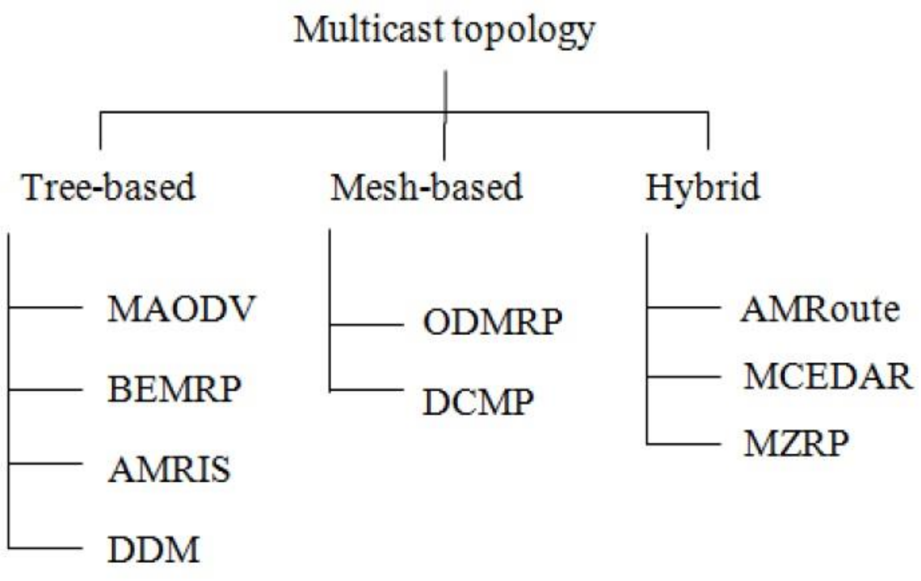


Fig 1.2 Classification of Topology based Routing Protocols

1.7.1 Tree Based Multicast Routing Protocols

The following sections describe the tree based multicasting routing protocols.

1.7.2 Multicast Ad-hoc On-demand Distance Vector (MAODV)

MAODV stands for Multicast Ad-hoc On-Demand Distance Vector Routing Protocol. This is the multicast extension of AODV.MAODV is a shared tree on-demand protocol. MAODV is proposed by Royer and Perkins in 1999. MAODV construct the shared tree more efficiently and it has low control overhead. MAODV support all the capabilities like unicast, multicast and broadcast. In MAODV each node maintains three tables-Routing Table (RT), Multicast Routing Table (MRT) and Request Table. The function of Request Table is same as that of in AODV, to store routing information. The maintenance of multicast group sequence no. is done by the group leader node. Figure 4 shows the working of MAODV.

The Multicast Ad hoc On-Demand Distance Vector (MAODV) protocol enables dynamic, self-starting, multi hop routing between participating mobile nodes wishing to join or participate in a multicast group within an ad hoc network. The membership of these multicast groups is free to change during the lifetime of the network. MAODV enables mobile nodes to establish a tree connecting multicast group members. Mobile nodes are able to respond quickly to link breaks in multicast trees by repairing these breaks in a timely manner. In the event of a network partition, multicast trees are established independently in each partition, and trees for the same multicast group are quickly connected if the network components merge. One distinguishing feature of MAODV is its use of sequence numbers for multicast groups. Each multicast group has its own sequence number, which is initialized by the multicast group leader and incremented periodically. Using these sequence numbers ensures that routes found to multicast groups are always the most current ones available. Given the choice between two routes to a multicast tree, a requesting node always selects the one with the greatest sequence number. MAODV is the multicast protocol associated with the Ad hoc On-Demand Distance Vector (AODV) routing protocol, and as such it shares many similarities and packet formats with AODV. The Route Request and Route Reply packet types are based on those used by AODV, as is the unicast Route Table. Similarly, many of the configuration parameters used by MAODV are defined by AODV. The reader is referred to the AODV Internet draft [2] for the suggested values of those parameters, as well as for details on the unicast operation of AODV

1.7.3 Bandwidth Efficient Multicast Routing Protocol (BEMRP)

Bandwidth Efficient Multicast Routing Protocol (BEMRP) is source-tree multicast routing protocol. There are three main phases-Tree-Initialization phase, Tree-Maintenance phase and Route Optimization phase. In Tree-Maintenance phase there are two schemes are used to rejoin the group. That are-Broadcast-multicast scheme and Local rejoin scheme. In Broadcast-multicast scheme the upstream node floods broadcast-multicast packet and when isolated node receives this packet and rejoin the group. In Local rejoin scheme the isolated node floods Join control packet and upstream nodes send back reply packet when isolated node receives this reply and rejoin the group.

1.7.4 Ad Hoc Multicast Routing Protocol Utilizing Increasing Id-numbers (AMRIS)

AMRIS is an On-Demand shared tree based multicast protocol. Multicast session member-id (msm-id) is assign to each node, which indicates the logical height of node in multicast delivery tree rooted at the sender that has the smallest msm-id (s-id) in the tree . Sid is defined as the node that initiated the multicast session by broadcasting the NEW-SESSION message to its surrounding nodes. In this there are two stages-Initialization and Maintenance. Figure 6 shows the working of AMRIS.

1.7.5 The Differential Destination Multicast Protocol (DDM)

The Differential Destination Multicast (DDM) is an efficient source (sender) based tree approach, designed for only small multicast group. DDM employs two types of packets: control packets and data packets, where it is understood that data packets may also contain control information. In this protocol, source node has the knowledge of all the members of the multicast group as their information is embedded in the data packets which are to be transferred to the destinations [1]. DDM has two modes-Stateless and Soft State. It has also Explicit Header means all destinations are placed in the packet headers. There are four types of control packets: JOIN, ACK, LEAVE and RSYNC

CHAPTER 2

LITERATURE SURVEY

There has been some work in reliable message delivery in networks with mobile hosts , little has been done in the area of multicast routing for mobile environments. The main reason appears to be a popular belief that similarly to the evolution of Internet routing multicast routing in mobile networks will be built on top of the unicast routing infrastructure. For this reason, most research has focused on solving the unicast routing problem with mobile endpoints. We maintain that, because of their broadcast capability, many types of mobile networks are better suited for multicast, rather than unicast, routing and, that it is more e ective to solve the multicast routing problem separately. Specially, multicast routing and packet forwarding protocols in AHNs must emphasize the following

2.1 Robustness versus Efficiency

Many multicast routing approaches rely on state in routers to keep track of multicast group members. This, coupled with the high volume of routing information exchanges and slow convergence make traditional multicast approaches untenable in highly dynamic AHNs composed of anemic (low-power, low storage capacity) hosts. Therefore, new techniques that stress rapid and robust delivery must be developed.

2.2 Active Adaptability

Hosts will migrate freely among ad hoc, fixed-infrastructure mobile, and wired networks. In order to adapt rapidly to infrastructure changes, an active networking approach can be employed: hosts can adapt in real-time by downloading appropriate multicast mechanisms.

2.3 Unlimited Mobility

Some existing multicast solutions are geared towards discrete mobility whereby periods of movement are interspersed with periods of rest. Others assume certain limits on direction, speed and number of simultaneously moving hosts. In contrast, we stress universal, unlimited mobility of all network components.

2.4 Integrated Multicast

Multicast solutions for AHNs will most likely different substantially from those for fixed networks (one of the main reasons is the marked difference in transmission rates). In order to overcome seamless and integrated multicast service, novel mechanisms must be developed for inter-operation of fixed and wireless multicast solutions. In light of these requirements, this paper identities the outstanding issues pertaining to multicast routing in fixed-infrastructure mobile and ad hoc networks. It also discusses an approach to multicast routing and packet forwarding for AHNs.

Initial support of multicast in the Internet is done by adding multicast-capable routers (mrouters) and using dedicated tunnels to facilitate multicasting packets from one mrouter to another. The job of each mrouter is to encapsulate and decapsulate each multicast packet as a regular Internet Protocol (IP) packet and send it through the tunnel to another mrouter. This set of mrouters in the Internet is called MBone (multicast backbone) . Presently, some of the existing Internet routers have been enhanced to support multicast, and there is no need to set up dedicated tunnels for them. This is called native multicast, and the Internet currently has a combination of both.

The advent of ubiquitous computing and the proliferation of portable computing devices have raised the importance of mobile and wireless networking. At the same time, the popularity of group-oriented computing has grown tremendously. However, little has been accomplished to-date in bringing together the technologies for group-oriented communication and mobile networking. In particular, most modern wireless/mobile and ad hoc networks do not provide support for multicast communication. A major challenge lies in adapting multicast communication to environments where mobility is unlimited and outages/failures are frequent. This paper motivates the need for new multicast routing protocols aimed specially at fully-mobile (ad hoc) networks. Our premise is that, due to their inherent broadcast capability, wireless networks are well- suited for multicast communication. Unlike the evolution of routing in wired networks, we believe that in ad hoc networks it is more e active to treat multicast routing as a separate problem. It was also identified that outstanding research issues pertaining to multicast routing in mobile and ad hoc networks, and discusses one possible approach to multicast routing and packet forwarding in ad hoc networks.

Multicasting is the transmission of datagrams to a group of hosts identified by a single destination address . Multicasting is intended for group-oriented computing. There are more and more applications where one-to-many dissemination is necessary. The multicast service is critical in applications characterized by the close collaboration of teams (e.g. rescue patrol, battalion, scientists, etc) with requirements for audio and video conferencing and sharing of text and images. The use of multicasting within a network has many benefits. Multicasting reduces the communication costs for applications that send the same data to multiple recipients. Instead of sending via multiple unicasts, multicasting minimizes the link bandwidth consumption, sender and router processing, and delivery delay. Maintaining group membership information and building optimal multicast trees is challenging even in wired networks. However, nodes are increasingly mobile. One particularly challenging environment for multicast is a mobile ad-hoc network (MANET). A MANET consists of a dynamic collection of nodes with sometimes rapidly changing multi-hop topologies that are composed of relatively low-bandwidth wireless links. Since each node has a limited transmission range, not all messages may reach all the intended hosts. To provide communication through the whole network, a source-to-destination path could pass through several intermediate neighbor nodes. Unlike typical wireline routing protocols, ad-hoc routing protocols must address a diverse range of issues. The network topology can change randomly and rapidly, at unpredictable times. Since wireless links generally have lower capacity, congestion is typically the norm rather than the exception. The majority of nodes will rely on batteries, thus routing protocols must limit the amount of control information that is passed between nodes. The majority of applications for the MANET technology are in areas where rapid deployment and dynamic reconfiguration are necessary and the wire line network is not available. These include military battlefields, emergency search and rescue sites, classrooms, and conventions where participants share information dynamically using their mobile devices. These applications lend themselves well to multicast operation. In addition, within a wireless medium, it is even more crucial to reduce the transmission overhead and power consumption. Multicasting can improve the efficiency of the wireless link when sending multiple copies of messages by exploiting the inherent broadcast property of wireless transmission. However, besides the issues for any ad-hoc routing protocol listed above, wireless mobile multicasting faces several key challenges. Multicast group members move, thus precluding the use of a fixed multicast topology. Transient loops may form during tree reconfiguration. As well, tree reconfiguration schemes should be simple to keep channel overhead low. Many multicast routing protocols have been proposed for ad-hoc networks, a survey can be found in. Comparing these protocols is typically done based on extensive simulation studies. Bagrodia et al. simulated several multicast routing protocols developed specifically for MANET, some tree-based, some based on a mesh structure. The reported results show that mesh protocols performed significantly better than the tree protocols in mobile scenarios. Lim and Kim evaluated multicast tree construction and proposed two new flooding methods that can improve the performance of the classic flooding method. Royer and Perkins explored the effect of the radio transmission range on the AODV protocol. They found that larger transmission ranges have many benefits (smaller trees, less frequent link breakages), but also cause more network nodes to be affected by multicast data transmission and reduce the effective bandwidth. They conclude that the transmission range should be adjusted to meet the targeted throughput while minimizing battery power consumption. Within the MANET working group at the IETF, two proposed multicast routing protocols for ad-hoc networks are AODV and ODMRP. To avoid confusion with the unicast functionality of ADOV, we will refer to the multicast operation of AODV as the MAODV protocol. To date, no side-by-side comparison of MAODV and ODMRP has been done. We decided to implement these two widely discussed multicast routing protocols for ad-hoc networks in ns-2.

Well established routing protocols do exist to offer efficient multicasting service in conventional wired networks. These protocols, having been designed for fixed networks, may fail to keep up with node movements and frequent topology changes in a MANET. As nodes become increasingly mobile, these protocols need to evolve to provide efficient service in the new environment. Therefore, adopting existing wired multicast protocols as such to a MANET, which completely lacks infrastructure, appears less promising. Host mobility increases the protocol overheads substantially. Rather, new protocols are being proposed and investigated that take issues such as topological changes into consideration. Moreover, the nodes of a MANET rely on batteries; thus routing protocols must limit the amount of control information passed between nodes. The majority of applications are in areas where rapid deployment and dynamic reconfiguration are necessary and a wire line network is not available. These include military battlefields, emergency search and rescue sites, classrooms, and conventions where participants share information dynamically using their mobile devices. These applications lend themselves well to multicast operation. In addition, within a wireless medium, it is even more crucial to reduce transmission overhead and power consumption. Multicasting can improve the efficiency of the wireless links, when sending multiple copies of messages, by exploiting the inherent broadcast property of the wireless medium when multiple mobile nodes are located within the transmission range of a node. However, besides the issues for any ad hoc routing protocol listed above, wireless mobile multicasting faces several key challenges. Multicast group members can move, thus precluding the use of a fixed multicast topology.

Regardless of the network environment, multicast communication is a very useful and efficient means of supporting group-oriented applications. This is especially the case in mobile/wireless environments where bandwidth is scarce and hosts have limited power. Example applications include audio- and video-conferencing as well as one-to-many data dissemination in critical situations such as disaster recovery or battle led scenarios. We argue that a global multicast solution for a future internetwork

CHAPTER-3

MAODV PROTOCOL

The multicast operation of the Ad hoc On-Demand Distance Vector (AODV) routing protocol (MAODV) is intended for use by mobile nodes in an ad hoc network. It offers quick adaptation to dynamic link conditions, low processing and memory overhead, and low network utilization. It creates bi-directional shared multicast trees connecting multicast sources and receivers. These multicast trees are maintained as long as group members exist within the connected portion of the network. Each multicast group has a group leader whose responsibility is maintaining the group sequence number, which is used to ensure freshness of routing information.

The Multicast Ad hoc On-Demand Distance Vector (MAODV) protocol enables dynamic, self-starting, multi hop routing between participating mobile nodes wishing to join or participate in a multicast group within an ad hoc network. The membership of these multicast groups is free to change during the lifetime of the network. MAODV enables mobile nodes to establish a tree connecting multicast group members. Mobile nodes are able to respond quickly to link breaks in multicast trees by repairing these breaks in a timely manner. In the event of a network partition, multicast trees are established independently in each partition, and trees for the same multicast group are quickly connected if the network components merge. One distinguishing feature of MAODV is its use of sequence numbers for multicast groups. Each multicast group has its own sequence number, which is initialized by the multicast group leader and incremented periodically. Using these sequence numbers ensures that routes found to multicast groups are always the most current ones available. Given the choice between two routes to a multicast tree, a requesting node always selects the one with the greatest sequence number. MAODV is the multicast protocol associated with the Ad hoc

On-Demand Distance Vector (AODV) routing protocol, and as such it shares many similarities and packet formats with AODV. The Route Request and Route Reply packet types are based on those used by AODV, as is the unicast Route Table. Similarly, many of the configuration parameters used by MAODV are defined by AODV. The reader is referred to the AODV Internet draft for the suggested values of those parameters, as well as for details on the unicast operation of AODV.

Route Requests (RREQs), Route Replies (RREPs), Multicast Activations (MACTs), and Group Hellos (GRPHs) are the message types utilized by the multicast operation AODV. RREQs and RREPs are handled as specified in , except for certain procedures controlled by new flags. These message types are handled by UDP, and normal IP header processing applies. So, for instance, the requesting node is expected to use its IP address as the source IP address for the messages. The range of dissemination of broadcast RREQs can be indicated by the TTL in the IP header. Fragmentation is typically not required.

As long as the multicast group members remain connected (within a “multicast tree''), MAODV does not play any role. When a node either wishes to join a multicast group or find a route to a multicast group, the node uses a broadcast RREQ to discover a route to the multicast tree associated with that group. For join requests, a route is determined when the RREQ reaches a node that is already a member of the multicast tree, and the node's record of the multicast group sequence number is at least as great as that contained in the RREQ. For non-join requests, any node with a current route to the multicast tree may respond to the RREQ. A current route is defined as an unexpired multicast route table entry whose associated sequence number for the multicast group is at least as great as that contained in the RREQ. The route to the multicast tree is made available by unicasting a RREP back to the source of the RREQ. Since each node receiving the request caches a route back to the source of the request, the RREP can be unicast back to the source from any node able to satisfy the request. Once the source node has waited the discovery period to receive RREPs, it selects the best route to the multicast tree and unicasts the next hop along that route a MACT message. This message activates the route. Nodes monitor the link status of next hops on the multicast tree. When a link break on the multicast tree is detected, the tree branch should be immediately repaired through the use of the RREQ/RREP/MACT messages. A multicast group leader is associated with each multicast group. The primary responsibility of this node is the initialization and maintenance of the group sequence number. A Group Hello message is periodically broadcast across the network by the multicast group leader. This message carries a multicast group and group sequence number and corresponding group leader IP address. This information is used for disseminating updated group sequence numbers throughout the multicast group and for repairing multicast trees after a previously disconnected portion of the network containing part of the multicast tree becomes reachable once again.

3.1 Route Tables

MAODV is a routing protocol, and it deals with route table management. Route table information must be kept even for ephemeral routes, such as are created to temporarily store reverse paths towards nodes originating RREQs. MAODV uses the following fields with each route table entry, which are based on the route table defined in the AODV Internet Draft

1. Destination IP Address
2. Destination Sequence Number
3. Hop Count (number of hops needed to reach destination)
4. Last Hop Count
5. Next Hop

6. Next Hop Interface
7. List of Precursors
8. Lifetime (expiration or deletion time of the route)
9. Routing Flags

The direction of the link is relative to the location of the group leader, i.e. UPSTREAM is a next hop towards the group leader, and DOWNSTREAM is a next hop away from the group leader. A node on the multicast tree must necessarily have only one UPSTREAM link. The IP Address of a Next Hop MUST NOT be used to forward multicast messages until after a MACT message has activated the route The Next Hop Interface fields in the Route Table and Next Hop field of the Multicast Route Table are used for recording the outgoing interface on which the next hop can be reached . Nodes MAY also maintain a Group Leader Table, which is an association between multicast groups and their corresponding group leaders. The fields of the group leader table are the following:

- Multicast Group IP Address
- Group Leader IP Address

3.2 MAODV Terminology

This protocol specification uses conventional meanings for capitalized words such as MUST, SHOULD, etc., to indicate requirement levels for various protocol features. This section defines other terminology used with MAODV that is not already defined in.

3.3 Group leader

A node which is a member of the given multicast group and which is typically the first such group member in the connected portion of the network. This node is responsible for initializing and maintaining the multicast group destination sequence number.

3.4 Group leader table

The table where ad hoc nodes keep information concerning each multicast group and its corresponding group leader. There is one entry in the table for each multicast group for which the node has received a Group Hello.

3.5 Multicast tree

The tree containing all nodes which are members of the multicast group and all nodes which are needed to connect the multicast group members.

3.6 Multicast route table

The table where ad hoc nodes keep routing (including next hops) information for various multicast groups.

3.7 Reverse route

A route set up to forward a reply (RREP) packet back to the source from the destination or from an intermediate node having a route to the destination.

3.8 Maintaining Multicast Tree Utilization Records

For each multicast tree to which a node belongs, either because it is a member of the group or because it is a router for the multicast tree, the node maintains a list of next hops i.e. those 1-hop neighbors that are likewise a part of the multicast tree. This list of next hops is used for forwarding messages received for the multicast group. A node forwards a multicast message to every such next hop, except that neighbor from which the message arrived. If there are multiple next hops, the forwarding operation MAY be performed by broadcasting the multicast packet to the node's neighbors; only the neighbors that belong to the multicast tree and have not already received the packet continue to forward the multicast packet.

3.9 Generating Route Requests

A node sends a RREQ either when it determines that it should be a part of a multicast group, and it is not already a member of that group, or when it has a message to send to the multicast group but does not have a route to that group. If the node wishes to join the multicast group, it sets the `J' flag in the RREQ; otherwise, it leaves the flag unset. The destination address of the RREQ is always set to the multicast group address. If the node knows the group leader and has a route to it, the node MAY place the group leader's address in the Multicast Group Leader extension and unicast the RREQ to the corresponding next hop for that destination. Otherwise, if the node does not have a route to the group leader, or if it does not know who the multicast group leader is, it broadcasts the RREQ and does not include the extension field.

After transmitting the RREQ, the node waits for the reception of a RREP. The node may resend the RREQ up to RREQ_RETRIES additional times if a RREP is not received. If a RREQ was unicast to a group leader and a RREP is not received within RREP_WAIT_TIME milliseconds, the node broadcasts subsequent RREQs for that multicast group across the network. If a RREP is not received after RREQ_RETRIES additional requests, the node may assume that there are no other members of that particular group within the connected portion of the network. If it wanted to join the multicast group, it then becomes the multicast group leader for that multicast group and initializes the sequence number of the multicast group. Otherwise, if it only wanted to send packets to that group without actually joining the group, it drops the packets it had for that group and aborts the session. When the node wishes to join or send a message to a multicast group, it first consults its Group Leader Table. Based on the existence of an entry for the multicast group in this table, the node then formulates and sends the RREQ.

3.10 Controlling Route Request broadcasts

To prevent unnecessary network-wide broadcasts of RREQs, the source node SHOULD use an expanding ring search technique. When a RREP is received, the Hop Count to the group leader used in the RREP packet is remembered as Last Hop Count for that node in the routing table. When a new route to the same multicast group is required at a later time (e.g., upon a link break in the multicast tree), the TTL in the RREQ IP header is initially set to this Last Hop Count plus TTL_INCREMENT. Thereafter, following each timeout the TTL is incremented by TTL_INCREMENT until TTL = TTL_THRESHOLD is reached.

3.11 Receiving Route Requests

When a node receives a RREQ, the node checks whether the 'J' flag of the RREQ is set. If the 'J' flag is set, the node can only respond if it is a member of the multicast tree for the indicated multicast group, and if its record of the multicast group sequence number is at least as great as that contained in the RREQ. If the 'J' flag is not set, then the node can respond if it has an unexpired route to the multicast group and the above multicast group sequence number criteria is met. If the node does not meet either of these conditions, it rebroadcasts the RREQ from its interface(s) but using its own IP address in the IP header of the outgoing RREQ. The Destination Sequence Number in the RREQ is updated to the maximum of the existing Destination Sequence Number in the RREQ and the multicast group sequence number in the multicast route table (if an entry exists) of the current node. The TTL or hop limit field in the outgoing IP header is decreased by one. The Hop Count field in the broadcast RREQ message is incremented by one, to account for the new hop through the intermediate node. The node always creates or updates a reverse route. This reverse route would be needed in case the node receives an eventual RREP back to the node which originated the

RREQ (identified by the Source IP Address). In addition to creating or updating the route table entry for the source node, the node receiving the RREQ also creates a next hop entry for the multicast group in its multicast route table. If no entry exists for the multicast group, it creates one, and then places the node from which it received the RREQ as next hop for that group. It leaves the Activated flag associated with this next hop unset. The direction for this next hop entry is DOWNSTREAM.

3.12 Generating Route Replies

If a node receives a join RREQ for a multicast group, and it is already a member of the multicast tree for that group, the node updates its multicast route table and then generates a RREP message. The Source and Destination IP Addresses in RREQ message are copied to corresponding fields in the RREP message. The RREP contains the current sequence number for the multicast group and the IP address of the group leader. Furthermore, the node initializes the Hop Count field of the RREP to zero. Additional information about the multicast group is entered into the Multicast Group Information extension. It unicasts the RREP back to the node indicated by the Source IP Address field of the received RREQ. A node can respond to a join RREQ only if it is a member of the multicast tree. If a node receives a multicast route request that is not a join message, it can reply if it has a current route to the multicast tree. Otherwise it continues forwarding the request. If a node receives a join RREQ for a multicast group and it is not already a member of the multicast tree for that group, it rebroadcasts the RREQ to its neighbors. When a node receives a RREQ for a multicast group which has its own IP address in the Destination IP address of the IP header, that means that the source node expects this destination node to be the multicast group leader. In this case, if the node is in fact not the group leader, it can simply ignore the RREQ. The source node will time out after

RREP_WAIT_TIME milliseconds and will broadcast a new RREQ without the group leader address specified.

3.13 Forwarding Route Replies

If an intermediate node receives a RREP in response to a RREQ that it has transmitted (or retransmitted on behalf of some other node), it creates a multicast group next hop entry for the node from which it received the RREP. The direction of this next hop is UPSTREAM, and the Activated flag is left unset. Additionally, the node updates the Lifetime field in the route table entry associated with the node from which it received the RREP. It then increments the Hop Count and Multicast Group Hop Count fields or the RREP and forwards this packet along the path to the source of the RREQ. When the node receives more than one RREP for the same RREQ, it saves the route information with the greatest sequence number, and beyond that the lowest hop count; it discards all other RREPs. This node forwards the first RREP towards the source of the RREQ, and then forwards later RREPs only if they have a greater sequence number or smaller metric.

3.14 Route Activation

When a node broadcasts a RREQ message, it is likely to receive more than one reply since any node in the multicast tree can respond. The RREP message sets up route pointers as it travels back to the source node. If the request is a join request, these route pointers may eventually graft a branch onto the multicast tree. Also, because multicast data packets may be transmitted as broadcast traffic, the route to the multicast tree must be explicitly selected. Otherwise, each node with a route to the tree which receives a multicast data packet will rebroadcast the packet, resulting in an inefficient use of network bandwidth. Hence, it is necessary to activate only one of the routes created by the RREP messages. The RREP containing the largest destination sequence number is chosen to be the branch added to the multicast tree (or the path to the multicast tree, if the request was a non-join). In the event that a node receives more than one RREP with the same (largest) sequence number, it selects the first one with the smallest hop count, i.e., the shortest distance to a member of the multicast tree. After waiting RREP_WAIT_TIME milliseconds, the node must select the route it wishes to use as its link to the multicast tree. This is accomplished by sending a Multicast Activation (MACT) message. The Destination IP Address field of the MACT packet is set to the IP address of the multicast group. If the node is joining the multicast group, it sets the join flag of this message. The node unicasts this message to the selected next hop, effectively activating the route. It then sets the Activated flag in the next hop Multicast Route Table entry associated with that node. After receiving this message, the node to which the MACT was sent activates the route entry for the link in its multicast route table, thereby finalizing the creation of the tree branch. All neighbors not receiving this message time out and delete that node as a next hop for the multicast group in their route tables, having never activated the route entry for that next hop. Two scenarios exist for a neighboring node receiving the MACT message. If this node was previously a member of the multicast tree, it does not propagate the MACT message any further. However, if the next hop selected by the source node's MACT message was not previously a multicast tree member, it will have propagated the original RREQ further up the network in search of nodes which are tree members. Thus it is possible that this node also received more than one RREP. When the node receives a MACT selecting it as the next hop, it unicasts its own MACT to the node it has chosen as its next hop, and so on up the tree, until a node which was already a part of the multicast tree is reached.

3.15 Multicast Tree Pruning

A multicast group member can revoke its member status at any time. However, it can only actually leave the multicast tree if it is not a tree router for any other nodes in the multicast group (i.e., if it is a leaf node). If a node wishing to leave the multicast group is a leaf node, it unicasts to its next hop on the tree a MACT message with the 'P' flag set and with the Destination IP Address set to the IP address of the multicast group. It then deletes the multicast group information for that group from its multicast route table. When its next hop receives this message, it deletes the sending node's information from its list of next hops for the multicast tree. If the removal of the sending node causes this node to become a leaf node, and if this node is also not a member of the multicast group, it may in turn prune itself by sending its own MACT message up the tree. When the multicast group leader wishes to leave the multicast group, it proceeds in a manner similar to the one just described. If it is a leaf node, it may leave the group and unicast a prune message to its next hop. The next hop acts in the manner described in Section 9.10, since the prune message is coming from its upstream neighbor. Otherwise, if the group leader is not a leaf node, it may not prune itself from the tree. It selects one of its next hops and unicasts to it the MACT with set 'G' flag.

3.16 Repairing a Broken Link

Branches of the multicast tree become invalid if a broken link results in an infinite metric being associated with the next hop route table entry. When a broken link is detected between two nodes on the multicast tree, the two nodes should delete the link from their list of next hops for the multicast group. The node downstream of the break (i.e., the node which is further from the multicast group leader) is responsible for initiating the repair of the broken link. In order to repair the tree, the downstream node broadcasts a RREQ with destination IP address set to the IP address of the multicast group and with the 'J' flag set. The destination sequence number of the RREQ is the last known sequence number of the multicast group. The node also includes the Multicast Group Leader Extension. The Multicast Group Hop Count field of this extension is set to the distance of the source node from the multicast group leader. A node MUST have a hop count to the multicast group leader less than or equal to the indicated value in order to respond. This hop count requirement prevents nodes on the same side of the break as the node initiating the repair from replying to the RREQ. The RREQ is broadcast using an expanding rings search, as described in Section 9.2.1. Because of the high probability that other nearby nodes can be used to rebuild the route, the original RREQ is broadcast with a TTL (time to live) field value equal to TTL_INCREMENT plus the Multicast Group Hop Count. In this way, the effects of the link breakage may be localized. If no reply is received within RREP_WAIT_TIME milliseconds, the node SHOULD increment the TTL of each successive RREQ by TTL_INCREMENT, until either a route is determined, or TTL_THRESHOLD is reached. After this point, if a route is still not discovered, each additional RREQ is broadcast with TTL = NET_DIAMETER. Up to RREQ_RETRIES additional broadcasts may be attempted after TTL = NET_DIAMETER is reached.

A node receiving this RREQ can respond if it meets the following conditions, It is a member of the multicast tree. Its record of the multicast group sequence number is at least as great as that contained in the RREQ. Its hop count to the multicast group leader is less than or equal to the contained in the Multicast Group Hop Count extension field.

At the end of the discovery period, the node selects its next hop and unicasts a MACT message to that node to activate the link, as described in Section 9.6. Since the node was repairing a tree break, it is likely that it is now a different distance from the group leader than it was before the break. If this is the case, it must inform its

DOWNSTREAM next hops of their new distance from the group leader. It does this by broadcasting a MACT message with the 'U' flag set, and the Hop Count field set to the node's new distance from the group leader. This 'U' flag indicates that multicast tree nodes should update their distance from the group leader. When a node on the multicast tree receives the MACT message with the 'U' flag set, it determines whether this packet arrived from its UPSTREAM neighbor. If it did not, the node discards the packet. Otherwise, it increments the Hop Count value contained in the MACT packet and updates it distance to the group leader to be this node value. If this node has one or more downstream next hops, it in turn must send a MACT message with a set 'U' flag to its next hops, and so on. When a link break occurs, it is possible that the tree will be repaired through different intermediate nodes. If the node UPSTREAM of the break is not a group member, and if the loss of that link causes it to become a leaf node, it sets a prune timer to wait for the link to be repaired. If, when this timer expires, the node has not received a MACT message selecting it to be a part of the repaired tree branch, it prunes itself from the tree by sending a MACT with set 'P' flag to its next hop, as previously described.

3.17 Tree Partitions

It is possible that after a link breaks, the tree cannot be repaired due to a network partition. If the node attempting to repair a tree link break does not receive a response after RREQ_RETRIES attempts, it can be assumed that the network has become partitioned and the multicast tree cannot be repaired at this time. In this situation, if the node which initiated the route rebuilding is a multicast group member, it becomes the new multicast group leader for its part of the multicast tree partition. It increments the group sequence number and then broadcasts a Group Hello (GRPH) for this multicast group. The 'U' flag in the GRPH is set, indicating that there has been a change in the group leader information. All nodes receiving this message update their group leader table to indicate the new group leader information. Nodes which are a part of the multicast tree also update the group leader and sequence number information for that group in their multicast route table. On the other hand, if the node which had initiated the repair is not a multicast group member, there are two possibilities. If it only has one next hop for the multicast tree, it prunes itself from the tree by unicasting a MACT message, with the 'P' flag set, to its next hop. The node receiving this message notes that the message came from its upstream link, i.e., from a node that is closer to the group leader than it is. If the node receiving this message is a multicast group member, it becomes the new group leader and broadcasts a GRPH message as indicated above. Otherwise, if it is not a multicast group member and it only has one other next hop link, it similarly prunes itself from the tree. This process continues until a multicast group member is reached. The other possibility is that the node which initiated the rebuilding is not a group member and has more than one next hop for the tree. In this case, it cannot prune itself, since doing so would partition the tree. It instead selects one of its next hops and unicasts a MACT with the 'G' flag set to that node. This flag indicates that the next group member to receive this message should become the new group leader. It then changes the direction of that link to be UPSTREAM. If the node receiving the MACT is a group member, then this node becomes the new group leader. Otherwise, the node unicasts its own MACT message with the 'G' flag set to one of its next hops, and changes the direction of that link. Once a group member is reached, the new group leader is determined.

3.18 Reconnecting Two Trees

In the event that a link break cannot be repaired, the multicast tree remains partitioned until the two parts of the network become connected once again. A node from one partition of the network knows that it has come into contact with a node from the other partition of the network by noting the difference in the GRPH message multicast group leader information. The multicast group leader with the lower IP address initiates the tree repair. For the purposes of this explanation, call this node GL1. GL1 unicasts a RREQ with both the 'J' and 'R' flags set to the group leader of the other network partition (GL2), using the node from which it received the GRPH as the next hop. This RREQ contains the current value of GL1's multicast group sequence number. If any node that receives the RREQ is a member of GL2's multicast tree, it MUST forward the RREQ along its upstream link, i.e. towards GL2. This prevents any loops from being formed after the repair. Upon receiving the RREQ, GL2 takes the larger of its and the received multicast group sequence number, increments this value by one, and responds with a RREP. This is the group leader which becomes the leader of the reconnected multicast tree. The 'R' flag of the RREP is set, indicating that this RREP is in response to a repair request. As the RREP is propagated back to GL1, nodes add the incoming and outgoing links to the multicast route table next hop entries if these entries do not already exist. The nodes also activate these entries, thereby adding the branch on to the multicast tree. If a node that was previously a member of GL1's tree receives the RREP, it MUST forward the packet along its link to its previous group leader (GL1). It then updates its group leader information to reflect GL2 as the new group leader, changes the direction of the next hop link associated with GL1 to DOWNSTREAM, and sets the direction of the link on which it received the RREP to UPSTREAM. When GL1 receives the RREP, it updates its group leader information and sets the link from which it received the RREP as its upstream link. The tree is now reconnected. The next time GL2 broadcasts a GRPH, it sets the `U' flag to indicate that there is a change in the group leader information and group members should update the corresponding information. All network nodes update their group leader table to reflect the new group leader information.

CHAPTER-4

SOFTWARE REQUIREMENTS

4.1 Ubuntu 16.04 Operating System

Ubuntu is a free and open-source Linux distribution based on Debian. Ubuntu is officially released in three editions: Desktop, Server and Core (for IoT devices and robots). Ubuntu is a popular operating system for cloud computing, with support for Open Stack.

Ubuntu is released every six months, with long-term support (LTS) releases every two years. The latest release is 18.10 ("Cosmic Cuttlefish"), and the most recent long-term support release is 18.04 LTS ("Bionic Beaver"), which is supported until 2028.Ubuntu is developed by Canonical and the community under a meritocratic governance model. Canonical provides security updates and support for each Ubuntu release, starting from the release date and until the release reaches its designated end-of-life (EOL) date. Canonical generates revenue through the sale of premium services related to Ubuntu. Ubuntu is named after the African philosophy of ubuntu, which Canonical translates as "humanity to others" or "I am what I am because of who we all are".

4.2 History

Ubuntu is built on Debian's architecture and infrastructure, and comprises Linux server, desktop and discontinued phone and tablet operating system versions. Ubuntu releases updated versions predictably every six months and each release receives free support for nine months (eighteen months prior to 13.04)with security fixes, high-impact bug fixes and conservative, substantially beneficial low-risk bug fixes. The first release was in October 2004.

Current long-term support (LTS) releases are supported for five years, and are released every two years. LTS releases get regular point releases with support for new hardware and integration of all the updates published in that series to date.

Ubuntu packages are based on packages from Debian's unstable branch. Both distributions use Debian's deb package format and package management tools (e.g. APT and Ubuntu Software). Debian and Ubuntu packages are not necessarily binary compatible with each other, however, so packages may need to be rebuilt from source to be used in Ubuntu. Many Ubuntu developers are also maintainers of key packages within Debian. Ubuntu cooperates with Debian by pushing changes back to Debian, although there has been criticism that this does not happen often enough. Ian Murdock, the founder of Debian, had expressed concern about Ubuntu packages potentially diverging too far from Debian to remain compatible. Before release, packages are imported from Debian unstable continuously and merged with Ubuntu-specific modifications. One month before release, imports are frozen, and packagers then work to ensure that the frozen features interoperate well together.

Ubuntu is currently funded by Canonical Ltd. On 8 July 2005, Mark Shuttle worth and Canonical announced the creation of the Ubuntu Foundation and provided an initial funding of US$10 million. The purpose of the foundation is to ensure the support and development for all future versions of Ubuntu. Mark Shuttle worth describes the foundation goal as to ensure the continuity of the Ubuntu project.

On 12 March 2009, Ubuntu announced developer support for third-party cloud management platforms, such as those used at Amazon EC2.

GNOME 3 has been the default GUI for Ubuntu Desktop since Ubuntu 17.10,while Unity is still the default in older versions, including all current LTS versions except 18.04 LTS. However, a community-driven fork of Unity 8, called Yunit, has been created to continue the development of Unity.[non-primary source needed] Shuttle worth wrote on 8 April 2017, "We will invest in Ubuntu GNOME with the intent of delivering a fantastic all-GNOME desktop. We're helping the Ubuntu GNOME team, not creating something different or competitive with that effort. While I am passionate about the design ideas in Unity, and hope GNOME may be more open to them now, I think we should respect the GNOME design leadership by delivering GNOME the way GNOME wants it delivered. Our role in that, as usual, will be to make sure that upgrades, integration, security, performance and the full experience are fantastic. "Shuttle worth also mentioned that Canonical will cease development for Ubuntu Phone, Tablet, and convergence.

32-bit i386 processors have been supported up to Ubuntu 18.04, but users "will not be allowed to upgrade to Ubuntu 18.10 as dropping support for that architecture is being evaluated".

4.3 Features

A default installation of Ubuntu contains a wide range of software that includes Libre Office, Firefox, Thunderbird, Transmission, and several lightweight games such as Sudoku and chess. Many additional software packages are accessible from the built in Ubuntu Software (previously Ubuntu Software Center) as well as any other APT-based package management tools. Many additional software packages that are no longer installed by default, such as Evolution, GIMP, Pidgin, and Synaptic, are still accessible in the repositories still installable by the main tool or by any other APT-based package management tool. Cross-distribution snap packages and flat packs are also available, that both allow installing software, such as some of Microsoft's software, in most of the major Linux operating systems (such as any currently supported Ubuntu version and in Fedora). The default file manager is GNOME Files, formerly called Nautilus.

Ubuntu operates under the GNU General Public License (GPL) and all of the application software installed by default is free software. In addition, Ubuntu installs some hardware drivers that are available only in binary format, but such packages are clearly marked in the restricted component.

4.4 Security

Ubuntu aims to be secure by default. User programs run with low privileges and cannot corrupt the operating system or other users' files. For increased security, the sudo tool is used to assign temporary privileges for performing administrative tasks, which allows the root account to remain locked and helps prevent inexperienced users from inadvertently making catastrophic system changes or opening security holes. Polkit is also being widely implemented into the desktop.

Most network ports are closed by default to prevent hacking. A built-in firewall allows end-users who install network servers to control access. A GUI (GUI for Uncomplicated Firewall) is available to configure it. Ubuntu compiles its packages using GCC features such as PIE and buffer overflow protection to harden its software. These extra features greatly increase security at the performance expense of 1% in 32-bit and 0.01% in 64-bit.Ubuntu also supports full disk encryption as well as encryption of the home and Private directories

4.5 Steps to install Ubuntu 16.04

In case your hardware uses UEFI then you should modify the EFI settings and disable Secure Boot feature. If your computer has no other Operating System already installed and you plan to use a Windows variant alongside Ubuntu 16.04/16.10, you should first install Microsoft Windows and then proceed with Ubuntu 16.04 installation.

In this particular case, on Windows installation steps, when formatting the hard disk, you should allocate a free space on the disk with at least 20 GB in size in order use it later as a partition for Ubuntu installation.

Step 1: Prepare Windows Machine for Dual-Boot

The first thing you need to take care is to create a free space on the computer hard disk in case the system is installed on a single partition.

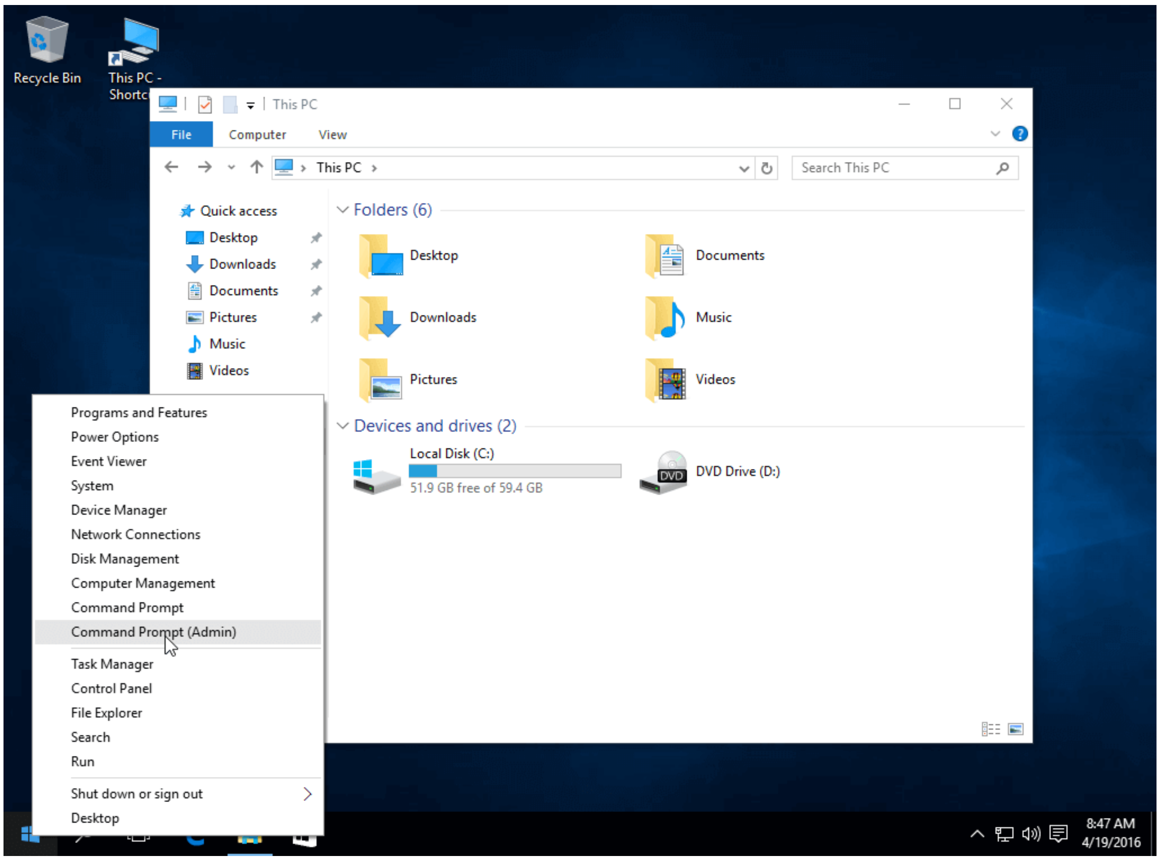
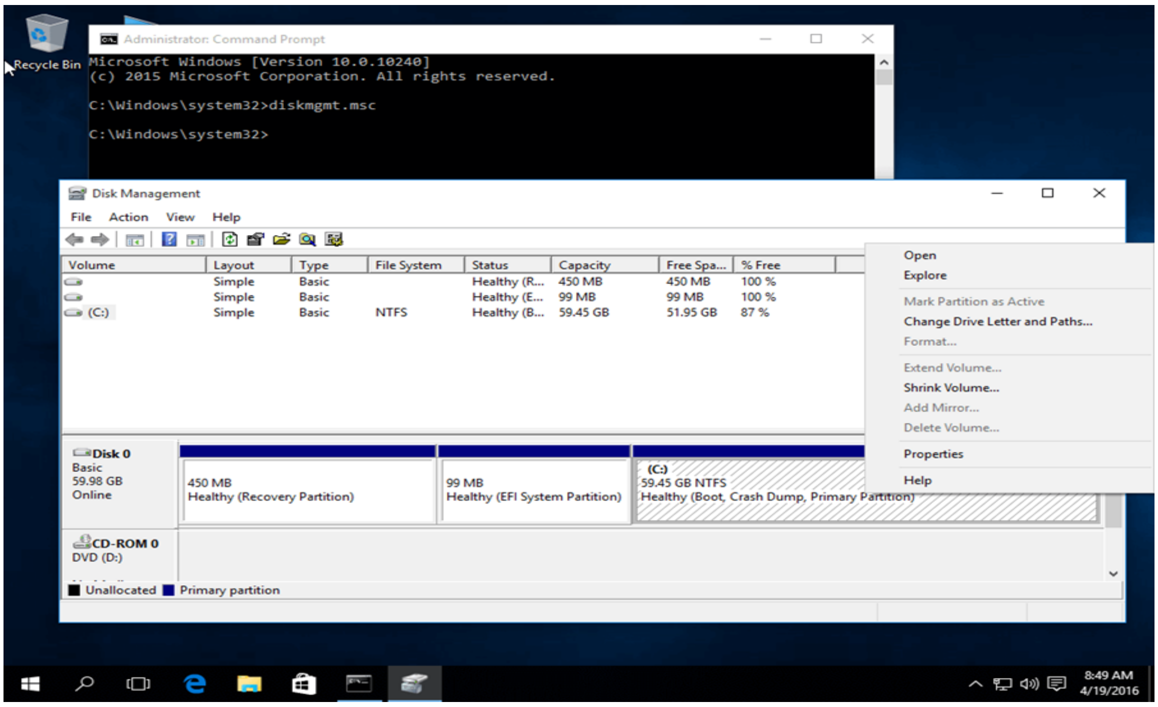


Fig 4.1 Figure shows space allocated to windows

Login to your Windows machine with an administrative account and right click on the Start Menu -> Command Prompt (Admin) in order to enter Windows Command Line.



```
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32>diskmgmt.msc

C:\Windows\system32>
```

Volume	Layout	Type	File System	Status	Capacity	Free Spa...	% Free
	Simple	Basic		Healthy (R...	450 MB	450 MB	100 %
	Simple	Basic		Healthy (E...	99 MB	99 MB	100 %
(C:)	Simple	Basic	NTFS	Healthy (B...	59.45 GB	51.95 GB	87 %

Disk 0
Basic
59.98 GB
Online

450 MB
Healthy (Recovery Partition)

99 MB
Healthy (EFI System Partition)

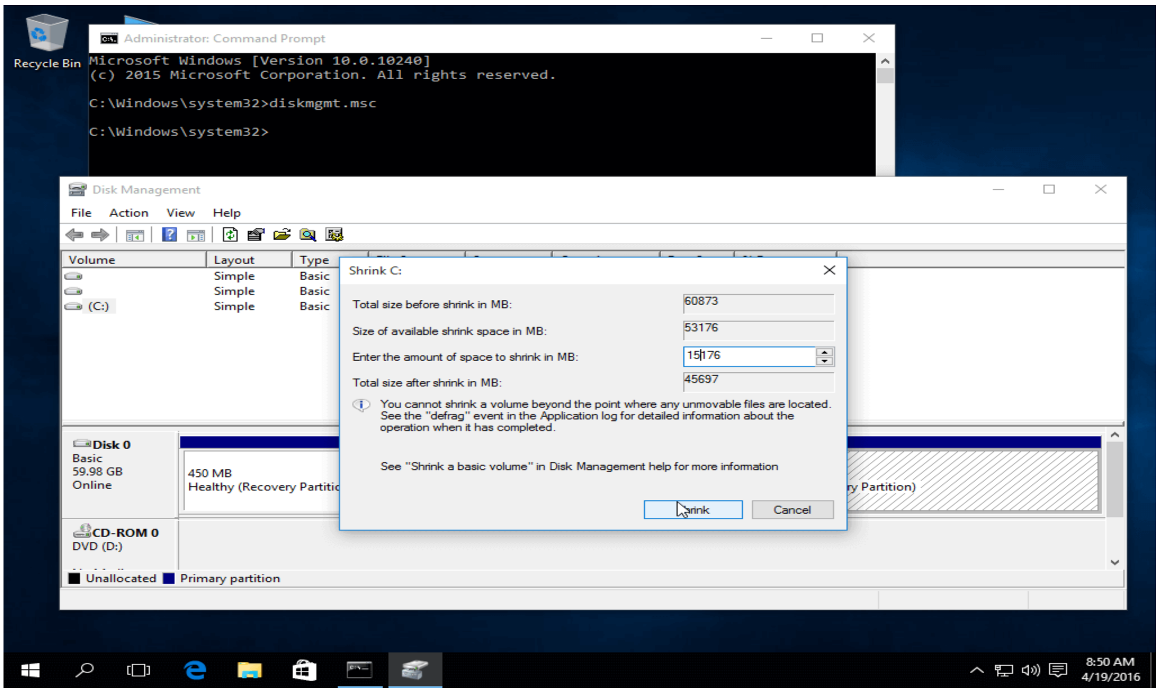
(C:)
59.45 GB NTFS
Healthy (Boot, Crash Dump, Primary Partition)

CD-ROM 0
DVD (D:)

■ Unallocated ■ Primary partition

Fig 4.2 Figure shows Disk Management tool

On Shrink C: enter a value on space to shrink in MB (use at least 20000 MB depending on the C: partition size)



```
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32>diskmgmt.msc

C:\Windows\system32>
```

Shrink C:

Total size before shrink in MB:	60873
Size of available shrink space in MB:	53176
Enter the amount of space to shrink in MB:	15176
Total size after shrink in MB:	45697

You cannot shrink a volume beyond the point where any unmovable files are located. See the "defrag" event in the Application log for detailed information about the operation when it has completed.

See "Shrink a basic volume" in Disk Management help for more information

Fig 4.3 Figure shows size allocation to LINUX

Hit Shrink to start partition resize as illustrated below (the value of space shrink from below image is lower and only used for demonstration purposes).

Once the space has been resized you will see a new unallocated space on the hard drive. Leave it as default and reboot the computer in order to proceed with Ubuntu 16.04 installation.

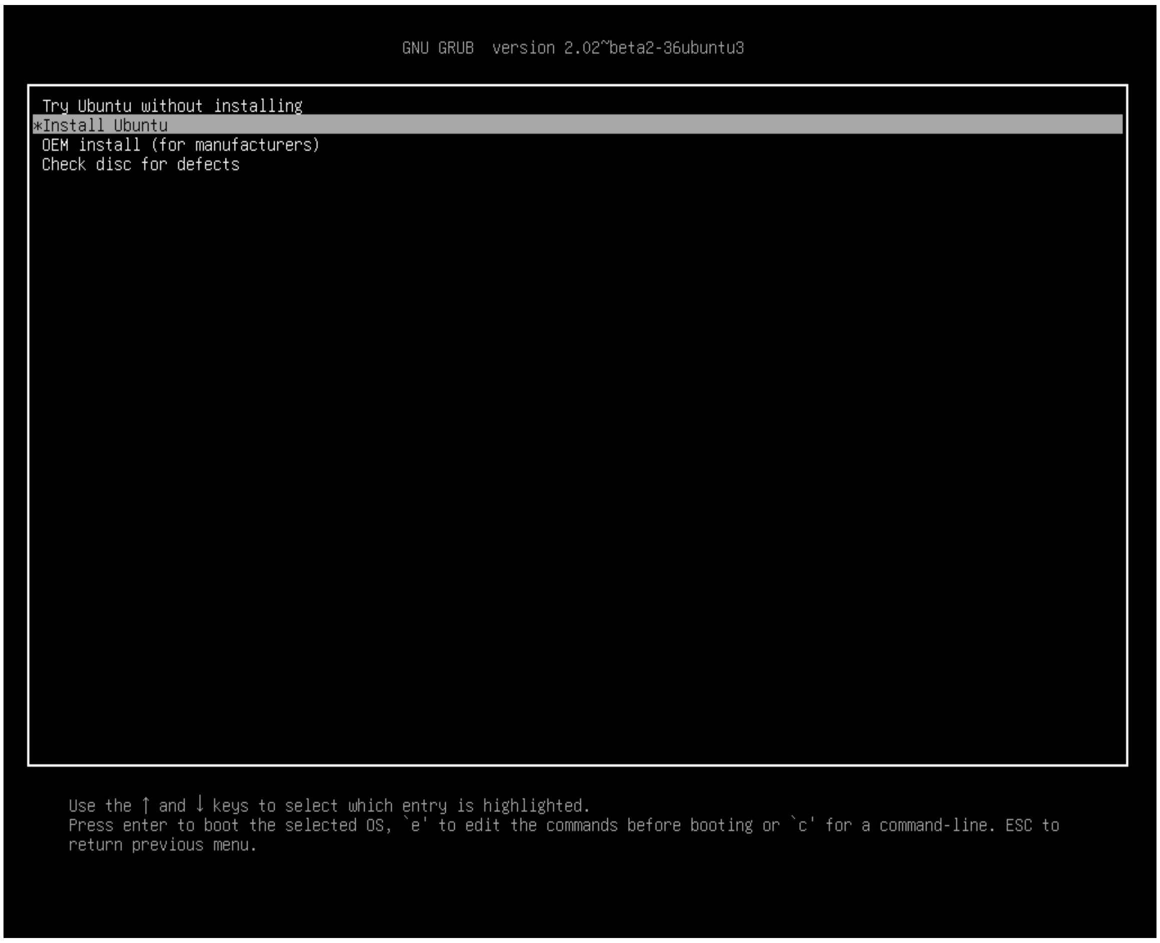
Step 2: Install Ubuntu 16.04 with Windows Dual-Boot

Now it's time to install Ubuntu 16.04. Go the download link from the topic description and grab Ubuntu Desktop 16.04 ISO image.

Burn the image to a DVD or create a bootable USB stick using a utility such as Universal USB Installer (BIOS compatible) or Rufus (UEFI compatible).

Place the USB stick or DVD in the appropriate drive, reboot the machine and instruct the BIOS/UEFI to boot-up from the DVD/USB by pressing a special function key (usually F12, F10 or F2 depending on the vendor specifications).

Once the media boot-up a new grub screen should appear on your monitor. From the menu select Install Ubuntu and hit Enter to continue.



```
GNU GRUB  version 2.02~beta2-36ubuntu3

 Try Ubuntu without installing
*Install Ubuntu
 OEM install (for manufacturers)
 Check disc for defects

   Use the ↑ and ↓ keys to select which entry is highlighted.
   Press enter to boot the selected OS, `e' to edit the commands before booting or `c' for a command-line. ESC to
   return previous menu.
```

Fig 4.4 Shows Linux GRUB

After the boot media finishes loading into RAM you will end-up with a completely functional Ubuntu system running in live-mode.

On the Launcher hit on the second icon from top, Install Ubuntu 16.04 LTS, and the installer utility will start. Choose the language you wish to perform the installation and click on Continue button to proceed further.

Next, leave both options from Preparing to Install Ubuntu unchecked and hit on Continue button again.

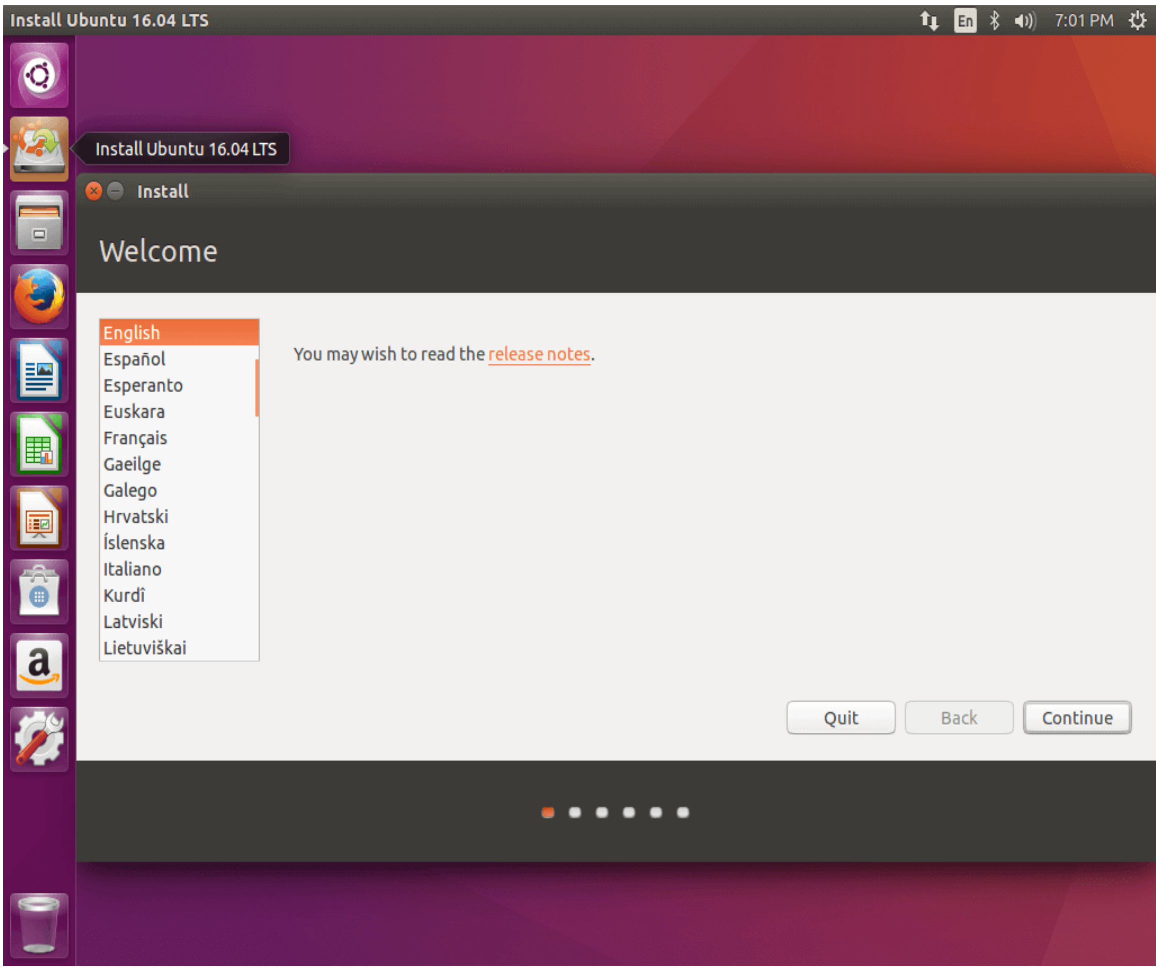
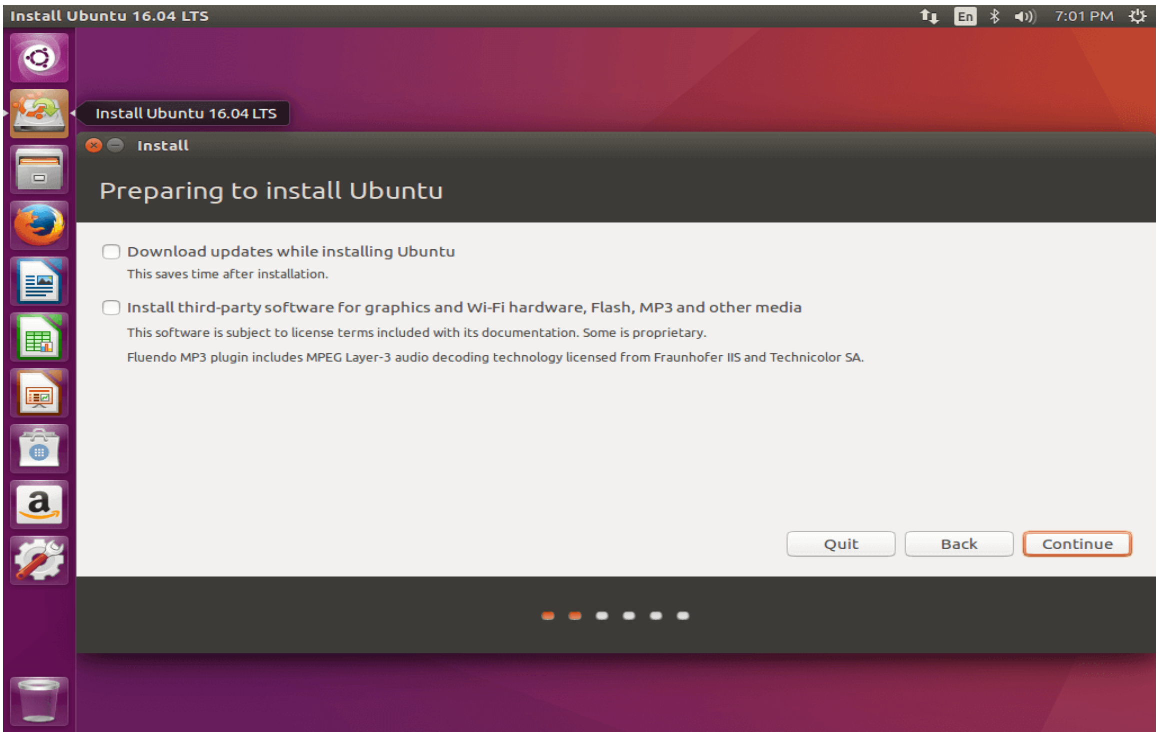


Fig 4.5 Shows language selection in Ubuntu installation

Now it’s time to select an Installation Type. You can choose to Install Ubuntu alongside Windows Boot Manager, option that will automatically take care of all the partition steps.

Use this option if you don’t require personalized partition scheme. In case you want a custom partition layout, check the something else option and hit on Continue button to proceed further. The option Erase disk and install Ubuntu should be avoided on dual-boot because is potentially dangerous and will wipe out your disk.


Install Ubuntu 16.04 LTS 7:01 PM

Install Ubuntu 16.04 LTS

Install

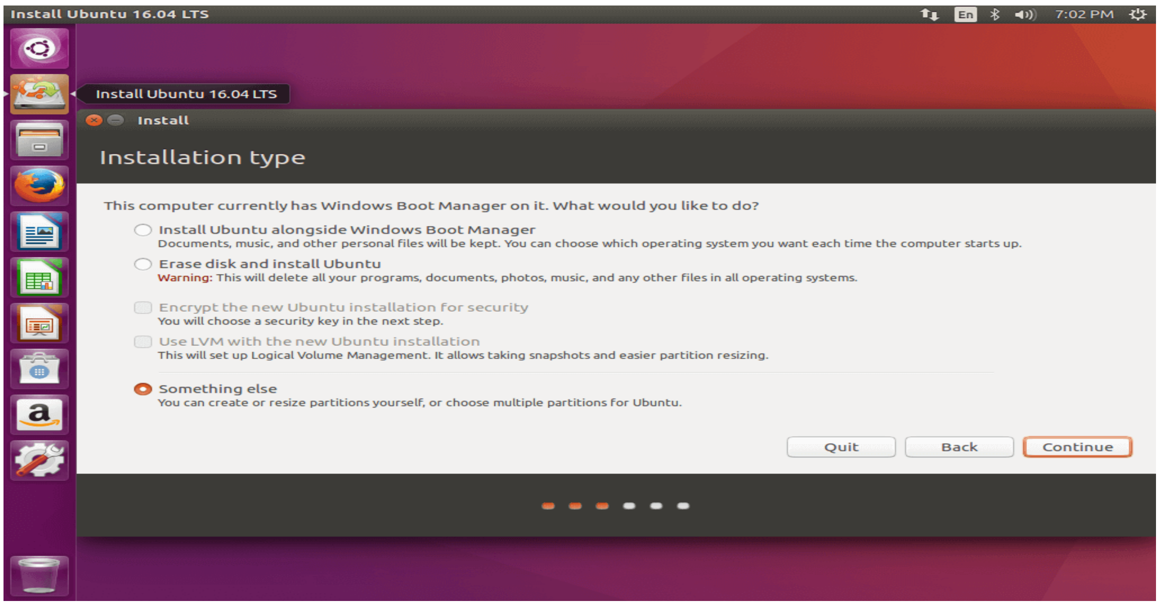
Preparing to install Ubuntu

☐ Download updates while installing Ubuntu
This saves time after installation.

☐ Install third-party software for graphics and Wi-Fi hardware, Flash, MP3 and other media
This software is subject to license terms included with its documentation. Some is proprietary.
Fluendo MP3 plugin includes MPEG Layer-3 audio decoding technology licensed from Fraunhofer IIS and Technicolor SA.

Quit Back Continue

Fig 4.6 Shows permissions for Ubuntu


Install Ubuntu 16.04 LTS 7:02 PM

Install Ubuntu 16.04 LTS

Install

Installation type

This computer currently has Windows Boot Manager on it. What would you like to do?

○ Install Ubuntu alongside Windows Boot Manager
Documents, music, and other personal files will be kept. You can choose which operating system you want each time the computer starts up.

○ Erase disk and install Ubuntu
Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.

☐ Encrypt the new Ubuntu installation for security
You will choose a security key in the next step.

☐ Use LVM with the new Ubuntu installation
This will set up Logical Volume Management. It allows taking snapshots and easier partition resizing.

[✓ Something else]
You can create or resize partitions yourself, or choose multiple partitions for Ubuntu.

Quit Back Continue

Fig 4.7 Shows permission for ubuntu

On this step we'll create our custom partition layout for Ubuntu 16.04. On this guide will recommend that you create two partitions, one for root and the other for home accounts data and no partition for swap (use a swap partition only if you have limited RAM resources or you use a fast SSD).

To create the first partition, the root partition, select the free space (the shrink space from Windows created earlier) and hit on the

+ icon below. On partition settings use the following configurations and hit OK to apply changes:

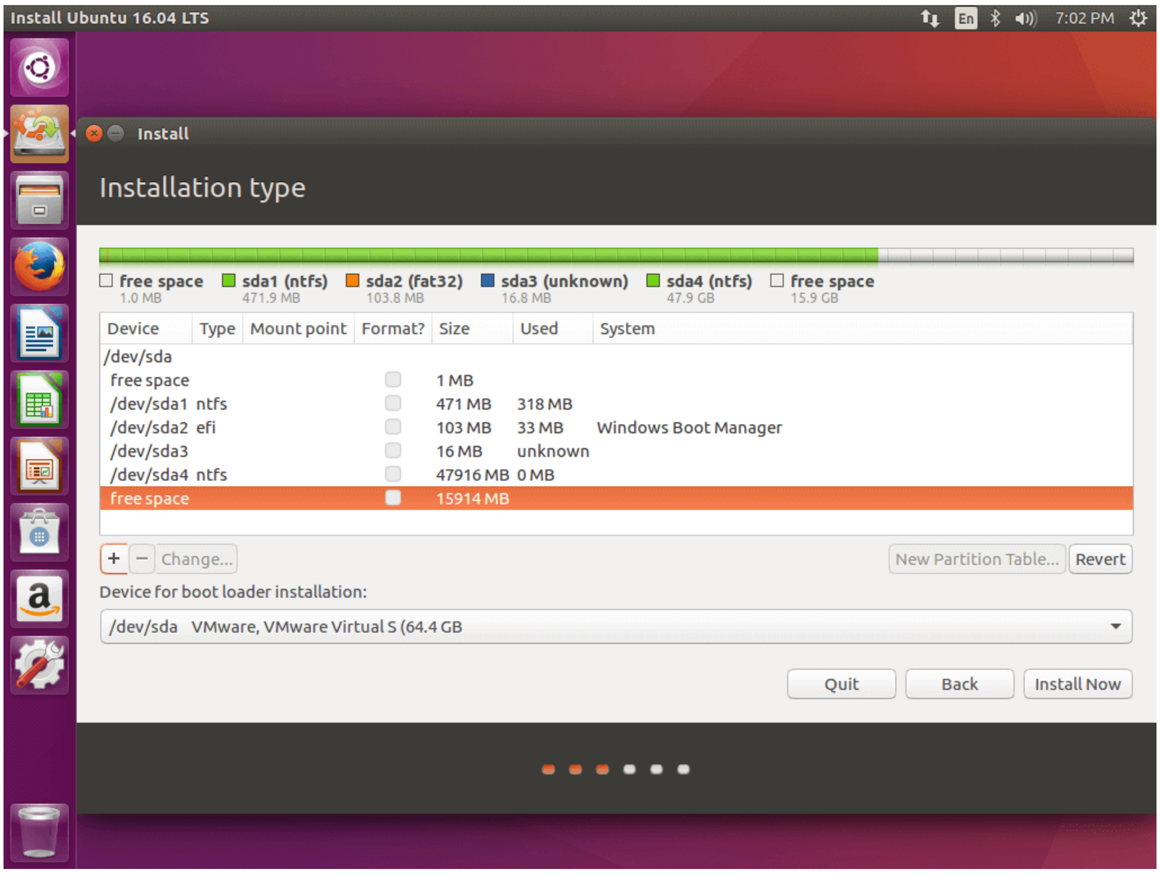
Size = at least 20000 MB

Type for the new partition = Primary

Location for the new partition = Beginning

Use as = EXT4 journaling file system

Mount point = /



Installation type

☐ free space	🟩 sda1 (ntfs)	🟧 sda2 (fat32)	🟦 sda3 (unknown)	🟩 sda4 (ntfs)	☐ free space
1.0 MB	471.9 MB	103.8 MB	16.8 MB	47.9 GB	15.9 GB

Device	Type	Mount point	Format?	Size	Used	System
/dev/sda						
free space			☐	1 MB		
/dev/sda1	ntfs		☐	471 MB	318 MB	
/dev/sda2	efi		☐	103 MB	33 MB	Windows Boot Manager
/dev/sda3			☐	16 MB	unknown	
/dev/sda4	ntfs		☐	47916 MB	0 MB	
free space			☐	15914 MB		

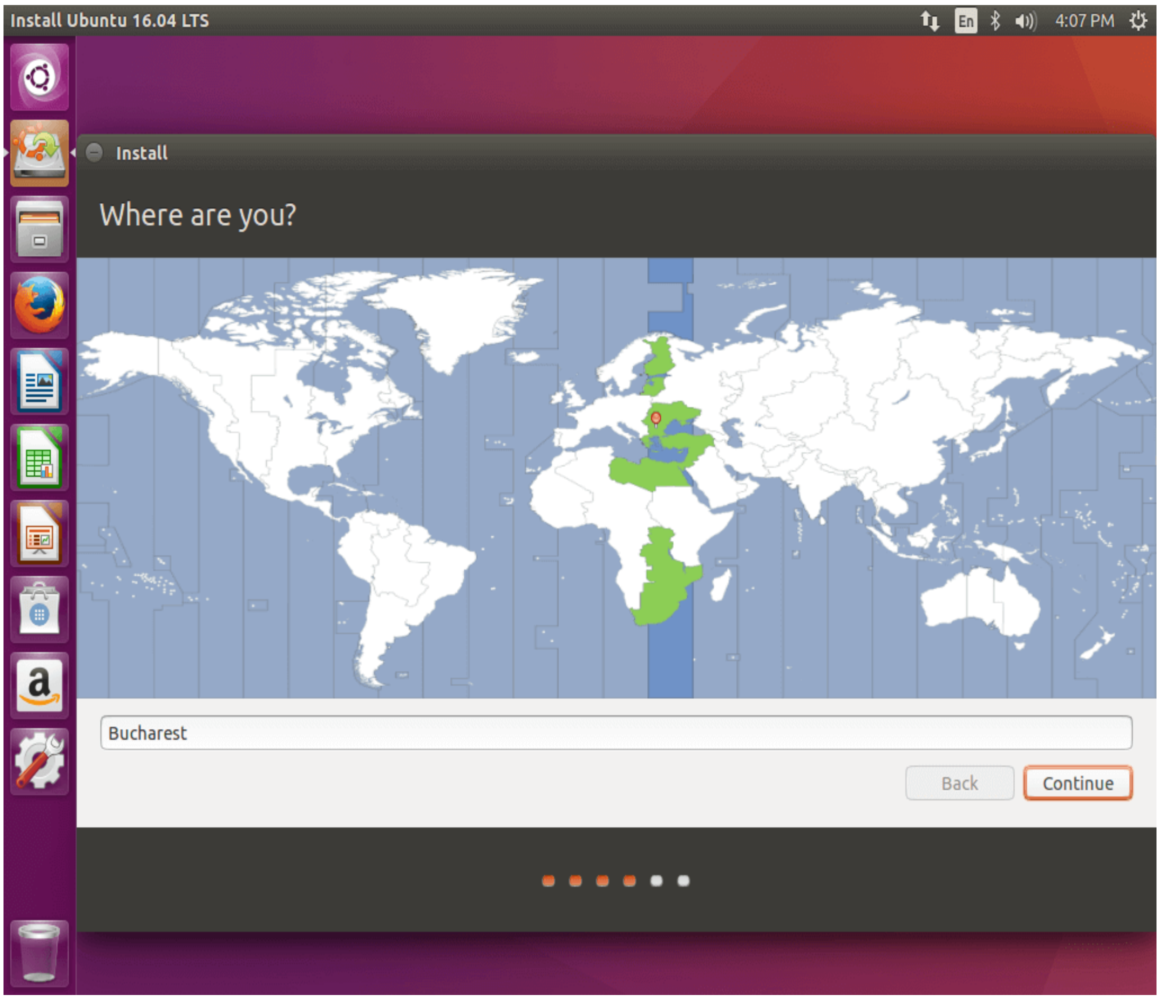
Device for boot loader installation:

/dev/sda VMware, VMware Virtual S (64.4 GB

Fig 4.8 Shows disk partitions

When finished, hit the Install Now button in order to apply changes to disk and start the installation process. A pop-up window should appear to inform you about swap space. Ignore the alert by pressing on Continue button. Next a new pop-up window will ask you if you agree with committing changes to disk. Hit Continue to write changes to disk and the installation process will now start.

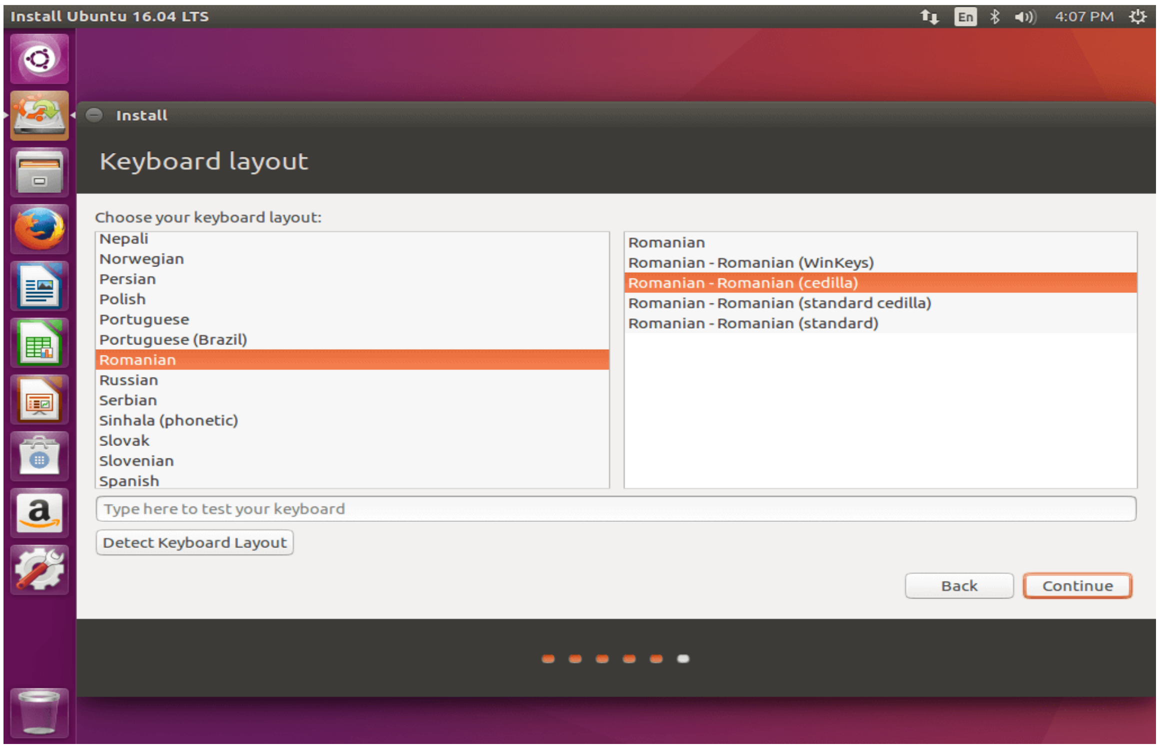
On the next screen adjust your machine physical location by selecting a city nearby from the map. When done hit Continue to move ahead.



Bucharest

Fig 4.9 Time zone selection

Next, select your keyboard layout and click on Continue button.Pick up a username and password for your administrative sudo account, enter a descriptive name for your computer and hit Continue to finalize the installation. This are all the settings required for customizing Ubuntu 16.04 installation. From here on the installation process will run automatically until it reaches the end. After the installation process reaches its end hit on Restart Now button in order to complete the installation.


Install Ubuntu 16.04 LTS

Install

Keyboard layout

Choose your keyboard layout:

Nepali	Romanian
Norwegian	Romanian - Romanian (WinKeys)
Persian	Romanian - Romanian (cedilla)
Polish	Romanian - Romanian (standard cedilla)
Portuguese	Romanian - Romanian (standard)
Portuguese (Brazil)	
Romanian	
Russian	
Serbian	
Sinhala (phonetic)	
Slovak	
Slovenian	
Spanish	

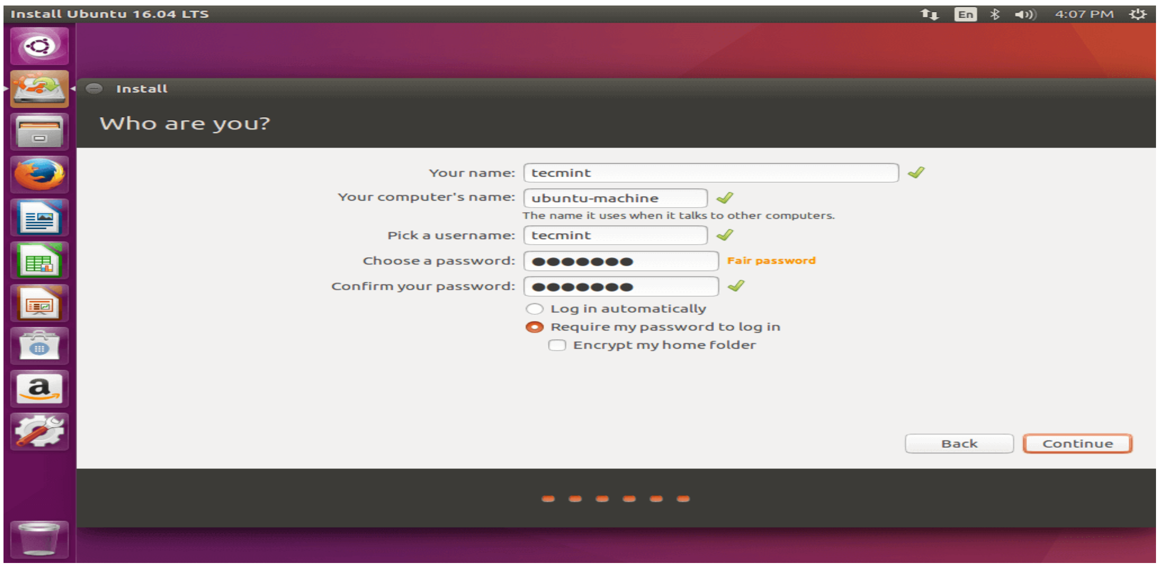
Type here to test your keyboard

Detect Keyboard Layout

Back Continue

Fig 4.10 Shows Keyboard layout selection

This are all the settings required for customizing Ubuntu 16.04 installation. From here on the installation process will run automatically until it reaches the end. After the installation process reaches its end hit on Restart Now button in order to complete the installation.


Install Ubuntu 16.04 LTS

Install

Who are you?

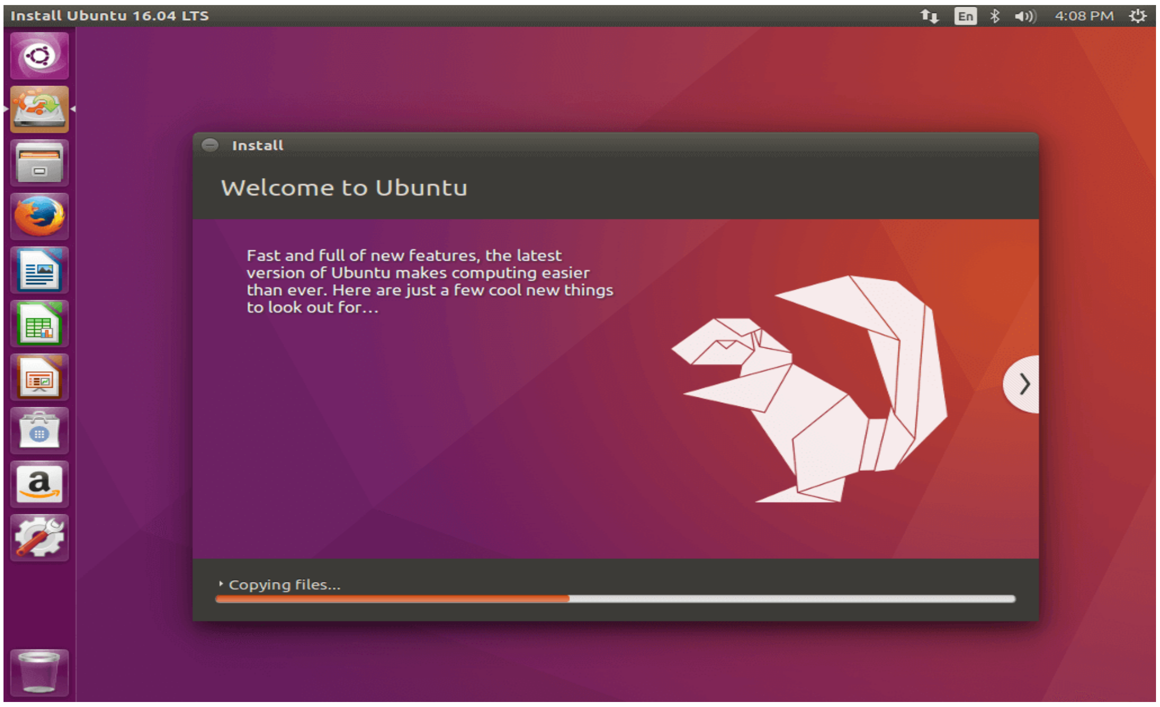
Your name:	tecmint
Your computer's name:	ubuntu-machine
	The name it uses when it talks to other computers.
Pick a username:	tecmint
Choose a password:	●●●●●●● Fair password
Confirm your password:	●●●●●●●

○ Log in automatically
● Require my password to log in
☐ Encrypt my home folder

Back Continue

Fig 4.11 Shows Username and password textboxes

The machine will reboot into the Grub menu, where for ten seconds, you will be presented to choose what OS you wish to use further: Ubuntu 16.04 or Microsoft Windows.

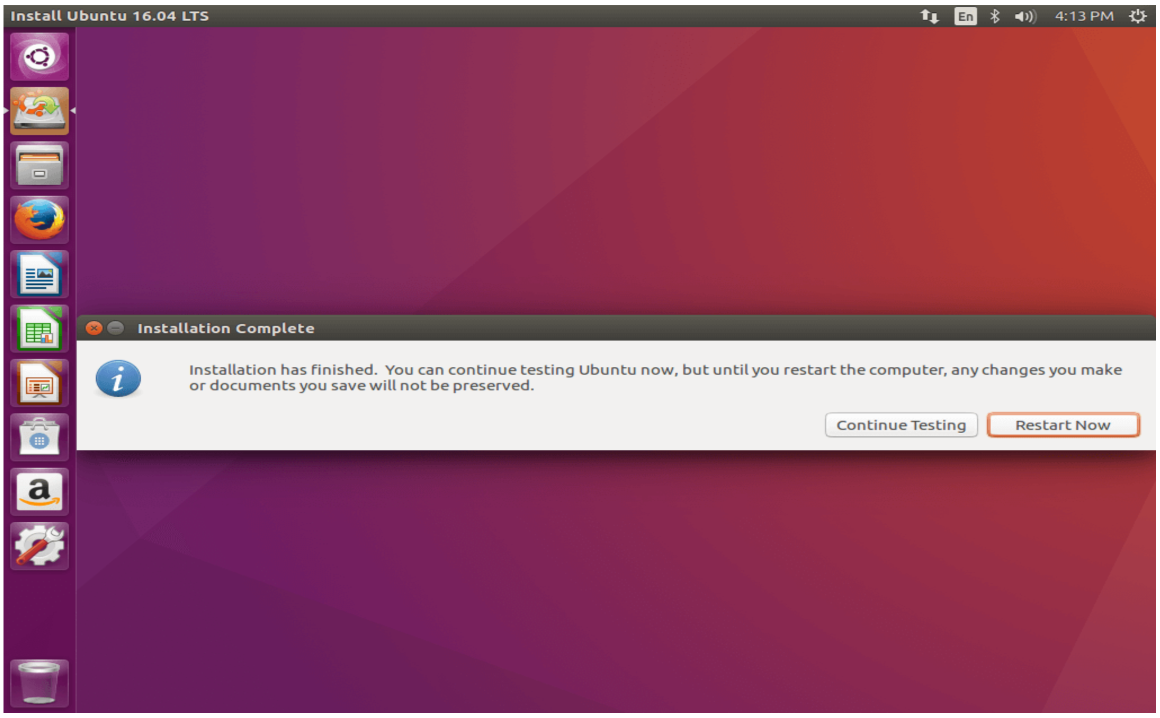

Install Ubuntu 16.04 LTS

Install

Welcome to Ubuntu

Fast and full of new features, the latest version of Ubuntu makes computing easier than ever. Here are just a few cool new things to look out for…

Copying files...

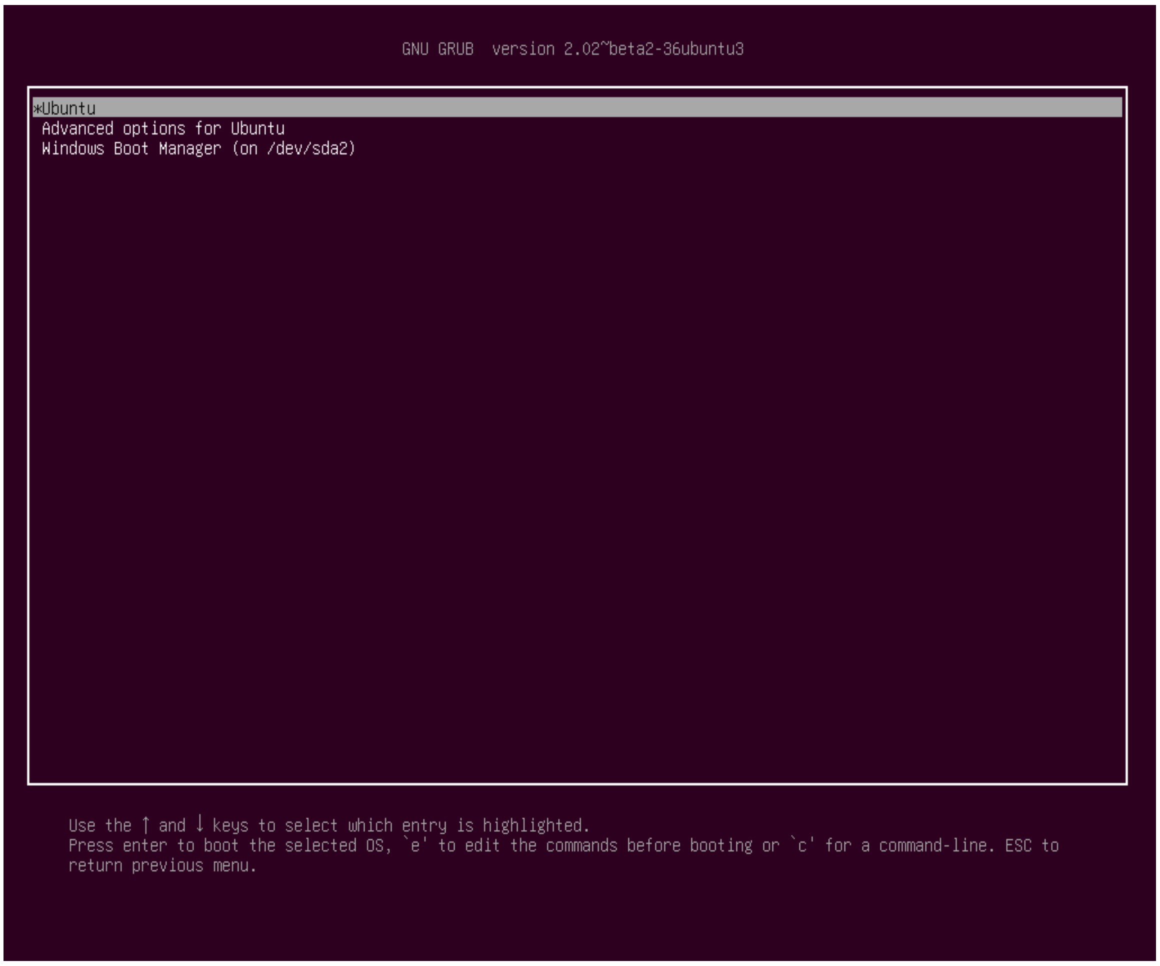

Install Ubuntu 16.04 LTS

Installation Complete

Installation has finished. You can continue testing Ubuntu now, but until you restart the computer, any changes you make or documents you save will not be preserved.

Fig 4.12 & 4.13 Shows Ubuntu installation progress

Ubuntu is designated as default OS to boot from.Thus, just press Enter key or wait for those 10 seconds timeout to drain.



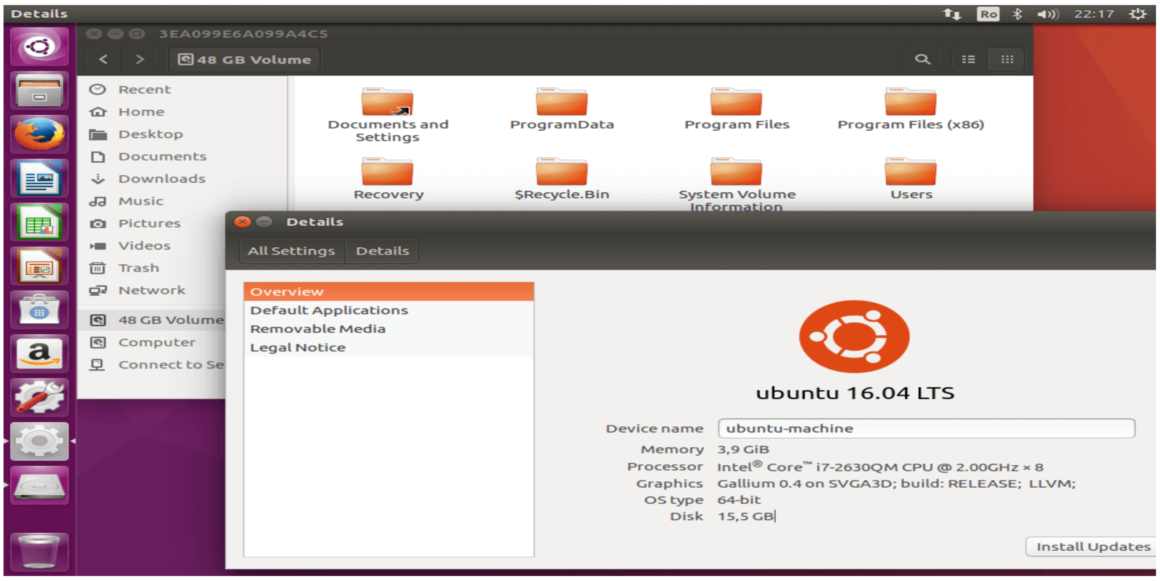
```
GNU GRUB  version 2.02~beta2-36ubuntu3

*Ubuntu
 Advanced options for Ubuntu
 Windows Boot Manager (on /dev/sda2)

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, `e' to edit the commands before booting or `c' for a command-line. ESC to
return previous menu.
```

Fig 4.14 Shows Ubuntu boot screen

After Ubuntu finishes loading, login with the credentials created during the installation process and enjoy it. Ubuntu 16.04 provides NTFS file system support automatically so you can access the files from Windows partitions just by clicking on the Windows volume. That’s it! In case you need to switch back to Windows, just reboot the computer and select Windows from the Grub menu.



ubuntu 16.04 LTS

Device name ubuntu-machine
Memory 3,9 GiB
Processor Intel® Core™ i7-2630QM CPU @ 2.00GHz × 8
Graphics Gallium 0.4 on SVGA3D; build: RELEASE; LLVM;
OS type 64-bit
Disk 15,5 GB

Fig 4.15 Shows Ubuntu Desktop

4.6 Network Simulator-2.34

Ns-2 began as a revision of ns-1. From 1997-2000, ns development was supported by DARPA through the VINT project at LBL, Xerox PARC, UCB, and USC/ISI. In 2000, ns-2 development was supported through DARPA with SAMAN and through NSF with CONSER, both at USC/ISI, in collaboration with other researchers including ACIRI. Ns-2 incorporates substantial contributions from third parties, including wireless code from the UCB Daedelus and CMU Monarch projects and Sun Microsystems.

Ns is an object oriented simulator, written in C++, with an OTcl interpreter as a frontend. The simulator supports a class hierarchy in C++ (also called the compiled hierarchy in this document), and a similar class hierarchy within the OTcl interpreter (also called the interpreted hierarchy in this document). The two hierarchies are closely related to each other; from the user's perspective, there is a one-to-one correspondence between a class in the interpreted hierarchy and one in the compiled hierarchy. The root of this hierarchy is the class Tcl Object. Users create new simulator objects through the interpreter; these objects are instantiated within the interpreter, and are closely mirrored by a corresponding object in the compiled hierarchy. The interpreted class hierarchy is automatically established through methods defined in the class Tcl Class. user instantiated objects are mirrored through methods defined in the class Tcl Object. There are other hierarchies in the C++ code and OTcl scripts; these other hierarchies are not mirrored in the manner of Tcl Object.

Why two languages? ns uses two languages because simulator has two different kinds of things it needs to do. On one hand, detailed simulations of protocols requires a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks run-time speed is important and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important. On the other hand, a large part of network research involves slightly varying parameters or configurations, or quickly exploring a number of scenarios. In these cases, iteration time (change the model and re-run) is more important. Since configuration runs once (at the beginning of the simulation), run-time of this part of the task is less important. ns meets both of these needs with two languages, C++ and OTcl. C++ is fast to run but slower to change, making it suitable for detailed protocol implementation. OTcl runs much slower but can be changed very quickly (and interactively), making it ideal for simulation configuration. ns (via tclcl) provides glue to make objects and variables appear on both languages.

4.7 Simulation workflow

The general process of creating a simulation can be divided into several steps

4.8 Topology definition

To ease the creation of basic facilities and define their interrelationships, ns-3 has a system of containers and helpers that facilitates this process.

4.9 Model development

Models are added to simulation (for example, UDP, IPv4, point-to-point devices and links, applications); most of the time this is done using helpers.

4.10 Node and link configuration

Models set their default values (for example, the size of packets sent by an application or MTU of a point-to-point link); most of the time this is done using the attribute system.

4.11 Execution

Simulation facilities generate events, data requested by the user is logged.

4.12 Performance analysis

After the simulation is finished and data is available as a time-stamped event trace. This data can then be statistically analysed with tools like R to draw conclusions.

4.13 Graphical Visualization

Raw or processed data collected in a simulation can be graphed using tools like Gnuplot, matplotlib or XGRAPH.

4.14 Installation of Ns-2.34 in Ubuntu 16.04

Step 1:

Download ns-allinone-2.34.tar.gz

Download link:

https://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.34/

Step 2:

Copy it to home folder(not compulsory, you can change as per your choice) Open Terminal (Altr+ctrl+t)

Step 3:

Install the basic packages required to install the ns2

```
$sudo apt-get update
```

$sudo apt-get install build-essential autoconf automake libxmudev

Step 4:

Extract tar file to home. In terminal default directory is home. Run follwing command to extract tar file to home directory $tar -zxvf ns-allinone-2.34.tar.gz if there is any error while extracting above file you can manually go to file location and right clicking file location you will see option for extract here.

Step 5:

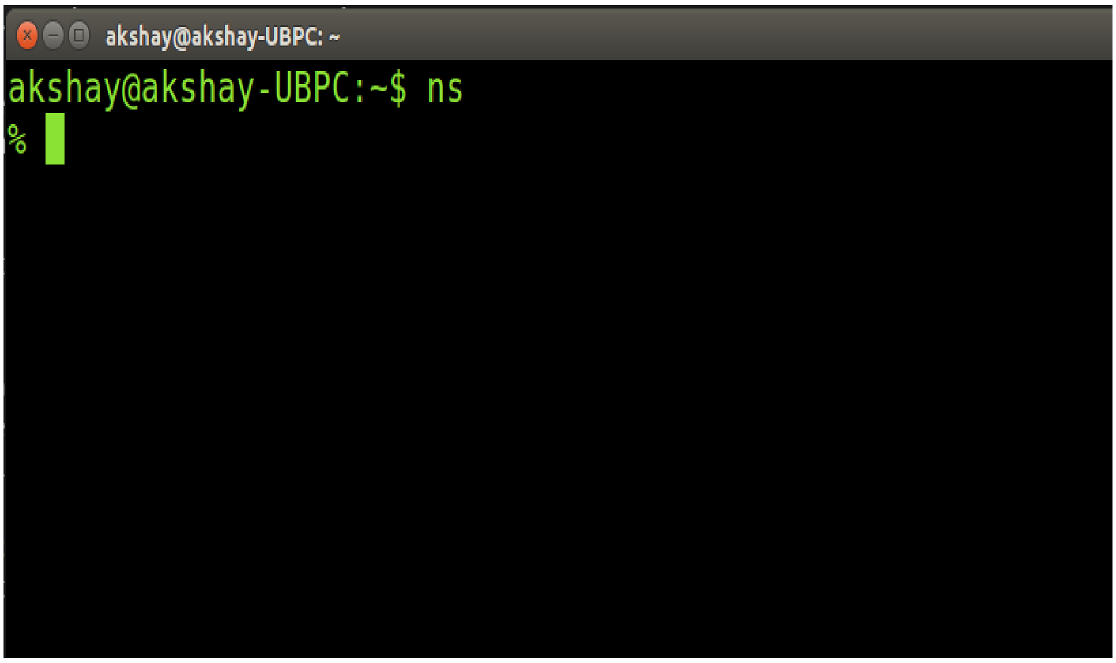
Go to ns-allinone-2.34 directory and run
./install
$cd ns-allinone-2.34
$./install

Step 6:

Go to ns-2.34 directory
$cd ns-2.34
$./configure $sudo make
$sudo make install

Step 7:

Go back and go to nam-1.14 directory
$cd ..
$cd nam-1.14
$./configure $sudo make $sudo make install
That's it. To check you can open new terminal(Ctrl+Alt+t) and enter following command $ns If it shows % sign then you have installed ns2 correctly to exit from ns2 and enter following command.



```
akshay@akshay-UBPC:~$ ns
%
```

Fig 4.16 Percentage symbol in terminal shows installation of ns.

4.15 Eval Vid-2.7 in NS-2.34

Conventionally, the Internet has been used for data applications like email, chat, Web surfing, etc. Multimedia technology has been around for quite some time, but with the pace of development that it has enjoyed in the last few years, it has become an integral part of almost every Web endeavor. It finds application in fields ranging from video lectures to catalogues and games to video clips. The combination of multimedia technology and the Internet is currently attracting a considerable amount of attention from researchers, developers and end users. Video transmission nowadays has become a necessity for almost every Internet user, irrespective of the device used to access the Internet. Typically, video transmission demands high bandwidth. Without compression, it is very difficult to transmit video over wired or wireless networks. With the increasing use of multimedia-capturing devices and the improved quality of capture, the demand for high quality multimedia has gone up. The traditional protocols used to carry such volumes of multimedia data have started showing some limitations. Hence, enhanced protocols are the need of the hour. Researchers working on developing such protocols need platforms to test the performance of these newly designed protocols. The performance of the suite of protocols should ideally be measured on a real network. But due to several limitations of real networks, simulation platforms are commonly used to test prototypes. This article is aimed at demonstrating the widely used and open source simulation platform, ns-2, for measuring the performance of protocols when multimedia content is transmitted over the network.

4.16 Architecture of ns-2 and EvalVid

Among researchers, network simulator 2 (ns-2) is one of the most widely used, maintained and trusted simulators. It is useful for simulating a wide variety of networks, ranging from Ethernet and wireless LANs, to ad hoc networks and wide area networks. Conventionally, ns-2 supports constant bit rate and traffic in bursts, which resembles most of the data applications on the Internet. Multimedia traffic has characteristics that are different from data traffic. Data traffic is mostly immune to jitter and less sensitive to delay, whereas multimedia applications demand a high quality of service. Generating traffic representing multimedia data is very important when evaluating the performance of any network. EvalVid is a framework and tool-set for evaluating the quality of video transmitted over a physical or simulated communication network. In addition to measuring the QoS (Quality of Service) parameters of the underlying network, like delays, loss rates and jitter, it also provides support for subjective video quality evaluation of the received video, based on frame-by-frame PSNR (Peak Signal to Noise Ratio) calculation. It can regenerate received video, which can be compared with the originally transmitted video in terms of many other useful metrics like video quality measure (VQM), structural similarity (SSIM), mean opinion score (MOS), etc.

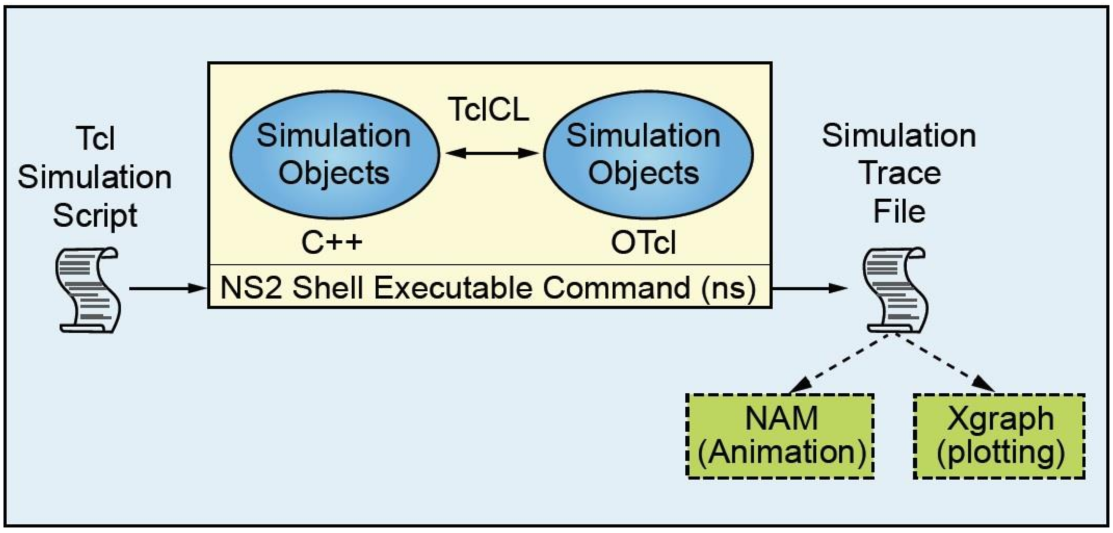


Fig 4.17 Basic architecture of ns-2

EvalVid is a framework and tool-set for evaluating the quality of video transmitted over a physical or simulated communication network. In addition to measuring the QoS (Quality of Service) parameters of the underlying network, like delays, loss rates and jitter, it also provides support for subjective video quality evaluation of the received video, based on frame-by-frame PSNR (Peak Signal to Noise Ratio) calculation. It can regenerate received video, which can be compared with the originally transmitted video in terms of many other useful metrics like video quality measure (VQM), structural similarity (SSIM), mean opinion score (MOS), etc.

CHAPTER-5

DESIGN IMPLEMENTATION

In order to integrate video in ns2 the following steps are to be followed. Most of tools like psnr, MP4, fix yuv etc. work without installing any extra software's like codec. But FFmpeg, etmp4, MP4Box tools require special codec software and library files. FFmpeg is a command line tool where FF means "fast forward" and MPEG is a video standard. It is mainly used to convert one multimedia format to another format.

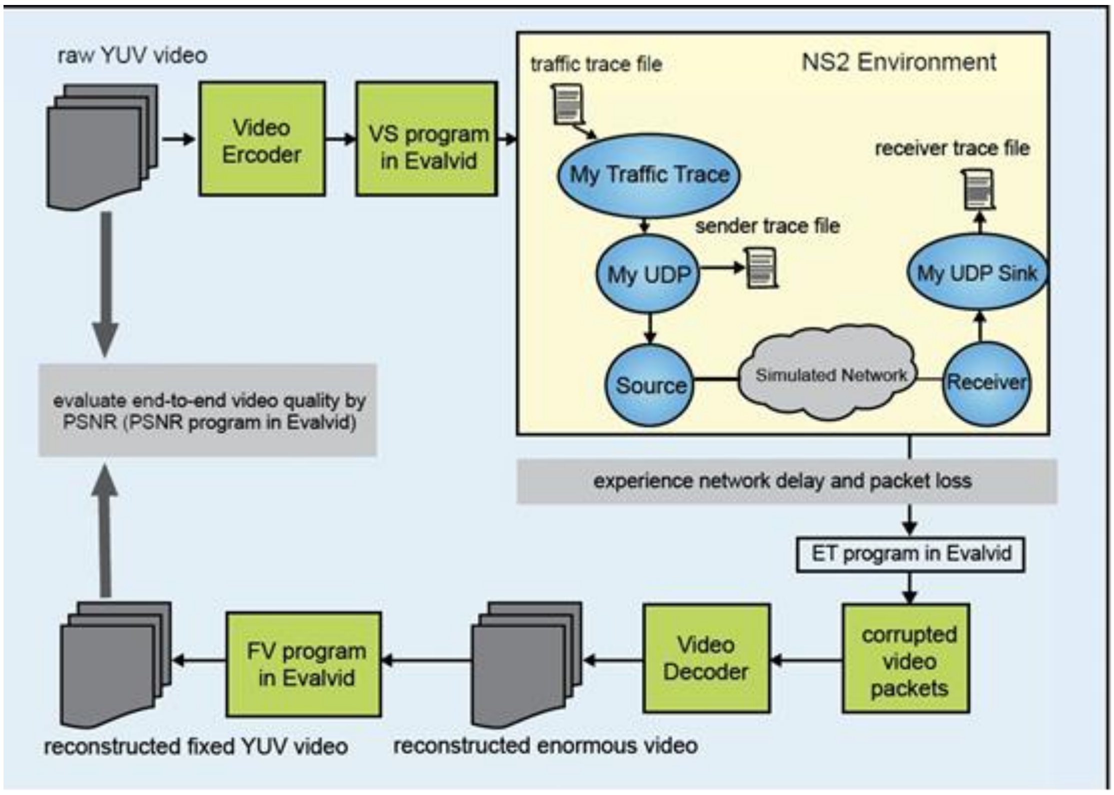


Fig 5.1 Integrated architecture of ns-2 and EvalVid

Then download YUV CIF reference video for akiyo_cif.264, H.264 is a lossless encoded video. The YUV model define color space in terms of brightness (luma) and color (chrominance). Previous black-and-white systems used only luma (Y) information and now color information (U and V) was added, so that black and white receiver would still be able to display a color picture as a normal black and white picture. Most commonly used YUV sampling ratios are 4:4:4 indicates no down sampling.

5.1 Step by step procedure

1. Convert akiyo_cif.264 to akiyo_cif.yuv:

$ffmpeg -i akiyo_cif.264 akiyo_cif.yuv

2. Encode the raw yuv file to m4v:

$ffmpeg -s cif -r 30 -i akiyo_cif.yuv -vcodec mpeg4 -b 64000 -bt 3200 -g 30 akiyo_cif.m4v

3. Convert m4v file to mp4 by using MP4Box:

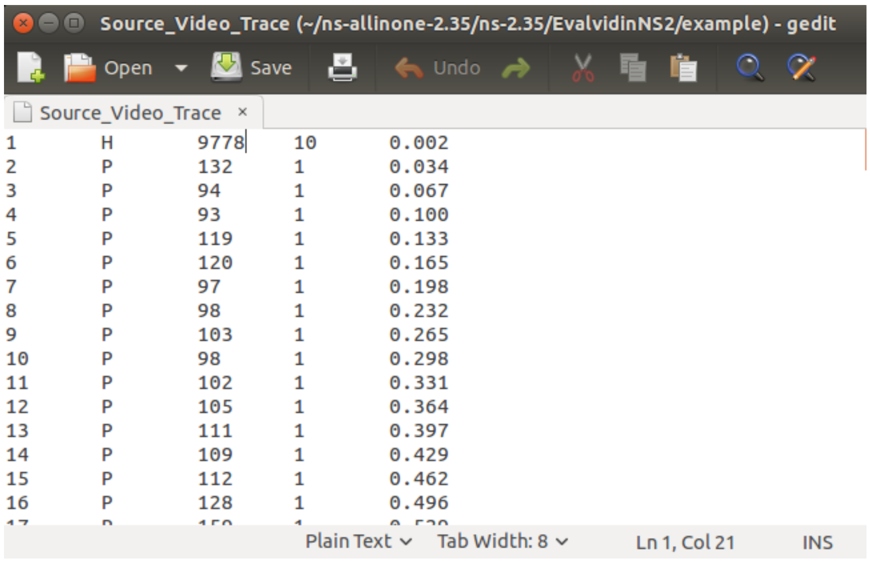
$MP4Box -hint -mtu 1024 -fps 30 -add akiyo_cif.m4v akiyo_cif2.mp4

MP4Box is a multimedia package available in GPAC. It is mostly used for manipulating ISO media files. (e.g. MP4, 3GP) and encoding and decoding multimedia files (e.g. MPEG, AVI).

4. The MP4trace is able to send a hinted mp4-file per RTP/UDP/IP to a specified destination. The output of mp4trace will be needed later, so it is redirected to a file.

$./mp4trace -f -s 224.1.2.3 12346 akiyo_cif2.mp4 >Source_Video_Trace

5. Then it creates a Source_Video_Trace with the traces shown in figure. And shows output on screen as Track 1: Video (MPEG-4) - 352x288 pixel, 300 samples, 00:00:10.000 Track 2: Hint (RTP) for track 1 - 300 samples, 00:00:10.000 NS2 script for wireless simulation scenario available in. After simulation we will get sd_be for sender trace file and rd_be for the receiver trace file.


Source_Video_Trace (~/ns-allinone-2.35/ns-2.35/EvalvidinNS2/example) - gedit
Open
Save
Undo
Source_Video_Trace ×
Plain Text
Tab Width: 8
Ln 1, Col 21
INS

```
1          H          9778      10         0.002
2          P          132       1          0.034
3          P          94        1          0.067
4          P          93        1          0.100
5          P          119       1          0.133
6          P          120       1          0.165
7          P          97        1          0.198
8          P          98        1          0.232
9          P          103       1          0.265
10         P          98        1          0.298
11         P          102       1          0.331
12         P          105       1          0.364
13         P          111       1          0.397
14         P          109       1          0.429
15         P          112       1          0.462
16         P          128       1          0.496
```

Fig 5.2 Snap of source video trace

6. Create a network topology representing the network to be simulated.
7. After simulation, ns-2 will create two files, Send_time_file and Recv_time_file (the filename as used in the tcl file), which record the sending time and received time of each packet, respectively.

5.2 Generation of Traffic and Topology file

Use the following command in the terminal of Linux to generate traffic files

./setdest -v <2> -n <nodes> -s <speed type> -m <min speed> -M <max speed>-t <simulation time> -P <pause type> -p <pause time> -x <max X> -y <max Y>)

Where n specifies number of nodes, s specifies speed of mobility nodes, m specifies minimum speed of nodes, M specifies maximum speed of nodes, t specifies total simulation time, P specifies pause time for nodes, x and y denotes the maximum value of x and y axis of the topology.Use the command

./ns cbrgen.tcl -type cbr -nn 10 -seed 2 -mc 5 -rate 4.0 >traffic2 to generate different traffic files.

5.3 Program Execution

Integrate the generated traffics and topologies in the programs by specifying the files names in the programs maodvbasic.tcl.

Using the source video trace, sender trace and receiver trace generated after execution of the program, calculate jitter rate, send time, receive time, receive time, number of senders, number of receivers, throughput, routing load, latency using the command

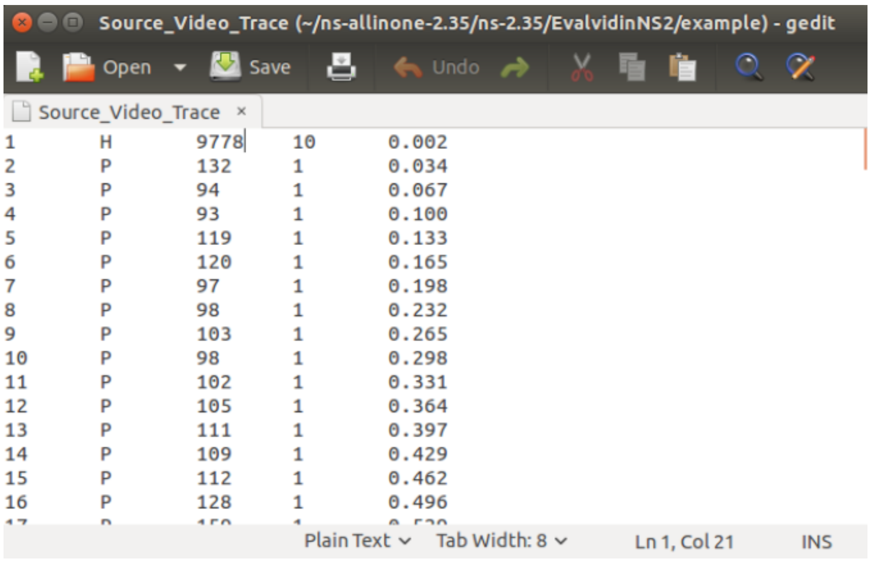
```
awk -f QoSeval.awk output.tr
```

A text file with results is generated in ns2.34 folder which contains values of jitter rate, send time, receive time, receive time, number of senders, number of receivers, throughput, routing load, latency.

CHAPTER-6

RESULTS

The source video file created is a text format file which indicates the brightness and luminance values as shown



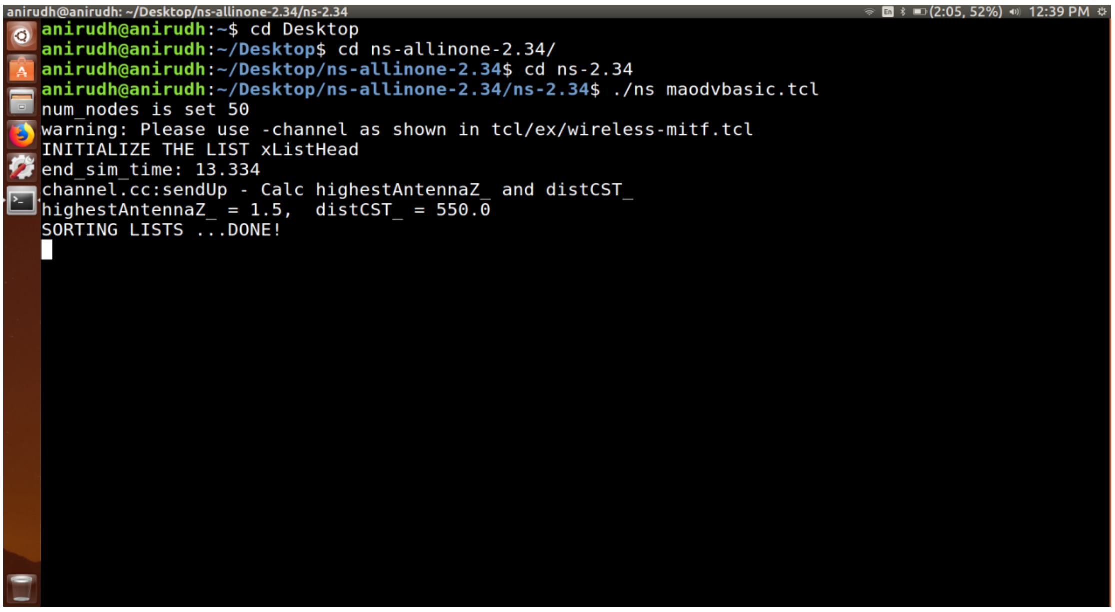
Source_Video_Trace (~/ns-allinone-2.35/ns-2.35/EvalvidinNS2/example) - gedit

```
1       H       9778    10      0.002
2       P       132     1       0.034
3       P       94      1       0.067
4       P       93      1       0.100
5       P       119     1       0.133
6       P       120     1       0.165
7       P       97      1       0.198
8       P       98      1       0.232
9       P       103     1       0.265
10      P       98      1       0.298
11      P       102     1       0.331
12      P       105     1       0.364
13      P       111     1       0.397
14      P       109     1       0.429
15      P       112     1       0.462
16      P       128     1       0.496
```

Plain Text Tab Width: 8 Ln 1, Col 21 INS

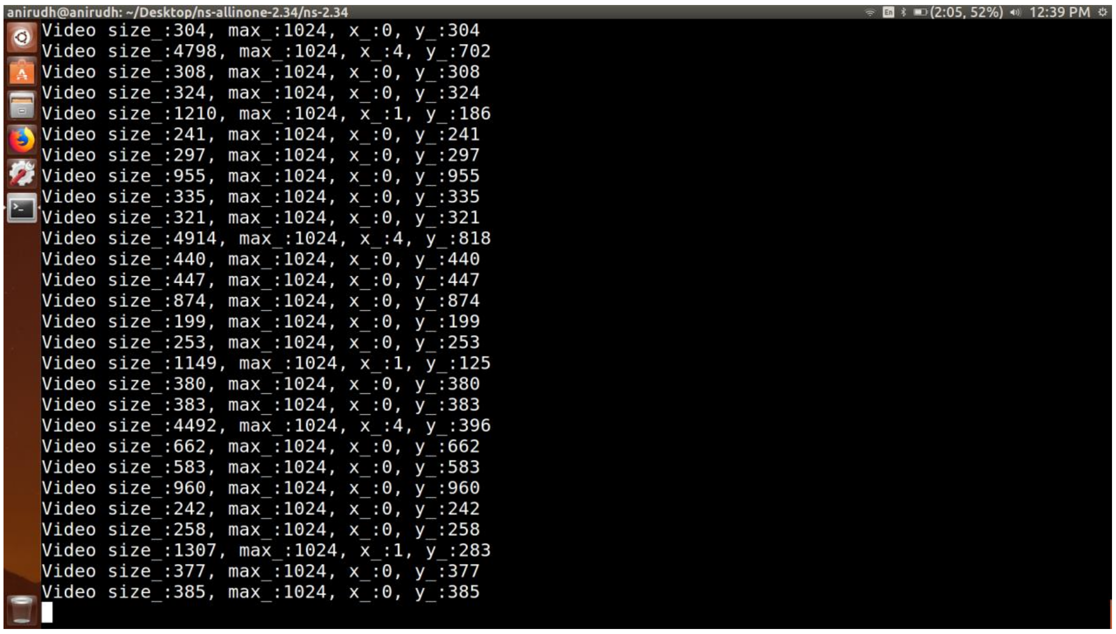
Fig 6.1 Screenshot of video trace file

When the execution of the program is started the following processing is done in the terminal window.



```
anirudh@anirudh:~$ cd Desktop
anirudh@anirudh:~/Desktop$ cd ns-allinone-2.34/
anirudh@anirudh:~/Desktop/ns-allinone-2.34$ cd ns-2.34
anirudh@anirudh:~/Desktop/ns-allinone-2.34/ns-2.34$ ./ns maodvbasic.tcl
num_nodes is set 50
warning: Please use -channel as shown in tcl/ex/wireless-mitf.tcl
INITIALIZE THE LIST xListHead
end_sim_time: 13.334
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
```

Fig 6.2 Figure shows the terminal window running ns-2.34



```
anirudh@anirudh: ~/Desktop/ns-allinone-2.34/ns-2.34                    (2:05, 52%)  12:39 PM
Video size :304, max :1024, x :0, y :304
Video size :4798, max :1024, x :4, y :702
Video size :308, max :1024, x :0, y :308
Video size :324, max :1024, x :0, y :324
Video size :1210, max :1024, x :1, y :186
Video size :241, max :1024, x :0, y :241
Video size :297, max :1024, x :0, y :297
Video size :955, max :1024, x :0, y :955
Video size :335, max :1024, x :0, y :335
Video size :321, max :1024, x :0, y :321
Video size :4914, max :1024, x :4, y :818
Video size :440, max :1024, x :0, y :440
Video size :447, max :1024, x :0, y :447
Video size :874, max :1024, x :0, y :874
Video size :199, max :1024, x :0, y :199
Video size :253, max :1024, x :0, y :253
Video size :1149, max :1024, x :1, y :125
Video size :380, max :1024, x :0, y :380
Video size :383, max :1024, x :0, y :383
Video size :4492, max :1024, x :4, y :396
Video size :662, max :1024, x :0, y :662
Video size :583, max :1024, x :0, y :583
Video size :960, max :1024, x :0, y :960
Video size :242, max :1024, x :0, y :242
Video size :258, max :1024, x :0, y :258
Video size :1307, max :1024, x :1, y :283
Video size :377, max :1024, x :0, y :377
Video size :385, max :1024, x :0, y :385
```

Fig 6.3 Figure shows the terminal window running ns-2.34

Network simulator takes the input video specified and transmits over a MANET traffic using MAODV protocol. After simulation is complete the NAM window opens up showing how video is transmitted over MANET graphically.

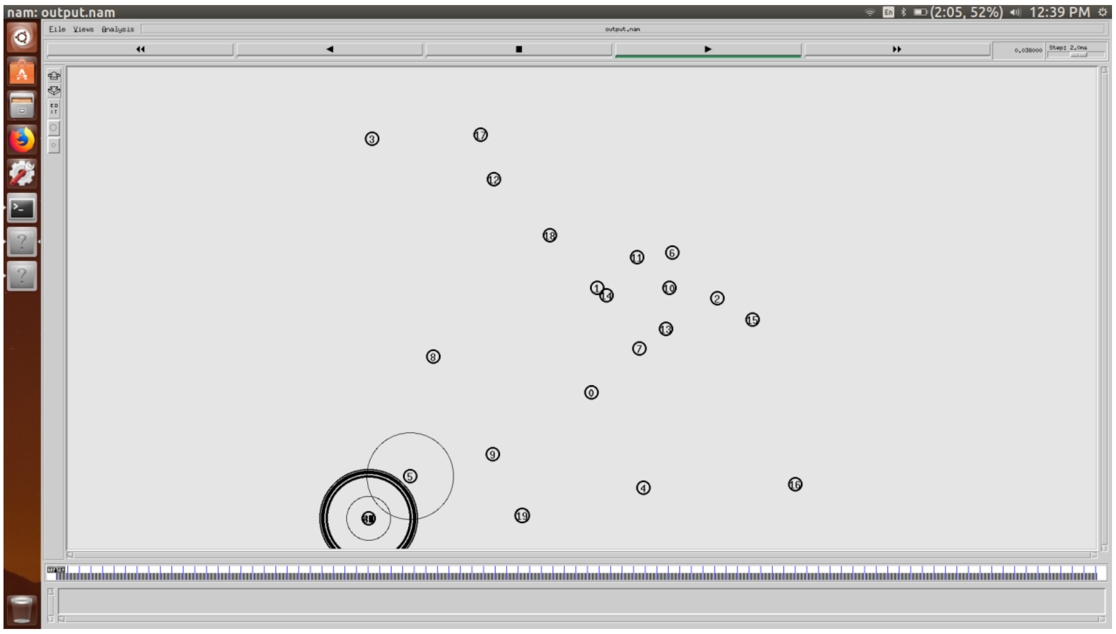


Fig 6.4 Figure shows the NAM window

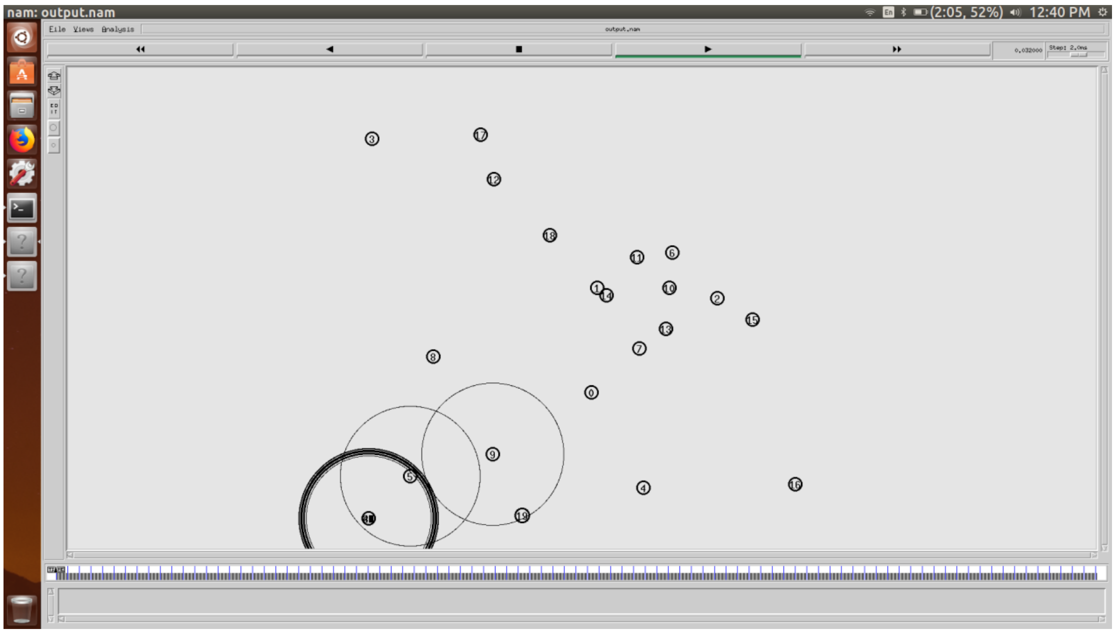
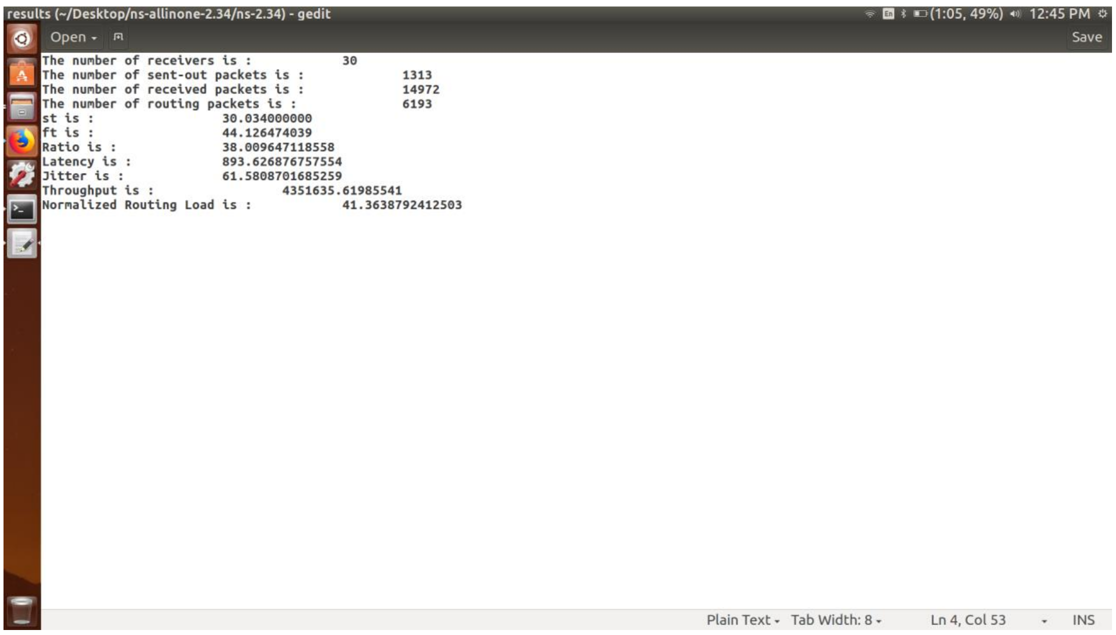


Fig 6.5 Figure shows data transmission pattern in the NAM window

The jitter, send time, receive time, receive time, number of senders, number of receivers, throughput, routing load, latency is calculated by running awk -f QoSeval.awk output.tr command in terminal window.

This saves a text file names results in which the following parameters will be displayed.



```
The number of receivers is :                30
The number of sent-out packets is :                 1313
The number of received packets is :                 14972
The number of routing packets is :                  6193
st is :                 30.034000000
ft is :                 44.126474039
Ratio is :              38.009647118558
Latency is :            893.626876757554
Jitter is :             61.5808701685259
Throughput is :                 4351635.61985541
Normalized Routing Load is :                41.3638792412503
```

Fig 6.6 The above figure shows parameters calculated.

The following protocol was tested for different number of senders and the QoS parameters like Jitter rate, routing load have been measured.

6.1 QoS Parameters

- **Jitter rate**

Jitter is a variation in packet transit delay caused by queuing, contention and serialization effects on the path through the network. In general, higher levels of jitter are more likely to occur on either slow or heavily congested links.

- **Throughput**

Network throughput is usually represented as an average and measured in bits per second (bps), or in some cases as data packets per second. Throughput is an important indicator of the performance and quality of a network connection. A high ratio of unsuccessful message delivery will ultimately lead to lower throughput and degraded performance.

- **Normalized Routing Load**

Normalized Routing Load (or Normalized Routing Overhead) is defined as the total number of routing packet transmitted per data packet. It is calculated by dividing the total number of routing packets sent (includes forwarded routing packets as well) by the total number of data packets received.

6.2 Graphs

All the below graphs QoS parameters on Y-axis and number os senders on X-axis.

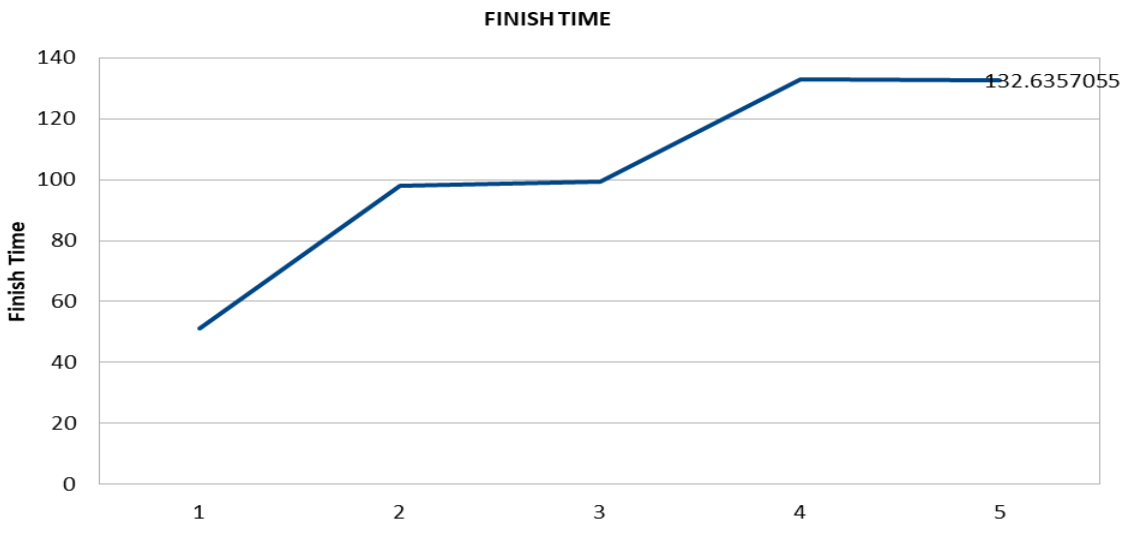


Fig 6.7 The about graph shows the finish time for different number of senders with X-axis showing number of senders and Y-axis showing finish time

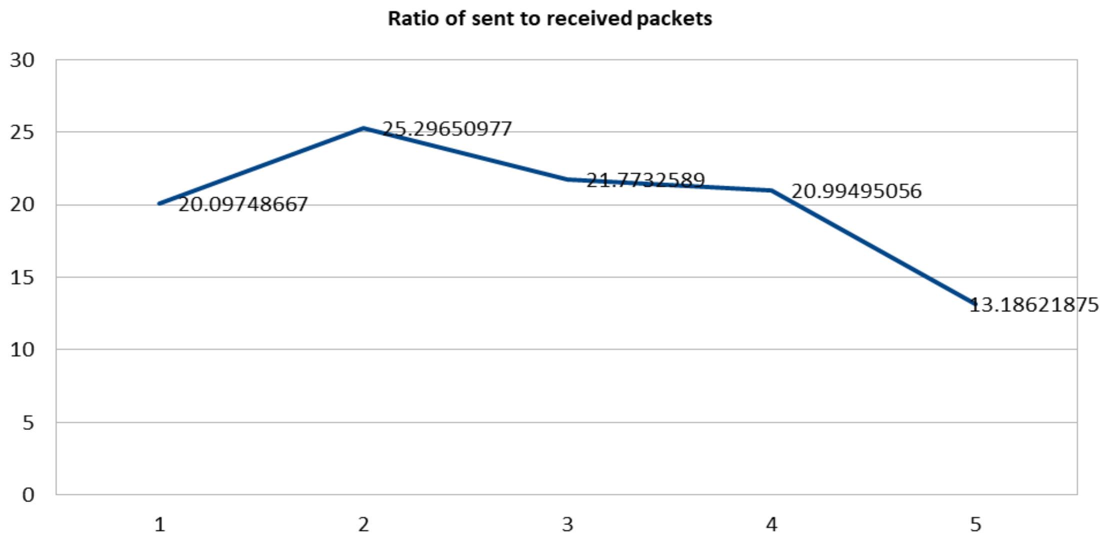


Fig 6.8 The above graph shows Ratio of sent to received packets for different number of receivers

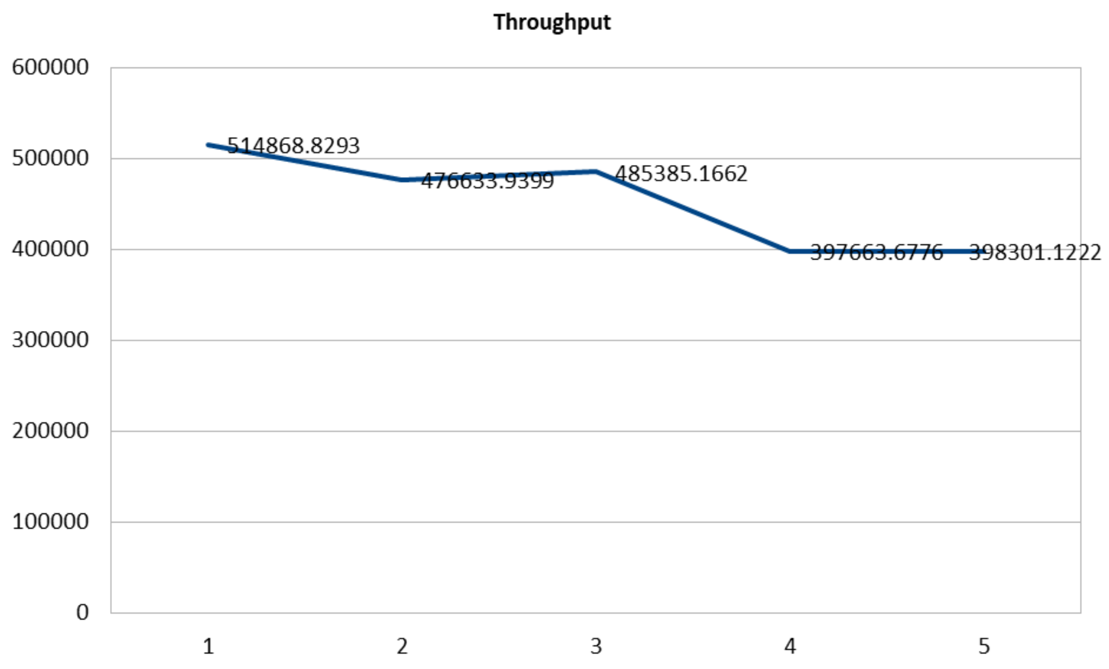


Fig 6.9 The above graph shows Throughput for different number of senders

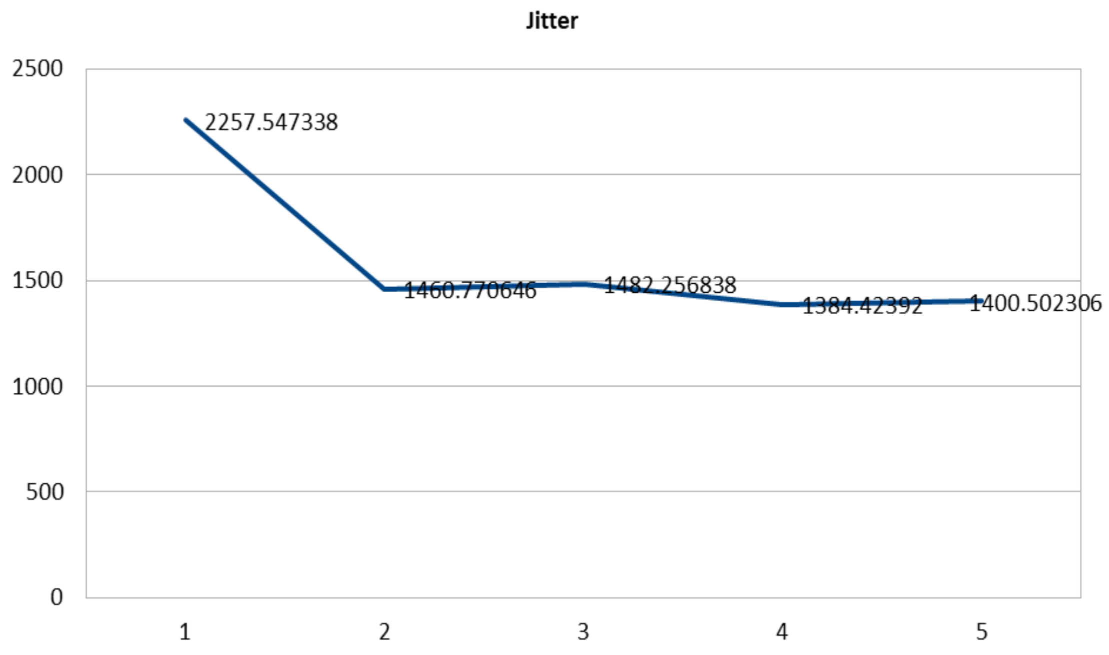


Fig 6.10 The above graph shows Jitter rate for different number of senders

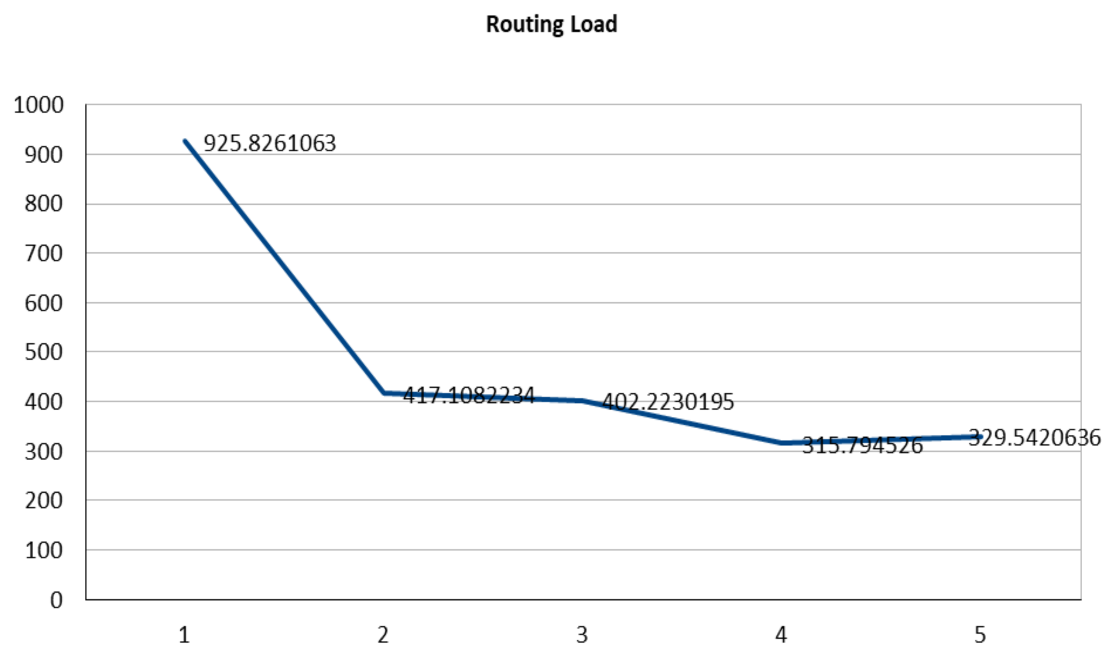


Fig 6.8 The above graph shows Routing load for different number of senders

All the above graphs were simulations results for different traffic scenarios. A total of 10 scenarios were generated and were tested with different number of senders. In all the above graphs, X-axis shows numbers of senders in which maximum number of senders is 5 and minimum is 1.

CONCLUSION AND FUTURE SCOPE

Validation tests were performed to ensure correct functionality of our algorithm. We have also performed cross-platform performance tests and presented a comparison analysis. As the number of senders are increasing, the QoS parameters were decreasing. Due to mobility of nodes, none of the receivers were able to receive complete video in a multicast group. The results calculated were fair enough when number of senders was limited to one. This Video transmission over manets can be used for military purpose and during times of natural disasters where network infrastructure gets damaged completely.

The values of QoS parameters calculated were less because of traffic congestion and high speed transmission of frames. A reduction in number of frames transmitted per second may lead to improved performance.

REFERENCES

[1] B. Xu, S. Hischke, and B Walke, “The Role of Ad Hoc Networking in Future Wireless Communications,” Proc. IEEE ICCT'2003, vol. 2, pp. 1353 – 1358, Beijing, China, Apr. 2003.

[2] E. M. Royer, and C. E. Perkins, “Multicast Ad Hoc On-Demand Distance Vector (MAODV) Routing,” IETF draft-ietf-manet-maodv-00.txt, Jul. 2000.

[3] F. H. P. Fitzek, B. Can, R. Prasad, and M. Katz, “Overhead and Quality Measurements for Multiple Description Coding for Video Services,” Proc. WPMC, pp. 524 – 428, Italy, Sep. 2004.

[4] J. Yoon, M. Liu, and B. Noble, “Random Waypoint Considered Harmful,” Proc. IEEE. INFOCOM 2003, San Francisco, USA, vol. 2, pp. 1312 – 1321, Apr. 2003.

[5] K. Obraczka, and G. Tsuduk, “Multicast Routing Issue in ad hoc Networks,” Proc. IEEE ICUPC 1998, vol. 1, pp. 751 – 756, Florence, Italy, Oct. 1998.

[6] R. Bruno, M. Conti, and E. Gregori, “Mesh Networks: Commodity Multi-hop Ad Hoc Networks,” IEEE Communications Magazine, vol. 43, no. 3, pp. 123 – 131, Mar. 2005.

[7] S. Mao, X. Cheng, Y. T. Hou, and H. D. Sherali, “Multiple Description Video Multicast in Wireless Ad Hoc Networks,” ACM/Kluwer Mobile Networks and Applications, vol.11, no.1, pp. 63 – 73, Jan. 2006.

[8] S. J. Lee, W. Su, J. Hsu, M. Carlo, and R. Bagrodia, “A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols,” Proc. INFOCOM 2000, Tel Aviv, Israel, vol. 2, pp. 565 – 574, Mar. 2000.

[9] W. Wei, and A. Zakhor, “Multiple Tree Video Multicast over Wireless Ad Hoc Networks,” IEEE Trans. on Circuits and Systems for Video Technology, vol. 17, No. 1, pp. 2 – 15, Jan. 2007.

[10] Y. Zhu, and T. Kunz, “MAODV Implementation for NS-2.26,” Systems and Computer Engineering Technical Report, SCE-04-01, Carleton University, Jan. 2004.